

Massimo Di Pierro

Application to Full Professor, 2014

SUPPORTING DOCUMENTS



# Contents

<b>Personal Statement (attached)</b>	<b>3</b>
<b>Curriculum Vitae (attached)</b>	<b>3</b>
<b>Books (attached)</b>	<b>3</b>
<b>Annotated Algorithms in Python</b>	<b>3</b>
<b>web2py Complete Reference Manual</b>	<b>3</b>
<b>web2py Application Development Cookbook</b>	<b>3</b>
<b>I Support Letters</b>	<b>5</b>
<b>Andreas Kronfeld - Fermi National Lab</b>	<b>7</b>
<b>Estia Eichten - Fermi National Lab</b>	<b>9</b>
<b>Steven Gottlieb - Indiana University</b>	<b>11</b>
<b>David Skinner - Berkeley Laboratory</b>	<b>13</b>
<b>Carl Luft - DePaul</b>	<b>15</b>
<b>Fran Boon - Sahana Foundation</b>	<b>16</b>
<b>David Mertz - Python Foundation</b>	<b>17</b>
<b>Robert McCormick - DePaul Information Services</b>	<b>18</b>
<b>Sharon Guan - DePaul ITS</b>	<b>19</b>
<b>Scott Erlinder - DePaul CDM/CIM</b>	<b>20</b>
<b>David Freedman - Creative Artists Associates</b>	<b>22</b>

<b>Lee Mtoti - Verizon</b>	<b>25</b>
<b>II Support Messages</b>	<b>27</b>
<b>III Papers</b>	<b>47</b>
<b>Before 2000</b>	<b>49</b>
<b>Between 2000-2002 (at Fermilab)</b>	<b>110</b>
<b>After 2002 (at DePaul)</b>	<b>201</b>
<b>IV Grant Applications</b>	<b>665</b>
<b>High Performance Computing in the Classroom</b>	<b>667</b>
<b>Scidac (awarded)</b>	<b>704</b>
<b>CAREER Grant</b>	<b>732</b>
<b>Parallel Algorithms for Particle Beam Simulation</b>	<b>775</b>
<b>V Sample Class Notes (not including books)</b>	<b>829</b>
<b>csc299</b>	<b>831</b>
<b>csc309</b>	<b>912</b>
<b>csc312</b>	<b>1074</b>
<b>csc416 (no longer offered)</b>	<b>1149</b>
<b>csc503</b>	<b>1168</b>
<b>etc582</b>	<b>1181</b>
<b>gam450</b>	<b>1332</b>
<b>it378</b>	<b>1368</b>
<b>tdc561 (no longer offered)</b>	<b>1510</b>

<b>CONTENTS</b>	<b>5</b>
<b>VI Other Documents</b>	<b>1553</b>
Semantic Web Work (unpublished work)	1555
QCDUTILS Manual (grant deliverable)	1569
MSCF Advising Guide	1649
DePaul ENGAGE letter	1652
Technology of the Year Article	1655
Selection of published videos	1658



# **Part I**

# **Support Letters**





Fermi National Accelerator Laboratory  
P.O. Box 500 • Batavia, Illinois 60510

E-MAIL: [ask@fnal.gov](mailto:ask@fnal.gov)  
PHONE: +1-630-840-3753  
FAX: +1-630-840-5435

October 3, 2014

Dean David Miller  
School of Computing and Digital Media  
DePaul University  
243 S Wabash Ave  
Chicago IL 60604

Dear Dean Miller:

It is my pleasure to provide my support to the promotion to full professor of **Massimo Di Pierro**. I am a senior physicist in Fermilab's Theoretical Physics Department and a Hans Fischer Senior Fellow in the Institute of Advanced Study of the Technical University of Munich. Massimo and I have known each other since 1999.

As you know, Massimo received a Ph. D. in theoretical high-energy physics from the University of Southampton. We hired him at Fermilab as a postdoctoral associate at that time. Our work in numerical lattice gauge theory is needed to interpret many experiments in high-energy physics. It is one of the most computationally intensive topics in science. For example, my colleague Paul Mackenzie is the PI of a project with the largest award of core-hours at the Argonne Leadership-class Computing Facility. In this letter, I would like to discuss some of Massimo's persistent and ongoing contributions to lattice gauge theory and, thus, high-energy physics.

Massimo is a very good physicist, but as a postdoc his interests turned more to the computing side of our work, leading to his appointment at DePaul. At some stage after Massimo joined DePaul, some experimental physics graduate students asked for a computational introduction to lattice gauge theory. We turned to Massimo, who gave such a fine set of lectures that they encouraged him to prepare a written version. The result [*Int. J. Mod. Phys. A* **21**, 405 (2006)] is a unique introduction to the subject, one that I still recommend.

A key contribution that Massimo made to computational physics is FERMIQCD, a library to facilitate parallel programming of lattice gauge theory. (Here, “QCD” refers to quantum chromodynamics, the most important lattice gauge theory for high-energy physics.) My collaborators and I used this package for many years. Indeed, as Massimo evolved from computational physicist to computer scientist, our rule for including him as an author of physics publications was the role of FERMIQCD in making the research a success. As you can see from Massimo’s publication list, our reliance on his software lasted for many years.

Massimo has also developed several tools to make the day-to-day analysis of lattice QCD data easier. *Vis* and *mc4qcd* are tools for workflow organization and also visualization to help us understand the complicated dynamics of QCD and other gauge theories. A more recent effort is to develop a web interface to databases of lattice-QCD data—we have millions of files approaching petabytes of data, as well as schemes to share the data among researchers the world over. Massimo’s expertise as a computer scientist has been invaluable and with his knowledge of our methodology and scientific aims, unique.

In summary, Massimo’s work and special perspective have made an impact and continue to benefit the field of numerical lattice gauge theory. While I am in no position to judge Massimo’s work in computer science or financial markets, if experts in those fields are as impressed as I am, the case for promotion should be strong and clear.

If you have any questions on Massimo’s contributions to high-energy physics, please do not hesitate to contact me.

Sincerely,



Andreas S. Kronfeld  
Scientist II, Theoretical Physics



Fermi National Accelerator Laboratory  
P.O. Box 500 • Batavia, Illinois • 60510

October 10, 2014

Prof. David Miller  
Dean, College of Computing and Digital Media  
DePaul University  
243 S Wabash Ave.  
Chicago, IL 60604

Dear Prof. Miller,

It is a great pleasure to add my strong support for the promotion of Massimo Di Pierro to a Full Professorship in the Department of Computing and Digital Media at DePaul University. I have known Massimo since he had a postdoctoral position here at Fermilab (1999-2002). He was an exceptional researcher with a wide range of talents. Massimo has more talent for software development than any other researcher I have ever seen at Fermilab.

Massimo contributed much to the world-wide Lattice QCD effort. This lattice effort is directed at obtaining (in unquenched QCD) the masses, decay constants and other coefficients needed to extract the fundamental parameters of the Standard Model from experiment measurements. This is a large project involving the development of improved lattice algorithms, actions, and perturbative corrections. The ultimate goal is to reduce typical theoretical errors to a few percent. These calculations are to be done on a tightly coupled cluster of PCs. Massimo has participated in both analytical and numerical work. He has collaborated most closely with Paul Mackenzie and Andreas Kronfeld in these efforts. He also created a very impressive set of software tools (FermiQCD) for the lattice effort. This QCD software package allows calculations to be distributed (using MPI) over a large number of CPU nodes of a tightly coupled PC cluster. The software is flexible, reliable and easy to use.

While Massimo was at Fermilab, he and I completed a phenomenological study of the spectrum and decays of excited  $B$  mesons. (Physical Review D 64, 114004 (2001)) Massimo quickly digested the extensive literature and skillfully threaded his way through the inherent uncertainties of the phenomenology. He did an extraordinary job, contributing many new ideas and all the programming, allowing us to do things I did not even imagine were possible when we began. As a measure of the value to the field of this paper, I point out that it has become a Famous paper (over 250 citations) in the notation of the arXiv (<http://www.arXiv.org>) and is still cited regularly in new related experimental and theoretical results.



## Massimo's Endorsement

Massimo is an exceptionally intelligent, articulate, and energetic leader, with a demonstrated record of achievement. He has my strongest support for promotion to Full Professor in the Department of Computing and Digital Media at DePaul.

Sincerely yours,

Estia J. Eichten  
Theoretical Physics Department  
Phone: (630) 840-3751  
Email: eichten@fnal.gov

Department of Physics, SW 117  
Indiana University  
Bloomington, IN 47405

October 3, 2014

Dean David Miller  
School of Computing and Digital Media  
DePaul University  
243 S Wabash Ave  
Chicago, IL 60302

Dear Dean Miller:

It is a great pleasure to write this letter in support of **Prof. Massimo DiPierro**, who is being considered for promotion to full professor. I have known Massimo for well over a decade and was very impressed by his abilities right from the start. We first met when Massimo was a postdoc at Fermilab and I was on a sabbatical visit there. It has been wonderful to collaborate with him on a number of projects and also to see his accomplishments in other efforts.

Massimo seems to be able to accomplish on his own what takes a team of other people to do. A prime example is his work on FermiQCD a code suite for which he is the sole developer, as far as I know. He was working on that when I first met him and it has proved a very useful parallel QCD code. Later, Massimo developed a great toolkit for visualization and analysis of lattice QCD data. Quite recently, Massimo was the prime mover behind an upgrade to the NERSC Gauge Connection. This service has been in existence for years, and is one of the original web portals. However, it was in need of major update to be able to interact with the International Data Grid, a world wide service for sharing data coming from lattice QCD calculations. I am not sure whether Massimo developed web2py in order to create the new interface, or he realized that web2py was the right tool for developing the interface, but the result, in any case, is a major improvement in the world-wide usefulness of the Gauge Connection.

Massimo has also been prolific as an author, with books in web2py and this “Annotated Algorithms in Python,” published last year. One of my roles is Associate

Editor-in-Chief of Computing in Science and Engineering, which is published jointly by the American Institute of Physics and IEEE Computer Society. Massimo is a valued member of the editorial board. I greatly enjoy seeing him at editorial board meeting and hearing his ideas for new articles and new columns. He has also contributed articles, one most recently on easy GPU code parallelization using Python and OpenCL.

In the world of elementary particle theory, Massimo is well known and well regarded for a number of calculations on charmonium spectrum, heavy-light meson leptonic and semi-leptonic decay, and  $B$ -anti- $B$  meson mixing. Suffice it to say that we really wish this were his main focus, but we understand his desire to broaden his work. I really have missed having him as a collaborator the past few years.

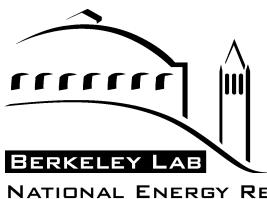
As I reviewed Massimo's CV, I was really impressed by teaching activities, in particular, his development of the Computational Finance program at DePaul. Further, as Massimo is most modest, I was not aware of his awards for developing web2py, though I was aware of how useful it is.

In summary, I want to support Massimo's promotion in the strongest terms. I think he is well deserving of this recognition. I know I would love to have him as a senior colleague on my own campus. Please do not hesitate to contact me if you have any questions.

Sincerely,



Steven Gottlieb  
Distinguished Professor



Letter of Support for Massimo DePiero 10/13/14

From: Dr. David E. Skinner

Dear Massimo,

I am extremely excited about your work towards faculty promotion. We at NERSC look forward to contributing to your work and ongoing collaboration with Lawrence Berkeley National Lab.

As you know, I am the leader of a DOE funded group at NERSC that focuses on meeting DOE Office of Science objectives through data and computing software solutions. The Web2Py approach is an exemplar in allowing our team to leverage capabilities in data-intensive science toward wider audiences. I look forward to supporting this project as a successful and durable bridge between the worlds of high performance computing and Internet technologies. In my 14 years of work at NERSC, I have seen many opportunities and frameworks to bridge these worlds. Web2Py ranks highly among these for a few reasons. 1) It has well crafted software design which is cognizant of the needs of both developers and users 2) It speaks the high performance dialects we need being based in python 3) It offers a rich of set of out-of-the-box functionality that bootstraps projects keeping them focused on science. Web2Py is software that works and a rare gift to the HPC community. Thank you.

My work and research has a long history tied to computing and data science. My group's focus on meeting the end-user needs from science communities may be useful in how you extend Web2Py. Scalable data practices, high performance analysis, and user interfaces that provide excellent user experience to scientists are key overlapping areas between your work and the ongoing mission of my group at NERSC. As an advisor I will eagerly assist in helping Web2Py meet the needs of the Office of Science user community. In addition, I am happy to lend my expertise to support this important research in overcoming future challenges.

Lastly I see many opportunities outside HPC for Web2Py as a web framework and as CS educational tool. My best wishes for you on your career path.

Best Regards,

Dr. David Skinner, Ph.D.

Strategic Partnerships Lead, NERSC  
[www.nersc.gov](http://www.nersc.gov)

1 Cyclotron Road, MS943-220  
Lawrence Berkeley National Lab  
Berkeley, CA 94720



# DEPAUL UNIVERSITY

---



**Department of Finance**  
1 East Jackson Boulevard  
Chicago, Illinois 60604-2287  
312/362-8826  
FAX: 312/362-6566

October 1, 2014

The Dean of the College of Computing and Digital Media  
DePaul University  
One East Jackson  
Chicago, Illinois 60604

To Dean David Miller of the School of CDM:

It is my pleasure to write this letter supporting Dr. Massimo DiPierro's application for promotion to Full Professor. I was Dr. DiPierro's counterpart in the Kellstadt Graduate School of Business when we created the Master of Science Degree in Computational Finance. It was my privilege to work with him when the program was established, and it continues to be my privilege to work with him on a regular basis to maintain the program and revise it as needed. Massimo always is thoughtful and well prepared and willing to engage in a free exchange of ideas about the various issues that affect the program. Working with Massimo has allowed me to witness first hand his passion for teaching excellence and his commitment to improving the Computational Finance curriculum. Massimo is a firm believer in the concept that students learn by doing and that a curriculum should be rigorous in both theory and application.

Massimo's commitment to the program's continued success is demonstrated clearly by his interactions with the students. He is the one who provides curriculum advice to all CDM and Commerce students enrolled in the program.

The Computational Finance program's success stems directly from Massimo's commitment to the Computational Finance Program's academic quality and students, and is the manifestation of his leadership skills. I believe that possession of such skills, and the willingness to employ them, are critical attributes for all full professors. Dr. Massimo DiPierro has demonstrated that he has the tools to be a full professor; I strongly recommend that he be promoted.

Sincerely yours,

A handwritten signature in black ink that reads "Carl F. Luft Ph.D."

Carl F. Luft, Ph.D.  
Associate Professor of Finance  
Academic Professor, Arditti Center for Risk Management

8A Hardwick Road  
Hethe  
Bicester  
OX27 8EY  
United Kingdom  
25<sup>th</sup> September 2014

To the DePaul Tenure and Promotion Committee,

I would recommend Prof. Massimo DiPierro for promotion to Full Professor because of his work on web2py.

This software has revolutionised the ease of developing secure and powerful web applications which is why the Sahana Software Foundation chose it for the base of the Sahana Eden humanitarian management platform.

Not only has Massimo built the original version but he has mentored a growing community of developers to both make use of, maintain and extend the application. His commitment to backwards-compatibility has helped enormously with our ability to focus on our own mission rather than constantly having to update for framework changes. He has also significantly extended the tool specifically for needs which we have raised – one time personally (the Scheduler) and one time by finding another appropriate contributor (Spatial database support). I have the utmost respect for Massimo's Vision, Passion, Humanity and attention to Detail. He is a great leader who can inspire students and train them to become as fully proficient as they are able.

Thanks for your consideration,

Fran Boon  
Chairman of the Sahana Software Foundation Product Development Committee

A handwritten signature in black ink, appearing to read "Fran Boon".

The Sahana Software Foundation is dedicated to the mission of saving lives by providing information management solutions that enable organizations and communities to better prepare for and respond to disasters.

900 Wilshire Blvd, Suite 1500, Los Angeles, CA 90017  
<http://sahanafoundation.org>

David Mertz, Ph.D.  
Director, Python Software Foundation  
6637 W 6th St, Los Angeles CA 90048  
mertz@gnosis.cx

September 27, 2014

### **Recommendation for Massimo DiPierro**

I am pleased to have worked with Massimo in my capacity as a Director of the Python Software Foundation, and as the designer and administrator of its voting system and voting security protocol. His volunteer contributions to the Foundation, and to the Python programming language community, have been invaluable, and have been enabled by both his technical expertise and his dedication to community service.

The Python Software Foundation (PSF) is a 501(c)(3) non-profit corporation whose mission is to promote, protect, and advance the Python programming language, and to support and facilitate the growth of a diverse and international community of Python programmers.

In particular, as part of the operation of the PSF, we conduct various elections among our membership on a recurring basis. For this purpose, I designed a security and cryptographic protocol for conducting elections, which is able to guarantee anonymity, collective verifiability, and security from tampering (including detection of tampering by an election administrator). In developing this protocol, I relied on my prior experience as Chief Technology Officer of the non-profit Open Voting Consortium, in which capacity I conducted and published security research around election systems.

Massimo volunteered to implement this protocol as a web-based system, using the web2py framework, a widely used Free Software Python web framework, of which he is principal author. In addition, his E-Vote system (now hosted as [vote.python.org](http://vote.python.org)) has been the voting system used by the PSF since 2012; it was subsequently also adopted by The Open Source Initiative (another community-oriented non-profit supporting Free Software).

Working with Massimo, we did more than simply implement a previously published protocol; he and I were able to add a number of improvements to the protocol, develop more sophisticated threat models, and provide additional verifiability guarantees. Recent versions of the E-Vote software also support directly a variety of vote tabulation models, including those utilizing ranked-preference voting.

Massimo is an elected member of the PSF—under recent revisions to our Bylaws, this converts his status to a Fellow of the Foundation, one of 216 worldwide. As well as being a PSF Fellow and having contributed the above described E-Vote software to the Free Software community, Massimo created the PyCon conference registration system that was used in 2009 and 2010. He has given one tutorial and two competitively accepted talks at previous PyCon conferences.

A handwritten signature in black ink, appearing to read "David Mertz".

# DEPAUL UNIVERSITY



Information Services  
1 East Jackson Boulevard  
Chicago, Illinois 60604-2201

October 2<sup>nd</sup>, 2014

Dear Dean Miller:

I am writing to you in my capacity as co-chair of the Teaching, Learning Technology (TLT) Committee.

This letter is provided at the written request of Dr. Massimo DiPierro who asked me to serve as a reference on his behalf.

Dr. DiPierro has served as CDM's TLT representative for the last seven years, and for five of these years Dr. DiPierro was also co-chair of the committee. He has been an active member of the committee, attending the monthly meetings regularly, bringing issues to the committee from fellow faculty members, contributing to the discussions and voting on items that were raised.

During Dr. DiPierro's tenure the technologies reviewed by the committee include, but were not limited to: Content Management Systems, Learning Management Systems, ePortfolio solutions, Video Streaming Solutions, and other assorted classroom technologies.

Dr. DiPierro was instrumental in the selection of Desire2Learn for DePaul, and was a key resource in championing its use across the faculty, helping shepherd the successful cutover from our legacy Blackboard system.

It has been a pleasure to serve with Dr. DiPierro these last seven years.

If you would like to discuss this further, please feel free to contact me.

Sincerely,

A handwritten signature in blue ink, appearing to read "R. McCormick".

Robert McCormick

Vice President for Information Services

Co-chair of TLT

# DEPAUL UNIVERSITY



**Faculty Instructional Technology Services**  
**2350 N Kenmore Ave**  
**Chicago, IL 60614**  
**(312) 362-6812**  
**[fits.depaul.edu](mailto:fits.depaul.edu)**

Oct 7, 2014

Dr. David Miller  
Dean of College of Computing and Digital Media  
DePaul University

Dear Dr. Miller,

It is my pleasure and privilege to write this letter to support Dr. Massimo DiPierro's application for full professor. I had known Dr. DiPierro since he assumed the co-chair role of the Teaching, Learning and Technology (TLT) committee about seven years. Having served as a TLT co-chair for six years myself and as the head of the faculty technology support unit, I worked closely with Massimo in transitioning TLT leadership and in identifying issues for TLT discussion.

In addition to TLT, Massimo and I have worked together in one of the expert terms during the vision2018 planning process in 2011. In this Educational Innovation and Instructional Technology (EIIT) team, we worked closed with experts from various areas of the university to establish goals and objectives for technology integration at DePaul. With professional knowledge of computing and thorough understanding of both technology development and our institution's technological needs, Massimo played a vital role in this team. As the result, the report drafted by our team was highly praised by the Vision2018 taskforce and was used as a critical document to help developing the Vision 2018 goals.

In reviewing Dr. DiPierro's application for full professor, I would appreciate your consideration of the services he has provided for our university and the contribution he has made to improve teaching and learning through the use of technology.

Sincerely,

A handwritten signature in black ink that reads "Sharon Guan".

Sharon Guan, Ph.D.  
Director, Faculty Instructional Technology Services  
DePaul University  
[xguan@depaul.edu](mailto:xguan@depaul.edu)  
773-325-7726

# DEPAUL UNIVERSITY

Sept 29, 2014



To:

Dean David Miller  
College of Computing and Digital Media  
DePaul University  
243 S. Wabash  
Chicago, IL 60604

Attn:

Promotion and Tenure Committee for the School of Cinema and Interactive Media

College of Computing  
and Digital Media  
243 South Wabash Avenue  
Chicago, Illinois 60604-2302  
312/362-8381  
FAX: 312/362-6116

When I found out that Professor Massimo Di Pierro was submitting materials for the position of Full Professor, I was very happy indeed to write this letter of support for him. Massimo has been one of the most supportive, inquisitive and generous people I have met in my time on the Teaching, Learning and Technology Committee.

I was invited as a representative for the Digital Cinema Program (later to become the school of CIM) in 2007. Massimo was co-chair at the time and remained co-chair until 2012. We are both still members of the committee.

Over that time the committee has monthly researched and evaluated many technologies including Content Management Systems, Learning Management Systems, e-Portfolio solutions, Video Streaming Solutions. The committee was responsible for the selection of D2L as a replacement for Blackboard.

Massimo presented to committee with various and applicable strategies, programs and solutions to the growing needs of our faculty and students. He was the most pro-active yet down to earth faculty member of the committee and served as the most knowledgeable source for information for all involved.

My specialty in being on the committee was to be an advisor to the management of the expanding use of video and audio for teaching and archiving. Massimo and I had many discussions on this matter and he was both knowledgeable of the hurdles and possibilities for furthering the use of visual and audio materials across the curricula of the University. I felt we worked in a peer-to-peer relationship that was respectful, stimulating and, many times, almost joyful. He made the work very enjoyable.

Massimo's work with deciding on a new content management system (D2L) was tireless. He evaluated all systems thoroughly before presenting them to the committee, giving both their strengths and weaknesses, in a very objective manner for review- and this was very appreciated for the uninitiated.

I would not hesitate for a moment in recommending Professor Di Pierro for full professor. He embodies technical excellence, a work ethic that is beyond question and a Vincentian state of mind that all here at DePaul who know him can attest.

He has my full support for promotion to full professor.

If you need any further information or wish to speak to me on any matters relating to this letter, you can contact me below.

Sincerely,



Scott Erlinder  
Assistant Professor  
College of CDM  
Digital Cinema Program  
DePaul University

David Freedman  
Creative Artists Agency  
2000 Avenue of the Stars  
Los Angeles, CA 90067

Oct 10, 2015

To Whom It May Concern:

I am writing this letter to share the value of Massimo's Web2py Framework and open source software to our company, Creative Artists Agency (CAA). Headquartered in Los Angeles, CAA is a global entertainment talent and sports conglomerate representing top talent across motion pictures, television, music and all of the major sports. In addition to our representation businesses, we have a number of brand-focused businesses and manage a slate of entertainment-related venture capital funds. In recent years, we have undergone a significant global expansion and business diversification.

I am a 15 year veteran technology executive at Creative Artists Agency. We use upward of a dozen custom applications to operate our business that represent millions of dollars in development. For the first decade of my tenure, we closely aligned our technology strategy with a few major technology vendors including Microsoft, Cisco, EMC , Dell and HP. In recent years, we've largely migrated away from these vendors in favor of cloud-hosted open source software, including Web2py, for all of our application development. This is the norm for web-based consumer software companies and increasingly the norm for information technology operations at large corporations. For CAA, this shift is driven by a number of factors. First, we want to spend as much time as possible focused on building software that helps us better service our clients. Open source completely eliminates complex licensing purchase issues that encumber productivity. Further, we find the open source community to be far more responsive to our business needs than traditional enterprise vendors. Where we have unique needs, we are able to make our own contributions to open source projects rather than spending unproductive time trying to convince a vendor to eventually include our requirements in a future product release. For CAA, this represents a dramatic shift in software delivery accountability. With open source, we are able to fully own our outcomes. There are no outside parties on whom we

depend though we often receive excellent support from the open source community.

CAA's open source software development is currently based on Node.js and Python (Web2py in particular). We chose Web2py against tens of other web frameworks. We found Web2py to offer a unique blend of simplicity and scalability. Our developers are incredibly productive on the Web2py platform because it offers well architected shortcuts to some of the most common web development scenarios. Unlike other frameworks, these shortcuts do not constrain our developers to a specific way of doing things. In cases where we want to take a different approach, we can do so without consequence.

The approach described above is somewhat controversial in the Python community as it undermines one of its core principals. Massimo has provided humble leadership in getting the community to reconsider this potential flaw in philosophy.

Web2py is thoroughly and thoughtfully documented. Its structure is simple and pragmatic – among the best documented frameworks in open source. This makes it easy to ramp up developers that are unfamiliar with the framework. I was able to onboard a team of 6 engineers in Karachi, Pakistan in the course of one week with little effort on my part.

Open source software is often challenged by the risk of abandonment because open source tools are often authored by one or two individuals. When their priorities take their focus off of the tools they publish, maintenance is left to those who consume their tools. Massimo has effectively mitigated the risk of abandonment by building a strong community of Web2py contributors with a very active online community forum. He further strengthens and engages this community through an annual Web2py conference.

Information security is a top priority for CAA. Web2py inherently addresses many common security flaws and provides a level of insurance against inexperienced developers making mistakes that could compromise security.

Open source software is very much the engine of the new digital economy and is at the center of CAA's software development strategy. Massimo's Web2py framework is among the best in open source and is being chosen by

companies like CAA against commercial products that benefit from millions of investment dollars. We are hugely thankful for his creation!

Sincerely,

*David Freedman*

David Freedman



LEE 'Simba' MTOTI  
700 Hidden Ridge, HQW02B16  
Irving, Texas 75038

Phone 972-791-7471  
[Simba.mtoti@verizon.com](mailto:Simba.mtoti@verizon.com)

09 October 2014

To the Review Committee:

I am writing to you in support of Associate Professor Massimo Di Pierro and his desire to be a full professor at DePaul University – School of Computing.

I have known Massimo for the last 12 years. During this period, he has been very instrumental in influencing my academic and professional career. While a student at DePaul, Massimo was a great resource and was always willing to guide and mentor me in my academic career.

As a professional, I have reached out to Massimo for assistance in solving some problems I have faced in my career.

An example of such a project was:

The design, architect of a decentralized system for risk assessment, risk management and decision analysis. In that project we were looking for a full-stack web framework that could address our needs. Massimo came to the rescue by adding enhancements to web2py.

Having just received an Advanced Computer Security Certificate from Stanford, I can definite compare Massimo to the likes of Dan Boneh and Neil Daswani in his academic contribution to my professional field of web application security.

As an alumnus, I strongly believe that DePaul needs professors like Massimo because he is helping the reputation of security program in the School of Computing. I enthusiastically recommend Massimo Di Pierro as an applicant for full professorship at DePaul – School of Computing.

Please do not hesitate to contact me, if you require any further information.

Yours faithfully,

Lee Mtoti

Distinguished Member of Technical Staff - Apps Dev Security



## **Part II**

# **Support Messages**



# #massimo4full tweets

## (from former students, web2py users, and collaborators)

Noah Gift Bot on Thu Oct 09 says

*#massimo4full A great guy who is full of sanity, even in the darkest of hours. I would trust this guy when the zombie apocalypse hits.*

<https://www.twitter.com/noahgift/status/520044193121439744>

Brian Ray on Thu Oct 09 says

*To learn one best surround themselves with learned, passionate, and professional #massimo4full*

<https://www.twitter.com/brianray/status/520044322658713600>

Lian Yang on Thu Oct 09 says

*#massimo4full*

<https://www.twitter.com/LianYang3/status/520045035799072768>

La Guitarra y Vino on Thu Oct 09 says

*#massimo4full Thanks Massimo for your great web2py book and for your great advice and knowledge #MassimoRocks #Web2py*

<https://www.twitter.com/laguitarrayvino/status/520045274996039681>

Emine Aydil Ozer on Thu Oct 09 says

*#massimo4full Massimo Di Pierro is a knowledgeable professor at DePaul School of Computing. He tries to make best use of his classes.*

<https://www.twitter.com/EmineAydilOzer/status/520045892012085248>

Rodrigo Lira on Thu Oct 09 says

*It's a pleasure help you massimo. You've done a wonderful work with wep2py. It's amazing! All py community are supporting you!*

*#massimo4full*

<https://www.twitter.com/rodrigoclira/status/520045936152567809>

Sebastian Demian on Thu Oct 09 says

*#massimo4full Graduating next month. Looking back, Massimo was one of the best instructors I had at DePaul. Wish there were more like him.*

<https://www.twitter.com/sebidemian/status/520046019828916224>

Samuel Sowah on Thu Oct 09 says

*#massimo4full Your creation, Web2py, has become my greatest dev asset. I'd say it's the best web dev framework that exists. Thank you sir.*

<https://www.twitter.com/SASOGEEK/status/520046428182568961>

Mark GhostTap Studio on Thu Oct 09 says

*As a student, there was one professor that stood out @ depaul. Now as a teacher myself, I strive to emulate his style! Ty! #massimo4full*

<https://www.twitter.com/nightst4r/status/520047163502370816>

李翰辰 on Thu Oct 09 says

*#massimo4full It is best experience to have Massimo as our director and teacher for computational finance program. Creative and talented*

<https://www.twitter.com/HanchenLi/status/520047265986011136>

Harin Gupta on Thu Oct 09 says

*He taught me the language of cmputrs, smethin I thot I cud nvr undrstnd. As a result I m minoring in cmputr sci alng wth math*

*#massimo4full*

<https://www.twitter.com/HarinGupta/status/520048270848966657>

Jeff Sheffield on Thu Oct 09 says

*#massimo4full Massimo's work on web2py has been inspirational.*

<https://www.twitter.com/jeffsheffield/status/520048483999686656>

adhika kamaludin on Thu Oct 09 says

#massimo4full was my professor in physics. The magic formula for calculating physics is very useful even after I worked in game studio  
<https://www.twitter.com/adhikakamaludin/status/520048485169504257>

Jorge Puente-Sarrín on Thu Oct 09 says

#massimo4full Massimo, IMHO, the creator of the more pythonic full stack web framework!  
<https://www.twitter.com/puentesarrin/status/520048683316809728>

Abdullah H. AlJaber on Thu Oct 09 says

#massimo4full he did web2py conferences at #DePaul which opened my eyes to #Python. I Highly recommend him.  
[https://www.twitter.com/Al\\_Jaber/status/520049391646683138](https://www.twitter.com/Al_Jaber/status/520049391646683138)

Maurice Ling on Thu Oct 09 says

#massimo4full Support on Web2py by Massimo is instrumental to enable me to publish 2 papers so far. I know that support is never far away.  
<https://www.twitter.com/mauriceling/status/520051034807205889>

Mark Smith on Thu Oct 09 says

#massimo4full massimo taught me to love algorithms & 2 persist to find the perfect pattern. Over the years he has offered excellent advice.  
<https://www.twitter.com/smittyccb10/status/520051459463716864>

Adam Conkey on Thu Oct 09 says

Massimo took time after a night class to help me understand a problem from my personal research. He's been a great help! #massimo4full  
<https://www.twitter.com/adamconkey/status/520052213515689984>

Tammi Titsworth on Thu Oct 09 says

#massimo4full I've worked with him on several articles/issues for CiSE magazine--he always delivers excellent, insightful work.  
<https://www.twitter.com/tammitopia/status/520052367568674817>

Trinh Nguyen on Thu Oct 09 says

#web2py was my first #python framework in web development. Super easy, super clean! Thanks Massimo Di Pierro! #massimo4full  
<https://www.twitter.com/dangtrinhnt/status/520053262397894656>

Athir Mahmud on Thu Oct 09 says

A wonderful colleague, brilliant computer scientist, & enthusiastic professor. @CDMDePaul #massimo4full  
<https://www.twitter.com/AthirMahmud/status/520053458263871488>

Phyllis Gunning on Thu Oct 09 says

#massimo4full Massimo's innovative contributions to the Lincoln Elementary PTO have brought us into the future! A valuable community partner  
<https://www.twitter.com/PhyllisGunning/status/520053511581487104>

Alfonso de la Guarda on Thu Oct 09 says

#massimo4full one of the greatest modern hackers in the world... your contributions to the #Python community are very important!  
<https://www.twitter.com/alfonsodg/status/520053552626950147>

Guido van Rossum on Thu Oct 09 says

Thanks for rowing upstream with web2py, your Python contributions, and intellectually stimulating conversations! #massimo4full  
<https://www.twitter.com/gvanrossum/status/520054305085071360>

Zhenyang Lu on Thu Oct 09 says

#massimo4full As a MSCF Alumni, I vote for you to be tenured professor of DePaul! sei il migliore, professore DiPierro! - Zhenyang  
<https://www.twitter.com/zz71703486/status/520054353311178752>

David Watson on Thu Oct 09 says

I've read Massimo's book and used web2py to build two startups. His work and support have been invaluable to me. #massimo4full  
<https://www.twitter.com/davidthewatson/status/520054395338518529>

Nan Li 1qop1 on Thu Oct 09 says

#massimo4full Massimo is an creative, talented, enthusiastic professor. I have not touched twitter in like 4 years. I highly recommended him  
<https://www.twitter.com/1qop1/status/520055129152974848>

Erion Omeri on Thu Oct 09 says

#massimo4full Really excited for prof Massimo DiPierro! One of the best CS profs at @DePaulU Loved his class and #web2py is super amazing!

<https://www.twitter.com/WeddingGreenBay/status/520055695203635200>

Juan B. Cabral on Thu Oct 09 says

@mdipierro helped me to make a great conf at PyConAr2012 and also taught me web2py that nowadays i use 4 teach web progamming  
#massimo4full

<https://www.twitter.com/JuanBCabral/status/520056006554824704>

Caleb Hattingh on Thu Oct 09 says

#massimo4full Massimo Di Pierro is a force of nature on the web2py project, and I wholeheartedly support his promotion to full professor.

[https://www.twitter.com/caleb\\_hattingh/status/520057239973797888](https://www.twitter.com/caleb_hattingh/status/520057239973797888)

Huu Phuoc Tran on Thu Oct 09 says

Massimo is an inspirational professor, and a true individual innovators. DePaul students definitely need you !!! #massimo4full

<https://www.twitter.com/phuocidi/status/520058146622951425>

Scott Corsey on Thu Oct 09 says

#massimo4full

<https://www.twitter.com/Corsey5/status/520059812445319169>

Sebastián Bassi on Thu Oct 09 says

@JuanBCabral veo q tuvo el endorsment d Guido, deberia ser mas q suficiente. #Massimo4full

<https://www.twitter.com/sbassi/status/520059969795031040>

Dandan Guo on Thu Oct 09 says

Prof. Dipierro tries his best to help his student. He built a group to public the job information and answer any question. #massimo4full

<https://www.twitter.com/DananGuo/status/520060136383987712>

Touko on Thu Oct 09 says

#massimo4full Thanks for going out of your way to help with your knowledge and expertise. Really sharp people's help isn't easy to find!

<https://www.twitter.com/ToukoAkimoto/status/520060770818617345>

Alex McFerron on Thu Oct 09 says

#massimo4full Prof. Massimo helped my career and education at DePaul tremendously by suggesting project ideas that fit my interests

<https://www.twitter.com/alexMac05/status/520061819415568384>

Noor Syed on Thu Oct 09 says

#massimo4full

<https://www.twitter.com/nsyed4/status/520062545504124929>

Eric Bratt on Thu Oct 09 says

#massimo4full Massimo is hands-down the best prof I've had. I plan to apply what I learned in csc438 to help real ppl solve real problems.

[https://www.twitter.com/eric\\_bratt/status/520062818146877440](https://www.twitter.com/eric_bratt/status/520062818146877440)

Sean Reifsneider on Thu Oct 09 says

Your presentation at ChiPy a few years ago on web2py really changed my perspective on aspects of web programming. #massimo4full

<https://www.twitter.com/jafo/status/520065009972281344>

Miles Rosenberg on Thu Oct 09 says

#massimo4full Massimo steered me into IT security audit conc., a field I absolutely love. Always supportive of students. Great!

<https://www.twitter.com/MilesEvan7/status/520065222032515074>

Julio F Schwarzbek on Thu Oct 09 says

#massimo4full - Not only an excellent speaker, but also a mentor, a python zen master that I am proud to follow, godspeed!

[https://www.twitter.com/techfuel\\_net/status/520065916650811392](https://www.twitter.com/techfuel_net/status/520065916650811392)

Dustin DeMoss on Thu Oct 09 says

#massimo4full Massimo is a great thinker and leader. He has helped me numerous times, especially when I was in need. A true academic!

<https://www.twitter.com/DustinDeMoss/status/520066083214983168>

Li-Wey Lu on Thu Oct 09 says

*I was very impressed with the way Massimo organized and controlled the Web2Py All-Day-Conference. We need more people like him #massimo4full*

<https://www.twitter.com/LeewayLoo/status/520067117769437184>

Saurabh Kumar on Thu Oct 09 says

*@mdipierro Web2py is awesome, great work :) #massimo4full*

[https://www.twitter.com/\\_saurabh\\_kumar/status/520067251370606592](https://www.twitter.com/_saurabh_kumar/status/520067251370606592)

shreddd on Thu Oct 09 says

*Props to @mdipierro for his excellent work with web2py that supports the QCD portal (<http://t.co/LbqzLnoIst>) at NERSC #massimo4full*

<https://www.twitter.com/shreddd/status/520069095350231041>

Percy Hatcherson on Thu Oct 09 says

*I didn't know it at the time, but Massimo's course on web app development was my first introduction to my current career #massimo4full*

[https://www.twitter.com/primitive\\_type/status/520070724782784512](https://www.twitter.com/primitive_type/status/520070724782784512)

Shadow Fiend on Thu Oct 09 says

*#massimo4full Many thx to Massimo 4 dev web2py i used 4 web dev, a super ez&#amp; powerful frmwk. Also super active&#amp; helpful in t web2py group.*

<https://www.twitter.com/bigmm520/status/520071714357522434>

Shabbir on Thu Oct 09 says

*Depaul had many great teachers, but Massimo stuck out the most. Good luck with everything the future holds. #massimo4full  
#web2pyinventor*

<https://www.twitter.com/shabzcohelp/status/520071810168008705>

Goodwill Bits on Thu Oct 09 says

*I was proud to invite @mdipierro to 2012 PyCon PyWeb summit, given his amazing work bringing WebDev to students #massimo4full*

<https://www.twitter.com/goodwillbits/status/520072657807495168>

Adriano Farano on Thu Oct 09 says

*@mdipierro thank you so much for what you've done for our startup #massimo4full*

<https://www.twitter.com/farano/status/520073068341780480>

killiantobin on Thu Oct 09 says

*Great teachers are some of the most influential people in our lives. I jumped at every chance to take your classes. Thanks, #massimo4full*

<https://www.twitter.com/killiantobin/status/520073642625880064>

Siyu Wang on Thu Oct 09 says

*#massimo4full He is a very good teacher*

<https://www.twitter.com/wangsiyuyy/status/520075195306872833>

malaniz on Thu Oct 09 says

*#massimo4full thanks for your support in my PhD, you told me that with effort I will be a good researcher. I am doing the best. Cheers!*

<https://www.twitter.com/ingnucious/status/520075755846656000>

jeremy johnson on Thu Oct 09 says

*Web2Py is a great framework (with good docs) for learning to build websites with python. Thanks massimo #massimo4full*

<https://www.twitter.com/j1z0/status/520076435780681730>

Jonathan Huerta on Thu Oct 09 says

*shout out to @mdipierro for helping me get started with programming and understanding the basics to coding #massimo4full*

[https://www.twitter.com/Jon\\_Huerta\\_/status/520079770076581888](https://www.twitter.com/Jon_Huerta_/status/520079770076581888)

Sihong Liu on Thu Oct 09 says

*#massimo4full Professor Massimo is one of the most knowledgeable professor in DePaul University. His classes are very useful for my career.*

<https://www.twitter.com/SihongLiu/status/520081418521567232>

Immo Salo on Thu Oct 09 says

*Massimo, thanks for all the help with #Web2py on Google App Engine! Helped me with my 1st book about Cloud Computing in 2010  
#massimo4full*

<https://www.twitter.com/immon/status/520081745522073600>

Wynn Drahorad on Thu Oct 09 says

*With Prof. DiPierro as your teacher, you know he has your best education in mind. Enthusiastic, relatable, a great instructor.  
#massimo4full*

<https://www.twitter.com/Blaze6181/status/520081791671992320>

Yan Wu on Thu Oct 09 says

*#massimo4full Massimo got me interested in computational finance. I just landed a job at Milliman Finance Risk Management thanks to him.*

<https://www.twitter.com/yan5/status/520083515459645440>

Jordi Molins on Thu Oct 09 says

*#massimo4full Massimo was helpful in developing his C++ tools to analyze the Ising model and QCD.*

<https://www.twitter.com/MolinsJordi/status/520089447627370497>

Tamás Sebestyén on Thu Oct 09 says

*Massimo and #web2py made me re-discover the power of web apps after many years #massimo4full*

<https://www.twitter.com/akusebi/status/520091390781689856>

scott on Thu Oct 09 says

*#massimo4full one of the best professors i've ever had! fully support his promotion to full professor.*

<https://www.twitter.com/scosant/status/520091927274721280>

Joel Vasallo on Thu Oct 09 says

*One of the most impactful professors I've had @DePaulU! His contributions and dedication to web2py and his students is top!*

*#massimo4full*

[https://www.twitter.com/\\_\\_jvasallo\\_\\_/status/520093906227445760](https://www.twitter.com/__jvasallo__/status/520093906227445760)

Valter Foresto on Thu Oct 09 says

*#massimo4full Massimo has created a great web framework for the human being*

<https://www.twitter.com/valterforesto/status/520097415173378048>

rahulserver on Thu Oct 09 says

*#massimo4full Thanks for the awesome framework web2py. Man! you've changed the way we devs build web apps. You've really brought a #Change.*

<https://www.twitter.com/rahulserver/status/520104120829497344>

Gael Princivalle on Thu Oct 09 says

*#massimo4full Thanks a lot to Massimo, web2py give me everyday the tools that I need for web-based app. in a perfect Open Source philosophy.*

<https://www.twitter.com/GPrincivalle/status/520110048094060544>

Marco Passanisi on Thu Oct 09 says

*Thank Massimo for having conceived web2py that it made my job easier. Go #massimo4full*

<https://www.twitter.com/MarcoPassanisi/status/520115213505282048>

Al on Thu Oct 09 says

*Massimo is one of my favorite professors I took at DePaul. He is very intelligent and an outstanding teacher in programming.*

*#Massimo4Full*

<https://www.twitter.com/Abdelati786/status/520121305195368448>

Ahmed Soliman on Thu Oct 09 says

*I believe Massimo helped many to explore the Python programming language through his fantastic web2py framework. #massimo4full*

<https://www.twitter.com/AhmedSoliman/status/520121983582093312>

Francisco A. Tapias on Thu Oct 09 says

*#massimo4full I make my living programming with Massimo's technologies, Web2Py, the best web python framework of the world.*

<https://www.twitter.com/gestionexperta/status/520124754972250112>

alfredo vittoria on Thu Oct 09 says

#massimo4full Thanks for the support for the improvement of the knowledge of web2py  
<https://www.twitter.com/alfvit66/status/520126279211692033>

Fran Boon on Thu Oct 09 says

With @Web2Py the @SahanaFOSS community is enabled to rapidly develop flexible solutions for disaster response. #massimo4full  
<https://www.twitter.com/franboon/status/520126542546886656>

Григорий Хацевич on Thu Oct 09 says

#massimo4full Thank you for your contribution to make web-development accessible for everyone, and also for your social responsibility!  
[https://www.twitter.com/g\\_hatsevich/status/520127703022702592](https://www.twitter.com/g_hatsevich/status/520127703022702592)

Avi Abramovitch on Thu Oct 09 says

Just got my new QA automation job at RedisLabs. So many thanks to Massimo+team+web2py+usergroup for months of educating me.  
#massimo4full  
<https://www.twitter.com/aviavia/status/520131380164833282>

Kev on Thu Oct 09 says

#massimo4full - Yes!  
<https://www.twitter.com/snr0db/status/520132158099173376>

Bavandlapally Krishn on Thu Oct 09 says

#massimo4full  
<https://www.twitter.com/bkrishna2020/status/520137457031847936>

Athos Georgiou on Thu Oct 09 says

#massimo4full Dr Massimo DiPierro teaches with confidence and has all the knowledge and experience to back him up. Top Notch Professor!  
<https://www.twitter.com/athosg82/status/520139214365528064>

Ramkrishan Bhatt on Thu Oct 09 says

#massimo4full web2py made me to explore the software development and put into the pen drive with UBUNTU , Its more focus and light and easy  
<https://www.twitter.com/ramkrishanbhatt/status/520141307788816385>

Habtom Gesgis on Thu Oct 09 says

#massimo4full I did some spatialization on web2py. Apart from his creation of the powerful web2py, Massimo is quick to support. Thank you.  
<https://www.twitter.com/DrGesgis/status/520143278449369088>

Jay kelkar on Thu Oct 09 says

#massimo4full Most deserving. Knowledgeable, dedicated, helpful, willing to take a stand. And available. Grace your uni. he will.  
<https://www.twitter.com/jkelkar/status/52014696553995648>

Youcheng Lin on Thu Oct 09 says

#massimo4full I fully support to his promotion to a full professor for his great knowledge and dedicated attitude in teaching. Youcheng  
<https://www.twitter.com/YouchengL/status/520148160425893889>

Igor Gassko on Thu Oct 09 says

#massimo4full Massimo's web2py initiative is brilliant! I wish he was my professor during my CS degree.  
<https://www.twitter.com/igorgassko/status/520148480421941251>

Gamefulness on Thu Oct 09 says

Web2Py allowed me to build my final project to achieve my bachelor: An online development environment for computer vision  
#massimo4full  
<https://www.twitter.com/Gamefulness/status/520162739142942720>

Matt Spraggs on Thu Oct 09 says

Thanks for providing some great tools for generating HPC code, e.g. OCL and QCL #massimo4full  
<https://www.twitter.com/Orentago/status/520163791279910914>

Nikhil Kodikar on Thu Oct 09 says

#massimo4full I wish I had a professor who had built a world class framework like Web2py. He'd be a great asset to any univ. Embrace him !!  
<https://www.twitter.com/NikhilKodilkar/status/520171924807487489>

Ross Peoples on Thu Oct 09 says  
*Learned so much from @web2py over the years. Was the first OSS project I contributed to and I carry its lessons with me.* #massimo4full  
<https://www.twitter.com/rosspeoples/status/520174215279177728>

Nico de Groot on Thu Oct 09 says  
*#massimo4full I've met Massimo at the Pycon conference, his personal and Computer Science qualities are reflected in Web2py & the community*  
<https://www.twitter.com/ncde groot/status/520175308851249153>

Tim Richardson on Thu Oct 09 says  
*web2py greatly lowers the barriers to learning advanced modern web techniques. Huge economic benefits. Great outcome.* #massimo4full  
<https://www.twitter.com/timrdsn/status/520176363035033600>

Arvind Gupta on Thu Oct 09 says  
*@rosspeoples @web2py is the simplest framework for webapps with high productivity & low learning curve*  
<http://t.co/DVMJahD5Tz> #massimo4full  
<https://www.twitter.com/arvindcg/status/520185796096053249>

Adnan Smajlovic on Thu Oct 09 says  
*With security first on his mind he built fantastic web2py. We all won. Generations of his future lucky students will win too!* #massimo4full  
<https://www.twitter.com/adnansmajlovic/status/520185884243537922>

Tomek Krasuski on Thu Oct 09 says  
*#massimo4full he deserves it :)*  
<https://www.twitter.com/TKrasuski/status/520186818860306432>

Fabio Lapuinka on Thu Oct 09 says  
*#massimo4full*  
<https://www.twitter.com/lapuinka/status/520191683900616704>

Jay Martin on Thu Oct 09 says  
*Massimo Di Pierro, through his elegant sharing of CS knowledge and experience, has inspired my own learning more than anyone!* #massimo4full  
<https://www.twitter.com/webapphero/status/520193278419824641>

app\_faker on Thu Oct 09 says  
*#massimo4full Thanks for building and supporting your great #web2py framework. It really helped us building some next gen webapps*  
[https://www.twitter.com/app\\_faker/status/520193307066892288](https://www.twitter.com/app_faker/status/520193307066892288)

Rene Dohmen on Thu Oct 09 says  
*#massimo4full Thanks for the greatest full stack python web framework. We learned a lot from it.*  
<https://www.twitter.com/acidjunk/status/520194185572253696>

Jonathan Benn on Thu Oct 09 says  
*Massimo's work on web2py and his book helped me improve my Python and web development skills and it looks great on my resume!* #massimo4full  
<https://www.twitter.com/JonathanBenn/status/520195324087123968>

Cary Brown on Thu Oct 09 says  
*#massimo4full You have consistently shown respect, consideration, diligence, proactive thinking and a willingness to listen to students. TY!*  
[https://www.twitter.com/\\_CBrown\\_/status/520195905811542016](https://www.twitter.com/_CBrown_/status/520195905811542016)

Francois Chesnay on Thu Oct 09 says  
*#massimo4full Massimo created a fabulous web framework Web2Py, which is very useful to train both experienced and newbies in programming.*  
<https://www.twitter.com/pythonanalyst/status/520206755528048641>

Jaime Sempere on Thu Oct 09 says

#massimo4full #web2py I really love web2py and all of the work done by Massimo on google group community. Thanks for all your work and help.

<https://www.twitter.com/JaimeSempere/status/520210330203340800>

Ovidio Marinho on Thu Oct 09 says

#massimo4full With the technical support of Professor Massimo we can greatly improve our business in IT here in Brazil. Thank you Massimo.

<https://www.twitter.com/ovidiofalcao/status/520210406602579969>

Chris Baron on Thu Oct 09 says

#massimo4full He got me a gig at The Field, helped me try to publish a paper, and jump started my freelance era with a contact. A good man

<https://www.twitter.com/cbaron/status/520210432338825216>

John F Halter on Thu Oct 09 says

#massimo4full Massimo is excellent! I learned a ton in his class, and would recommend it to anyone! Massimo for full professor!

<https://www.twitter.com/JohnFHalter/status/520211617397497859>

Leonel Câmara on Thu Oct 09 says

Massimo's framework @web2py, his book, and video lectures. Helped me become much more productive. I fully support #massimo4full

[https://www.twitter.com/leonel\\_camara/status/520216069970329600](https://www.twitter.com/leonel_camara/status/520216069970329600)

margarita uk on Thu Oct 09 says

Massimo De Pierro helped my organization look for a solution to a complex issue and offered advice on how to finish a project.

#massimo4full

<https://www.twitter.com/margeuk1/status/520216780640243714>

Christina Y on Thu Oct 09 says

Massimo Di Pierro's class on web dev at DePaul was the reason I got my first job out of college. Thank you professor! #massimo4full

<https://www.twitter.com/SoManyBears/status/520216855886045185>

Allen Moy on Thu Oct 09 says

Massimo was a great professor. I had him for a professor and he was excellent at teaching fundamentals in data structures. #massimo4full

<https://www.twitter.com/muiallen168/status/520218229021474817>

André Kablu on Thu Oct 09 says

Massimo is a great teacher with worldwide influence. We're like a big world classroom! GO GO #massimo4full !!!

<https://www.twitter.com/KabluBR/status/520218561751441409>

Super Grace on Thu Oct 09 says

#massimo4full is a great professor who is always there to help the students! I really enjoy his classes too!!

<https://www.twitter.com/Supergcao/status/520218630227636224>

Jamie Sampson on Thu Oct 09 says

His simulation course taught simulation, Python and financial return calculations. Used all successfully on job interviews! #massimo4full

<https://www.twitter.com/FlyTimeTrackDad/status/520220677119307776>

Charlie on Thu Oct 09 says

#massimo4full massimo's web2py teaches and enables rapid web dev from prototypes to polished products-Ideation to realisation for anyone!

<https://www.twitter.com/CharlieBopperr/status/520220831918456832>

Iuri Guilherme S. M. on Thu Oct 09 says

#Web2Py is one of the few truly "changing world" initiatives that we have. I sincerely hope it goes on and/or evolves.

#massimo4full

<https://www.twitter.com/aindatenhoconta/status/520224177521831940>

Ariel Yang on Thu Oct 09 says

#massimo4full Massimo is an excellent professor! He cares about students even after they graduated from school. You have my full support! :D

[https://www.twitter.com/cute\\_bubbles/status/520226876254478336](https://www.twitter.com/cute_bubbles/status/520226876254478336)

Brian Lemberger on Thu Oct 09 says

*I took his web programming course, it was excellent prep for my current job. Web2Py is an amazing framework! #massimo4full*  
[https://www.twitter.com/castle\\_bravo/status/520226942428401664](https://www.twitter.com/castle_bravo/status/520226942428401664)

Peter Szczepanski on Thu Oct 09 says

*Massimo DiPierro is one of the most passionate, knowledgeable, and influential Computer Science professors I had at @DePaulU.*  
*#massimo4full*  
<https://www.twitter.com/CompSciPeter/status/520230577950060544>

David Bain on Thu Oct 09 says

*We've used the web framework web2py in teaching web programming at the University of the West Indies. #massimo4full #web2py*  
<https://www.twitter.com/pigeonflight/status/520232319899021312>

tavooca on Thu Oct 09 says

*Massimo is a great support person without any profit, thanks #massimo4full*  
<https://www.twitter.com/tavooca/status/520232320637632512>

Shane on Thu Oct 09 says

*An awesome professor, who has helped us so much with computer algorithms in real life situations! #massimo4full*  
<https://www.twitter.com/zonisx/status/520234184149708802>

Al-Ourani on Thu Oct 09 says

*#massimo4full I can never be thankful enough for all your support Prof Massimo behind classroom and even in airport, but may God bless you.*  
<https://www.twitter.com/alourani/status/520234349031993344>

Kiran Subbaraman on Thu Oct 09 says

*I like web2py & the community that supports it; thanks Massimo D Pierro. Helped me with the tech part of my start-up.*  
*#massimo4full*  
<https://www.twitter.com/kirsn/status/520235206041538562>

Jiayu Chen on Thu Oct 09 says

*#massimo4full I have taken professor's Fin662, It is very useful and his teaching style is really attractive. He is a wonderful professor!*  
<https://www.twitter.com/Jiayu1991/status/520236621291982849>

Brian Moloney on Thu Oct 09 says

*Massimo is a go-to resource for talented computer scientists. @iscape\_chicago has hired many of his students over the years.*  
*#massimo4full*  
[https://www.twitter.com/Brian\\_Moloney/status/520246755242549248](https://www.twitter.com/Brian_Moloney/status/520246755242549248)

Derek H on Thu Oct 09 says

*A fine professor. I wish I had the opportunity to have more than just one class with him. #massimo4full*  
<https://www.twitter.com/realvicda/status/520254152744509440>

Stereodose on Thu Oct 09 says

*#massimo4full Web2py and its community have been an essential part of allowing Stereodose to exist; big thanks to Massimo!*  
[https://www.twitter.com/stereo\\_dose/status/520254678119231489](https://www.twitter.com/stereo_dose/status/520254678119231489)

Ralph Loizzo on Thu Oct 09 says

*A man for the people, championing education, sharing knowledge through open source (web2py). Raise a glass for #massimo4full*  
<https://www.twitter.com/FriarTech/status/520255782202650624>

Dmitriy Khomitskiy on Thu Oct 09 says

*Great professor and mentor. The best at @DePaulU school of computer science. #massimo4full*  
<https://www.twitter.com/soularis999/status/520258193252188161>

Giorgio Zoppi on Thu Oct 09 says

*#massimo4full was useful for web2py.*  
[https://www.twitter.com/jo\\_zoppi/status/520260431516925952](https://www.twitter.com/jo_zoppi/status/520260431516925952)

Neil Dahlke on Thu Oct 09 says

*@mdipierro easily one of the best professors I had! Who else writes a web server from scratch (and memory) in one class? #massimo4full*

<https://www.twitter.com/neildahlke/status/520263838168408064>

Cássio Botaro on Thu Oct 09 says

#massimo4full thanks to make my life easier. We love web2py. #sentibol <http://t.co/920iKbP4GA>  
<https://www.twitter.com/cassibotaro/status/520263875607134208>

mschray on Thu Oct 09 says

#massimo4full Has done a great job building web2py and ensuring it runs flawlessly on #Windows. His students also think he is awesome  
<https://www.twitter.com/mschray/status/520267123256733696>

Tamas++ on Thu Oct 09 says

Web2py is an awesome framework, I have been using it to build sites and apps for more than 6 years!! Thank you @mdipierro!  
#massimo4full  
<https://www.twitter.com/TamasPlusPlus/status/520267175865491456>

Mark Brown on Thu Oct 09 says

#massimo4full Massimo taught me some great computational/modeling skills and opened my eyes to numerous strategies in the trading industry.  
<https://www.twitter.com/MarkBrownMKB/status/520267484465623041>

Nick Vargish on Thu Oct 09 says

#massimo4full Massimo Di Piero's technical skill and excellent leadership produced Web2Py, a framework that enabled many successes for me.

<https://www.twitter.com/nickvargish/status/520268170284630016>

Joe Barnhart on Thu Oct 09 says

#massimo4full Because of MDP and web2py, I started a new web biz and launched a new career. He means more to me than my MSEE Berkeley profs.

<https://www.twitter.com/liv2cod/status/520269628073713665>

forvaidya on Thu Oct 09 says

#massimo4full I am immensely benefited by web2py framework authored by MP. This one of simplest, yet powerful framework I have ever used.

<https://www.twitter.com/forvaidya/status/520273613853712384>

Bruno Cezar Rocha on Thu Oct 09 says

An experienced Developer, Inventor, Scientist and Professor. I learned a lot from him working together in web2py development

#massimo4full

<https://www.twitter.com/rochacbruno/status/520276990130024448>

Ryan L on Thu Oct 09 says

I'll be consulting my notes from Prof DiPierro's class for years to come; an energetic, creative & passionate prof. who cares

#massimo4full

<https://www.twitter.com/flyinggranolabar/status/520278752698191872>

Sean Neilan on Thu Oct 09 says

Massimo Di Pierro is the best professor ever and DePaul needs to give him tenure and stuff! #massimo4full

<https://www.twitter.com/wtfisseandoing/status/520279918392123395>

Derick on Thu Oct 09 says

Massimo was my first programming professor. He taught me my beginner course in python and was a great help! Knows his stuff!

#massimo4full

<https://www.twitter.com/BHawks4life23/status/520281643651563520>

Don O'Hara on Thu Oct 09 says

#massimo4full @massimo - Thanks for the tireless work on web2py, setting up conferences, answering questions, fixing code; all with a smile!

<https://www.twitter.com/donohara/status/520289011466653696>

Pinco Pallo on Thu Oct 09 says

After having evaluated the whole bunch of python frameworks out there, Massimo's one has proven being the best for our corp.

#massimo4full

<https://www.twitter.com/mscavazzin/status/520293568154697728>

Cartesian Creative on Thu Oct 09 says

*When you move on from 'Old-fashioned agile', Web2Py and Bluemix are natural choices. #massimo4full*

<https://www.twitter.com/Cartesive/status/520294571691892736>

Francisco Ribeiro on Thu Oct 09 says

*It's truly inspiring to follow @mdipierro brilliant contributions to FOSS and to learn from his dedication to support them all*

#massimo4full

<https://www.twitter.com/blackthorne/status/520299803536678914>

Roberto Perdomo on Thu Oct 09 says

*Great person and professional, humble and dedicated. Always willing to share their knowledge. Is a honor to meet him. #massimo4full*

<https://www.twitter.com/robertop23/status/520301208360452096>

Duncan Macneil on Thu Oct 09 says

*Even the purest Python framework forces a cut-down template language & needs third-party Ajax library. Not so with Web2Py.*

#massimo4full

[https://www.twitter.com/duncan\\_macneil/status/520301881881395200](https://www.twitter.com/duncan_macneil/status/520301881881395200)

Ramiro B. da Luz on Thu Oct 09 says

*Learned a lot with Massimo at PyConUS 2012 working on web2py sprint. Thank you for your help Massimo. #massimo4full*

<https://www.twitter.com/ramiroluz/status/520303137631272960>

Jim McNeill on Thu Oct 09 says

*#massimo4full Web2Py gave me the start I needed to finally get my head round Google App Engine - go Massimo!*

<https://www.twitter.com/TogetherJim/status/520309444333150208>

Steven Tang on Thu Oct 09 says

*#massimo4full Professor DiPierro is terrific on both personality and academic, he was the only one I took two courses from him*

@DePaul

<https://www.twitter.com/tang1013/status/520313755247198208>

Michele Comitini on Thu Oct 09 says

*collaborating with @mdipierro on web2py and other projects since 5 yrs. I keep learning from him. #massimo4full*

<https://www.twitter.com/michelecomitini/status/520315474329206784>

Karen Cardenas on Thu Oct 09 says

*I looked forward to Massimo's class every week bc he showed us solutions to coding problems in brilliantly simply ways. #massimo4full*

<https://www.twitter.com/KarenCardenas94/status/520317185437081601>

Paolo Betti on Thu Oct 09 says

*web2py is a great framework and with it I learned good python programming. #massimo4full*

[https://www.twitter.com/artifex\\_it/status/520322871508619264](https://www.twitter.com/artifex_it/status/520322871508619264)

Pystar on Thu Oct 09 says

*I created my first full web app in Python with web2py. A full stack Python web framework with impeccable documentation. #massimo4full*

<https://www.twitter.com/pystar/status/520323931899584512>

Steve Shepherd on Thu Oct 09 says

*#massimo4full He deserves to be a full professor. He has unwavering dedication to web2py and his unlimited hours of support are fantastic*

<https://www.twitter.com/sargshep/status/520325597919068160>

Alex Glaros on Thu Oct 09 says

*#massimo4full Massimo's vision has enable me to create a citizen engagement platform to promote better goverment world wide*

<https://www.twitter.com/alexgclaros/status/520329905087606784>

Shad Taylor on Thu Oct 09 says

*#massimo4full PROMOTE! PROMOTE! Passion is key and he has it. boolean prof\_knowledge = true; while(prof\_knowledge) { myJavaSkills++; };*

<https://www.twitter.com/shadtaylor/status/520341610622500865>

borocomphelp on Thu Oct 09 says

*#massimo4full Thanks Massimo, Web2py is immense! Great for learning web apps. Best of luck.*

<https://www.twitter.com/borocomphelp/status/520342125339111425>

YaBa on Thu Oct 09 says

*Hey #massimo4full Great job on web2py. Keep up the good work!! Best wishes.*

<https://www.twitter.com/yaba/status/520342351672123392>

Jerry Schnepf on Thu Oct 09 says

*#massimo4full Massimo is a truly talented educator. In his algorithms class and he explained very complex problems simply.*

<https://www.twitter.com/JerrySchnepf/status/520345403011514368>

Mark Graves, JR on Thu Oct 09 says

*Massimo's leadership inspired me to build #MyIRE on the @web2py platform. Wouldn't be here without him. #OpenSourceScience*

*#massimo4full*

<https://www.twitter.com/GravesMedical/status/520345911739052033>

Tsvi Mostovicz on Thu Oct 09 says

*#massimo4full Thanks Massimo for the development of web2py. It has allowed me to tiptoe into the world of web development. Thank you.*

<https://www.twitter.com/tmosto/status/520351752928124928>

Joe J Jasinski on Thu Oct 09 says

*I highly recommend Massimo's web2py class. Really interesting framework! #massimo4full*

<https://www.twitter.com/JoeJJasinski/status/520351942862589952>

goNubex on Thu Oct 09 says

*#massimo4full Thanks Massimo for your amazing work & dedication into web2py, best of luck.*

<https://www.twitter.com/goNubex/status/520354314728906752>

» on Thu Oct 09 says

*Massimo is a great teacher. I loved GAM 450, and Intro to Web Apps. both very challenging and informative. #massimo4full*

<https://www.twitter.com/p3llin0r3/status/520356257614790656>

Dan Bowker on Thu Oct 09 says

*#massimo4full opened my eyes to statistical computation and the possibilities within. THANKS!*

<https://www.twitter.com/danbowker14/status/520358851564953600>

Bryan Barnard on Fri Oct 10 says

*#massimo4full WebAppFramework course and commitment to web2py informed and inspired me and other students to explore open source. PROMOTE!!*

<https://www.twitter.com/nardbard/status/520383908852940800>

Satishkumar Sadasiva on Fri Oct 10 says

*#massimo4full*

<https://www.twitter.com/sksadasivan/status/520386385862070272>

Diomedes Tydeus on Fri Oct 10 says

*Massimo has been an inspiration and launched my ability to dive into frameworks and OSS; professor and leader. #massimo4full*

<https://www.twitter.com/DiomedesTydeus/status/520391267344076800>

Marco Tilio Porto on Fri Oct 10 says

*#massimo4full Thanks to Massimo and Web2py we were able to save about U\$ 500.000,00/month on medical shifts and improve health care.*

<https://www.twitter.com/mtcporto/status/520398854617698304>

Alex on Fri Oct 10 says

*Turning complex algorithms & data structures into understandable education #massimo4full*

[https://www.twitter.com/a3\\_d3/status/520402625276022784](https://www.twitter.com/a3_d3/status/520402625276022784)

Benjamin on Fri Oct 10 says

*Thank you Massimo for helping me become more innovative. I enjoyed your Numerical Methods and Monte Carlo Algo courses. #massimo4full*

[https://www.twitter.com/My\\_Quant\\_Mind/status/520402991560802304](https://www.twitter.com/My_Quant_Mind/status/520402991560802304)

stifan kristi on Fri Oct 10 says

*massimo have a clear solution in web2py groups when me & the others have a problem, thank you so much massimo  
#massimo4full*

<https://www.twitter.com/stevevanchristi/status/520413597558509568>

Diego Guerrero on Fri Oct 10 says

*#massimo4full Great professor, highly knowledgeable in psychics for game dev & programming, willing to listen to students suggestions!*

<https://www.twitter.com/dmguerrero/status/520432182066565121>

Ahmed Ibrahim on Fri Oct 10 says

*#massimo4full Massimo is the kind of professor that makes DePaul a great place to learn. It's been an honor to be a part of his classes*

<https://www.twitter.com/alfal83/status/520437649744465920>

Joseph Simpson on Fri Oct 10 says

*#massimo4full Massimo is a brilliant teacher, engineer, and pragmatic problem solver. Very few full professors perform at his level.*

<https://www.twitter.com/jjs0sbw/status/520443916081188864>

Babu Annamalai on Fri Oct 10 says

*Thanks Massimo for the wonderful web2py web framework. I discover and learn more as I use it #massimo4full*

<https://www.twitter.com/babua/status/520450382779019265>

Mariano Reingart on Fri Oct 10 says

*#massimo4full @web2py is an example of your very didactic & challenging perspective to help our critical thinking, thanks @mdpierro #python*

<https://www.twitter.com/reingart/status/520451267328352256>

Matheus Cardoso on Fri Oct 10 says

*@web2py Showed me that web dev wasn't scary. It showed that could be easy, fast, secure and fun! All thanks to @mdpierro.*

*#massimo4full*

<https://www.twitter.com/MatheusCAS/status/520451847761698817>

Denis Lozar on Fri Oct 10 says

*The entrance to the world of web development would be much more painful without @web2py. Thank you Massimo. #massimo4full*

<https://www.twitter.com/dencko/status/520468423412047872>

Vladimir Dronnikov on Fri Oct 10 says

*#massimo4full thanks massimo for web2py which opened me the door to web programming. he's very friendly & prof answering questions. be full!*

<https://www.twitter.com/dronnikov/status/520472430255419392>

Giuseppe Andronico on Fri Oct 10 says

*#massimo4full I know Massimo as a expert practicioner in Lattice Field Theory and a valuable expert in computer programming*

<https://www.twitter.com/gandros/status/520475938555842560>

felix jimenez on Fri Oct 10 says

*#massimo4full very gratefull with massimo by his contribution with web2py, the teaching, the attention with the community, thank you*

<https://www.twitter.com/felixjimenez09/status/520485366021623808>

tomasz bandura on Fri Oct 10 says

*#massimo4full*

<https://www.twitter.com/tomaszbandura/status/520486941293178880>

Antonio Ramos on Fri Oct 10 says

*#massimo4full*

<https://www.twitter.com/ramstein74/status/520492752547508224>

juanitela on Fri Oct 10 says

*#massimo4full web2py has great design and a superb manual. Massimo answers personally many requests for help and solves bugs very quickly!*

<https://www.twitter.com/juanitela/status/520493194354913280>

Rahul Dhakate on Fri Oct 10 says

#massimo4full I am using web2py in two projects one of which is a hms and other issue tracking system. I plan to start a company using them.

<https://www.twitter.com/dhakte/status/520508645100896256>

Demetrius Moorer on Fri Oct 10 says

#massimo4full Massimo is a fantastic advisor that understood my learning goals and helped me tailor my class schedule to achieve them!

<https://www.twitter.com/DMoorer86/status/520518338850865152>

Rolf Ach on Fri Oct 10 says

#massimo4full thank you @mdpierro for the wonderful web2py web framework!

<https://www.twitter.com/rolfach/status/520534709562929154>

Richard Warg on Fri Oct 10 says

#massimo4full web2py is brilliant, the tutorial & manual are first rate. The best I've seen in 50 years. (UC Davis IT retired)

<https://www.twitter.com/momus40/status/520540505923280900>

Charm on Fri Oct 10 says

#massimo4full Prof Massimo's course has helped me to do quantitative research at firm including develop dynamics prepayments model for MBS.

<https://www.twitter.com/srisuthc/status/520564803727659008>

Jordan Myers on Fri Oct 10 says

#massimo4full Prof. Di Pierro helped me understand how easy it is to automate tasks, which helped me immensely in my current internship.

<https://www.twitter.com/punInator/status/520578034705784833>

Sean Seguin on Fri Oct 10 says

He was always enthusiastic and passionate about the subject at hand, and he explained each subject in an understandable way

#massimo4full

<https://www.twitter.com/Seaguin/status/520582603796004866>

Craig Matthews on Fri Oct 10 says

#massimo4full The web2py project has facilitated my projects in web development.

<https://www.twitter.com/c1959/status/520587094377263104>

Alex.Sun on Fri Oct 10 says

#massimo4full Prof. Massimo was willing to help outside of course and was motivational. Job information he shared very helpful for my career

[https://www.twitter.com/ray\\_sun1116/status/520599112970813443](https://www.twitter.com/ray_sun1116/status/520599112970813443)

Jim Caine on Fri Oct 10 says

Massimo's Monte Carlo class is amongst the best classes I've taken - in class programming examples are awesome. #massimo4full

[https://www.twitter.com/jim\\_caine/status/520609068625326081](https://www.twitter.com/jim_caine/status/520609068625326081)

Jerry James on Fri Oct 10 says

#massimo4full Massimo is a great man that helped me understand web2py and python in general. Great teacher and friend

<https://www.twitter.com/xcobary/status/520611137126014976>

Jason Burton on Fri Oct 10 says

Always enthusiastic, Massimo excels at teaching and is inclusive of students in his projects. He is essential! #massimo4full

<https://www.twitter.com/jaslburton/status/520635567399526400>

Nicholas Anderson on Fri Oct 10 says

I graduated a year ago and I still remember Di Pierro's classes as being some of the best. #massimo4full

<https://www.twitter.com/seurimas/status/520637656196452353>

dena bell on Fri Oct 10 says

#Massimo4full has been an innovative and energetic member of our community organization. He has directed technology efforts with ease.

[https://www.twitter.com/dena\\_bees/status/520640034140348417](https://www.twitter.com/dena_bees/status/520640034140348417)

E. Gamarro on Fri Oct 10 says

#massimo4full Massimo is a critical and energetic thinker. As an instructor he challenged me and helped me break out of my comfort zone  
<https://www.twitter.com/gamarroe/status/520666076037021696>

Yuancheng Zhang on Fri Oct 10 says

#massimo4full Massimo uses his passion to bring me knowledge and hope.

[https://www.twitter.com/Mr\\_EnDaYe/status/520666293033111552](https://www.twitter.com/Mr_EnDaYe/status/520666293033111552)

Martin Senecal on Fri Oct 10 says

#massimo4full I stumbled on web2py in 2010, & today I am exactly where I want to be thanks to Massimo's patient willingness to teach & share

<https://www.twitter.com/martsenecal/status/520669305248038912>

Brian Rojas on Fri Oct 10 says

#massimo4full was a great teacher for me, learned a lot about game physics and had fun as well

<https://www.twitter.com/marchinram/status/520680867182555137>

marco on Fri Oct 10 says

#massimo4full Marco Ramos of Peru. His teachings and what inspires is part of my professional training and an example to follow.

[https://www.twitter.com/marco\\_coyla/status/520700702901813248](https://www.twitter.com/marco_coyla/status/520700702901813248)

Josh on Fri Oct 10 says

Here's to the prof who opened my eyes to the world of probability theory in algorithms and jump started my research career!

#massimo4full

<https://www.twitter.com/y3sh/status/520716413149802496>

Aubrey Louw on Sat Oct 11 says

Creating a prod database monitoring web application at work built on web2py. Couldn't do it w/out web tech class at depaul

#massimo4full

<https://www.twitter.com/alouw/status/520728752037593089>

zhiyong zhang on Sat Oct 11 says

#massimo4full gave me the most I'm still using in my work. I'm really thankful to his patience and knowledge.

<https://www.twitter.com/malagapeter/status/520744618804404224>

Alan Etkin on Sat Oct 11 says

#massimo4full You always read a Massimo's post or book knowing you're going to learn something new about computing

<https://www.twitter.com/spametki/status/520753936840404992>

dkdewitt on Sat Oct 11 says

#massimo4full sparked my interest in Python but also Web with Web2Py. I could right a paper on his influence on my learning. Thx Massimo!!

<https://www.twitter.com/dkdewitt/status/520770995963981824>

Yang Liu on Sat Oct 11 says

#massimo4full In his classes, I learned not only algorithms and programming skills, but also the creative thinking and cutting edge trends.

[https://www.twitter.com/yang\\_liu9/status/520791530794008576](https://www.twitter.com/yang_liu9/status/520791530794008576)

Ken Stox on Sat Oct 11 says

#massimo4full Massimo has demonstrated a prowess at forging young minds into precision machines. ;-&gt;

<https://www.twitter.com/IckyIckyPtang/status/520794223646216192>

Antonio Jesús Alba on Sat Oct 11 says

I met Massimo from his work in the development and optimization of the great framework web2py which I used sometimes #goodluck

#massimo4full

<https://www.twitter.com/malagaonrails/status/520819257714552833>

Andrew Willimott on Sat Oct 11 says

#massimo4full such a great contribution, to make such a capable web framework freely available, a great teacher, mentor, and "chair" of dev.

<https://www.twitter.com/WillimottAndrew/status/520823226440507394>

Kristjan Strojan on Sat Oct 11 says

*Massimo Di Pierro is one of a kind mentor and inventor. I cannot imagine my ventures without @web2py. Thanks, Massimo!!*

#massimo4full

<https://www.twitter.com/kristjanstrojan/status/520848776828030976>

Yair Eshel on Sat Oct 11 says

*I've learned so much from this guy, just by looking at improvements he made for my patches. #massimo4full*

<https://www.twitter.com/guruyaya/status/520851388348129280>

piero crisci on Sat Oct 11 says

*I want to thank #massimo4full for his great work. He makes possible to realize fast projects with web2py. I'm using it for agile developing*

<https://www.twitter.com/PieroCrisci/status/520931466663165952>

antonioxgarcia on Sat Oct 11 says

*Taught some of the most engaging courses I've taken #massimo4full*

<https://www.twitter.com/antonioxgarcia/status/520946132143190016>

JHancock on Sat Oct 11 says

*The thing that helped me is even though I was an inexperienced coder at the time he treated me like a real person. #massimo4full*

<https://www.twitter.com/JHancock/status/520971607217426432>

Matt McCabe on Sat Oct 11 says

*He is a great teacher who is passionate about what he does and helps expand the students understanding on the subject #massimo4full*

[https://www.twitter.com/McCAbeLincoln\\_/status/521002660695392256](https://www.twitter.com/McCAbeLincoln_/status/521002660695392256)

Matías Herranz on Sat Oct 11 says

*Awesome and inspirational speaker, big FLOSS supporter, great guy #massimo4full /cc @mdipierro*

<https://www.twitter.com/matiasherranz/status/521007753402146816>

yarko on Sat Oct 11 says

*Registration for PyCon'09-10; wkshop: full site &lt;1hr; WingIDE template debugging; support lively community - many fun days w/ #massimo4full*

<https://www.twitter.com/yarkot/status/521009158749827072>

Carlos Zenteno on Sat Oct 11 says

*#massimo4full Mass didn't teach me webdev @DePaul he did it @Life. Envious of students that enjoy him @school. Hope to meet him in future.*

<https://www.twitter.com/CarlosZenteno/status/521046941103169536>

Namsø on Sat Oct 11 says

*#web2py is an awesome python MVC framework that I enjoy much more than Rails #massimo4full*

<https://www.twitter.com/osmanwastaken/status/521048731706093568>

Kyle Horn on Sat Oct 11 says

*Great teacher who helped me understand the usefulness of Python! #massimo4full*

[https://www.twitter.com/Kylehorn\\_/status/521061687361683456](https://www.twitter.com/Kylehorn_/status/521061687361683456)

Russ King on Sat Oct 11 says

*#massimo4full - very deserved web2py is a fantastic achievement and Massimo is extremely helpful and capable in driving its success*

<https://www.twitter.com/newglobalstrat/status/521063666079113217>

Arnold Gowans on Sat Oct 11 says

*#massimo4full Dr. DiPierro's passion, knowledge, and involvement with the community has been instrumental in my success as a programmer*

<https://www.twitter.com/dgowanstweet/status/521064264979591168>

Emanuel Rocha Woiski on Sat Oct 11 says

*#massimo4full Thank you Massimo for your hard work and dedication to the web2py framework, a truly fine piece of software.*

<https://www.twitter.com/erwoiski/status/521065214561640448>

Luciano L. Podazza on Sun Oct 12 says

*@web2py helped me making my dreams come true. Wouldn't be possible without @mdipierro. Thanks Massimo for all the magic :) #massimo4full*

<https://www.twitter.com/keniobats/status/521088106045669376>

Richard Whittemore on Sun Oct 12 says

*As both a professor and academic advisor, professor Massimo has served me well and he would make a great full time professor. #massimo4full*

<https://www.twitter.com/r2whitt/status/521147702512336897>

Tito Garrido Ogando on Sun Oct 12 says

*It's incredible how Massimo is able to manage an awesome framework and an incredible community! It is an honor be part of it! #massimo4full*

<https://www.twitter.com/titogarrido/status/521307445956993024>

K.Moutselos on Sun Oct 12 says

*#massimo4full Massimo's Web2py is the best python web framework! His continuing presence and support to web2py coding community is inspiring*

<https://www.twitter.com/kmouts/status/521335727288631296>

Albert Abril on Sun Oct 12 says

*Massimo (@mdipierro) has been one of my most influencers on programming. I've learnt a lot thanks to his code and knowledge #massimo4full*

<https://www.twitter.com/desmondo/status/521378268138983425>

Carter Maslan on Sun Oct 12 says

*#massimo4full - Massimo is a talent multiplier that helps me his other teammates grow. A mentor via doing!*

<https://www.twitter.com/CarterMaslan/status/521398613826891776>

Suits and Tables on Sun Oct 12 says

*Professors that invest in their students development over their own accolades should be rewarded. DePaul is lucky! #massimo4full*

<https://www.twitter.com/SuitsandTables/status/521412658109485056>

Matt on Sun Oct 12 says

*#massimo4full Quick responses to questions, insightful and practical style, with up to date, real world experience. He's an asset to DePaul.*

<https://www.twitter.com/beaulingpin/status/521418851842416640>

Michael Beskin on Sun Oct 12 says

*#massimo4full Massimo's lectures helped me gain a grasp on important data structures and web development - with professional mindfulness.*

<https://www.twitter.com/mmbeskin/status/521444809848811520>

Peter Xie on Mon Oct 13 says

*#massimo4full With Prof. Massimo's suggestions, I changed my major from Finance to Computational Finance. He is a really nice Professor.*

<https://www.twitter.com/GuodongXie/status/521469082311200768>

c b l on Mon Oct 13 says

*#massimo4full Massimo demo'd his work for DePaul CSS and I was stunned! I didn't until that moment realize how lucky I was to study here!*

<https://www.twitter.com/depaulalum/status/521470285434720257>

Michael Howden on Mon Oct 13 says

*Massimo has always supported @SahanaFOSS #Disaster Management Software which is developed using @web2py #Massimo4Full*

<https://www.twitter.com/michaelhowden/status/521475204199886849>

Radzi on Mon Oct 13 says

*#massimo4full best of wishes and luck. Thanks for all your contributions in the web development arena.*

<https://www.twitter.com/imradzi/status/521575504281739268>

Teddy Nyambe on Mon Oct 13 says

*Introduced to #Python coding through #web2py and developing internal web apps. The dev to implementation cycled has reduced #massimo4full*  
<https://www.twitter.com/teddynyambe/status/521592949097136128>

Bob W on Mon Oct 13 says

*Who is always responsive and goes the extra mile to make the end result even better than requested? Massimo! #massimo4full*  
[https://www.twitter.com/d\\_conblues/status/521686773378736128](https://www.twitter.com/d_conblues/status/521686773378736128)

Alex Krautmann on Mon Oct 13 says

*Not often is a professor creating buzz on reddit or h-news. Also a great prof. I would tell anyone to take a class with him #massimo4full*  
<https://www.twitter.com/AlexKrautmann/status/521688721863999489>

Dr. Erin Mason on Mon Oct 13 says

*#massimo4full Massimo has connected me to other like-minded faculty and supported my own efforts in integrating technology in courses.*  
<https://www.twitter.com/ecmmason/status/521711346140717057>

William Zhang on Mon Oct 13 says

*#massimo4full thanks for your helpful classes*  
<https://www.twitter.com/WilliamZhang10/status/521757187710808064>

Steve Chou on Mon Oct 13 says

*He is a great teacher, not only has an attracting way in teaching, but also refers a lot open sources for career. #massimo4full*  
<https://www.twitter.com/SteveChou1212/status/521783664640679936>

# Part III

# Papers





ELSEVIER

7 November 1996

PHYSICS LETTERS B

Physics Letters B 388 (1996) 90–96

# Mass, confinement and CP invariance in the Seiberg-Witten model

Massimo Di Pierro<sup>a</sup>, Kenichi Konishi<sup>b,l</sup>

<sup>a</sup> Dipartimento di Fisica – Università di Pisa, Istituto Nazionale di Fisica Nucleare – sez. di Pisa, Piazza Torricelli, 2 – 56100 Pisa, Italy<sup>2</sup>

<sup>b</sup> Dipartimento di Fisica – Università di Genova, Istituto Nazionale di Fisica Nucleare – sez. di Genova,  
Via Dodecaneso, 33 – 16146 Genova, Italy

Received 30 May 1996

Editor: L. Alvarez-Gaumé

## Abstract

Several physics aspects of the Seiberg-Witten solution of  $N = 2$  supersymmetric Yang-Mills theory with  $SU(2)$  gauge group, supplemented with a small mass term for the “matter” fields which leads to an  $N = 1$  theory with confinement, are discussed. We find in particular that in the massive (confining) theory the low energy physics has an exact CP symmetry, while in a generic vacuum in the massless theory CP invariance is broken spontaneously.

In a celebrated work [1] Seiberg and Witten exploited the (generalized) electromagnetic duality,  $N = 2$  supersymmetry and holomorphic property of effective actions, to solve exactly a strongly interacting non Abelian theory in four dimensions, i.e., to compute the vacuum degeneracies, and in each vacuum, to determine the exact spectrum and interactions among light particles.

An especially interesting observation of Ref [1], made in the pure  $N = 2$  supersymmetric Yang-Mills theory with  $SU(2)$  gauge group, is that upon turning on the mass term<sup>3</sup>  $\int d^2\theta \text{Tr } m\Phi^2$ ,  $m \ll \Lambda$ , the light magnetic monopole field condenses, providing thus the first explicit realization of the confinement mechanism envisaged by 't Hooft. [2] In this note we wish to make some further observations on this model.

There are in fact several related questions which to our knowledge have not yet been discussed fully. Why do the vacua in the massive (confining) theory correspond precisely to those points of the quantum moduli space (QMS) of the  $N = 2$  theory where the magnetic monopole becomes massless? Usually, one expects that a dynamically generated mass in a non Abelian theory with scale  $\Lambda$  is of the order of  $\Lambda$ , while in this model the mass gap is of the order of  $m^{1/2}\Lambda^{1/2}/\log^{1/2}(\Lambda/m)$  (see below). Why is that so? Is the CP invariance spontaneously broken at low energies? Does the oblique confinement [2] take place? Do dyons condense? What is the relation between the Seiberg-Witten effective action and the more speculative (but supposedly exact) effective superpotential constructed for the  $N = 1$  theory with the “integrating in” procedure? [3] Finally, has all this got anything to do with what happens in an  $N = 0$  theory of interest such as QCD?

A first general remark is that one is here dealing

<sup>1</sup> E-mail: KONISHI@GE.INFN.IT.

<sup>2</sup> E-mail: df144023@ipifidpt.difi.unipi.it.

<sup>3</sup> We follow the notation of Ref [1].

with a system with large vacuum degeneracy, intact after full quantum corrections are taken into account. This vacuum degeneracy is (almost) eliminated by the mass perturbation,  $\int d^2\theta \text{Tr } m\Phi^2$ , leaving only a double degeneracy corresponding to the order parameter  $u = \langle \text{Tr } \phi^2 \rangle = \pm \Lambda^2$ .

As is usual in the standard degenerate perturbation theory in quantum mechanics, to lowest order the only effect of the perturbation is to fix the vacuum to the “right”, but unperturbed, one, with  $N = 2$  (hence  $SU_R(2)$ ) symmetric properties. This means that the  $SU_R(2)$  current of the original theory,

$$J_\mu^a = \text{Tr } \bar{\Psi} \bar{\sigma}_\mu \tau^a \Psi \quad (1)$$

where  $\Psi = (\begin{smallmatrix} \lambda \\ \psi \end{smallmatrix})$ , is well approximated in the low energy effective theory by the Noether currents (of the low energy theory)<sup>4</sup>

$$J_\mu^a = \bar{\Psi}_D \bar{\sigma}_\mu \tau^a \Psi_D + i(D_\mu S^\dagger) \tau^a S - iS^\dagger \tau^a D_\mu S, \quad (2)$$

where  $\Psi_D = (\begin{smallmatrix} \lambda_D \\ \psi_D \end{smallmatrix})$  is the dual of the (color diagonal part of)  $\Psi$ , and  $S \equiv (\begin{smallmatrix} M \\ M^\dagger \end{smallmatrix})$ . In arriving at Eq. (2) we identified the high energy and low energy  $SU_R(2)$  groups such that the doublet  $(\begin{smallmatrix} \lambda_D \\ \psi_D \end{smallmatrix})$  transforms as  $(\begin{smallmatrix} \lambda \\ \psi \end{smallmatrix})$ .

Let us consider the current-current correlation function

$$\begin{aligned} i \int d^4x e^{-iqx} \langle 0 | T\{ J_\mu^3(x) J_\nu^3(0) \} | 0 \rangle \\ = (q_\mu q_\nu - q^2 g_{\mu\nu}) \Pi(q^2), \end{aligned} \quad (3)$$

and an associated  $R$ -like ratio,

$$R = \text{Disc}_{q^2} \Pi(q^2), \quad (4)$$

appropriately normalized (by introducing hypothetic “leptons”). At high energies  $R$  just counts the number of  $\lambda$  and  $\psi$  “quarks” (asymptotic freedom), apart from calculable logarithmic corrections as well as an infinite number of not-so-easily-calculable power corrections (involving gluonic and higher condensates):

$$R_{q^2 \gg \Lambda^2} \simeq 6 + O(\alpha(q^2), q^4/\Lambda^4). \quad (5)$$

<sup>4</sup>The effective low energy action is [1]  $\frac{1}{4\pi} \text{Im} [\int d^4\theta (\partial F(A_D)/\partial A_D) \bar{A}_D + \int d^2\theta (\partial^2 F(A_D)/\partial A_D^2) W_D W_D/2]$ , with the addition of the standard terms for the magnetic monopole sector,  $M^\dagger e^{V_D} M + \tilde{M}^\dagger e^{-V_D} \tilde{M}|_D + \sqrt{2} A_D \tilde{M} M|_F + \text{h.c.}$  The mass term is given by  $\int d^2\theta m \Phi^2 + \text{h.c.} = \int d^2\theta m U(A_D) + \text{h.c.}$

At low energies, the same quantity is simply given by the weakly interacting dual fields and magnetic monopoles,

$$R_{m\Lambda \ll q^2 \ll \Lambda^2} \simeq 3, \quad (6)$$

where two out of three comes from the contributions from  $\lambda_D$  and  $\psi_D$  (one each) and one from  $M$  and  $\tilde{M}$  (one half each). This amounts to an exact resummation of an infinite number of power and logarithmic corrections.

To next order the effect of the explicit  $SU_R(2)$  breaking,

$$\partial^\mu J_\mu^- = \partial^\mu \text{Tr } \bar{\psi} \bar{\sigma}_\mu \lambda = i m \text{Tr } \lambda \psi; \quad (7)$$

$$\partial^\mu J_\mu^3 = \frac{1}{2} \partial^\mu \text{Tr } (\bar{\lambda} \bar{\sigma}_\mu \lambda - \bar{\psi} \bar{\sigma}_\mu \psi) = \frac{i}{2} (\text{Tr } m \psi^2 - \text{h.c.}) \quad (8)$$

must be taken into account.<sup>5</sup> In particular, the anomaly of Ref [4]

$$m u = m \langle \text{Tr } \phi^2 \rangle = 2 \frac{g^2}{32\pi} \langle \text{Tr } \lambda(x) \lambda(x) \rangle, \quad (9)$$

hence

$$(g^2/32\pi) \langle \text{Tr } \lambda(x) \lambda(x) \rangle = \pm m \Lambda^2/2, \quad (10)$$

as well as the nonvanishing monopole condensation (Eq. (12) below) show that the ground state is no longer  $SU_R(2)$  symmetric. Also, the supersymmetric Ward-Takahashi like identity

$$\langle \text{Tr } \psi^2 \rangle = 2 m^* \langle \text{Tr } \phi^* \phi \rangle, \quad (11)$$

and the fact that  $\phi$  is in the adjoint representation hence probably  $\langle \text{Tr } \phi^* \phi \rangle \sim \text{const.} \langle \text{Tr } \phi^2 \rangle \simeq \Lambda^2$ , suggests that  $\langle \text{Tr } \psi^2 \rangle \sim O(m\Lambda^2)$  also.

The strength “ $F_\pi$ ” with which light particles are produced directly from the vacuum by the  $SU_R(2)$  current operators, can be easily read off from the expression of the low energy currents Eq. (2). Let us recall that upon turning on the mass term the vacuum is found to be

<sup>5</sup>Although we are discussing here the first order mass corrections in a degenerate perturbation theory, many crucial relations are exact due to the nonrenormalization theorem (the form of the superpotential, etc). Most results below, in particular the exact CP invariance of the low energy theory, should survive higher order corrections which affect only the D type terms.

fixed at  $A_D = 0$ ,  $u = \Lambda^2$  as noted by Seiberg and Witten. [1] The superpotential  $\sqrt{2}A_D\tilde{M}M + mU(A_D)$ , minimized with respect to  $A_D$ ,  $\tilde{M}$  and  $M$ , indeed yields  $A_D = 0$  and the magnetic monopole condensation,

$$\langle M \rangle = \langle \tilde{M} \rangle = (-mU'(0)/\sqrt{2})^{1/2}. \quad (12)$$

Expanding the magnetic monopole fields around its vacuum expectation value

$$M = \langle M \rangle + M', \quad \tilde{M} = \langle \tilde{M} \rangle + \tilde{M}', \quad (13)$$

one finds that

$$F_\pi^- \sim F_\pi^3 \sim \langle M \rangle = O(m^{1/2}\Lambda^{1/2}). \quad (14)$$

Also, it is easy to see that the three light bosons produced directly by the currents are (to lowest order) the real and imaginary parts of  $M - \tilde{M}$  and the imaginary part of  $\langle M^\dagger \rangle(M + \tilde{M})$ , apart from normalization. (Actually, a linear combination of the real and imaginary parts of  $M - \tilde{M}$  becomes the longitudinal part of the dual vector boson by the Higgs mechanism.)

As for the masses of the light particles, they can be studied most easily from the fermion bilinear terms arising from the Yukawa interaction terms upon shifting the magnetic monopole fields (Eq. (13)). It reads

$$\begin{aligned} L_Y = & \sqrt{2}[-\langle M^\dagger \rangle \lambda_D(\psi_M - \psi_{\tilde{M}}) \\ & + \langle M \rangle \psi_D(\psi_{\tilde{M}} + \psi_M)] + \frac{m}{2}U''(0)\psi_D\psi_D \\ & + \sqrt{2}[-M'^\dagger \lambda_D\psi_M + \tilde{M}'^\dagger \lambda_D\psi_{\tilde{M}} \\ & - (A_D\psi_M\psi_{\tilde{M}} + \psi_D\psi_{\tilde{M}}M' + \psi_D\psi_M\tilde{M}')] \\ & + \frac{m}{2}(U''(A_D) - U''(0))\psi_D\psi_D + \text{h.c.} \end{aligned} \quad (15)$$

A subtle point in reading off the masses from Eq. (15) is that since the fields  $\lambda_D$  and  $\psi_D$  do not have the canonical kinematic terms, they must be re-normalized by  $\lambda_D \rightarrow g_D \lambda_D$ ;  $\psi_D \rightarrow g_D \psi_D$ . (In the formula (2) for the low energy currents such a rescaling has already been done.) In doing so the argument of the dual coupling constant  $g_D$  should be taken as  $(m/\Lambda)^{1/2}$  and not  $a_D = 0$ , since in the massive theory there is an infrared cutoff. (Such a replacement should automatically take place if the perturbation in  $m$  is pushed to higher order. An analogous phenomenon is known in the old chiral perturbation theory due to the small pion mass.) This explains the  $\log(\Lambda/m)$  dependence of the masses below.

From Eq. (15) one sees that the fields  $\lambda_D$  and  $\psi_2 \equiv (\psi_M - \psi_{\tilde{M}})/\sqrt{2}$  form a Dirac type massive fermion with mass

$$\begin{aligned} m_1 &= |2g_D\langle M^\dagger \rangle| = 2|g_D \cdot (\sqrt{2}im\Lambda)^{1/2}| \\ &= \frac{4 \cdot 2^{3/4} \pi |m\Lambda|^{1/2}}{\log^{1/2} |\Lambda/m|}, \end{aligned} \quad (16)$$

while in the subspace of two Weyl fermions  $\psi_D$  and  $\psi_1 \equiv (\psi_M + \psi_{\tilde{M}})/\sqrt{2}$  the mass matrix reads

$$\begin{pmatrix} mg_D^2 U''(0)/2 & g_D\langle M \rangle \\ g_D\langle M \rangle & 0 \end{pmatrix}, \quad (17)$$

where  $U''(0) = -1/2$ . The phases of these matrix elements can be chosen all real and positive by an appropriate phase rotation of the  $\psi_D$  and  $\psi_1$  fields (more about these phases below). A real symmetric matrix can be diagonalized by a real orthogonal matrix, leading to mass eigenvalues

$$m_{2,3} \simeq \gamma |m\Lambda|^{1/2} \pm \frac{1}{2}\delta|m|, \quad (18)$$

where

$$\delta = \frac{g_D^2}{4}; \quad \gamma = 2^{1/4}g_D; \quad g_D = \frac{2\sqrt{2}\pi}{\log^{1/2} |\Lambda/m|}. \quad (19)$$

In spite of the fact that the light particles have mass,  $m_\pi \sim O(m^{1/2}\Lambda^{1/2}/\log^{1/2}(\Lambda/m)) \ll \Lambda$ , and  $F_\pi \neq 0$ , these bosons cannot be interpreted quite as pseudo Goldstone bosons.<sup>6</sup> If one attempts to write the Dashen like formula [5] by saturating the current-current correlation functions such as Eq. (3) (at  $q^2 \leq m\Lambda$ ) by the lowest lying particles, one finds that the pole term does not actually dominates. The usual pole enhancement as compared to the continuum contribution (by a power of  $\Lambda/m$ ) is here compensated by the suppression  $F_\pi^2 = O(m)$ , hence the pole term and continuum contribute both as  $O(m^2)$ , and are of the same order as the right hand side  $m\langle \text{Tr } \psi^2 \rangle + \text{h.c.}$  which is also one power of  $m$  down as compared the standard Dashen's formula, due to the supersymmetric Ward-Takahashi like relation Eq. (11). Also, from Eq. (15) one sees that the particles ( $A_D, W_D$ ) become massive only through mixing with the magnetic

<sup>6</sup>We thank M. Testa for discussions on this issue.

monopoles. Since  $N = 1$  theory is expected to confine, with all particles massive, the two  $N = 1$  vacua could not be anywhere else in the QMS of  $N = 2$  theory, except at the two singular points  $u = \pm\Lambda^2$ .

Exactly the same physics arises if one starts with an effective action with the  $(\pm 1, \pm 1)$  “dyons”,  $\tilde{N}, N$ , coupled to  $A_D + A$  fields: one finds that the introduction of a small mass term fixes the vacuum to be at  $A_D + A = 0$  ( $u = e^{+i\pi}\Lambda^2$ ,  $\theta_{\text{eff}} = -2\pi$ ). Because of the Witten’s effect ( $q$  is the observed electric charge,  $n_e, n_m$  are the integer quantum numbers labelling the particles)

$$q = n_e + (\theta_{\text{eff}}/2\pi)n_m, \quad (20)$$

the  $(1, 1)$  particle (which would be massive at  $u = +\Lambda^2$  but massless at  $u = e^{+i\pi}\Lambda^2$ ) actually becomes a pure magnetic monopole there and condenses, leading to confinement of the color electric charges. After a redefinition of the low energy fields by  $A_D + A \rightarrow A_D$  (which is an allowed  $SL(2, Z)$  transformation) and a relabelling of the quantum number  $n_e \rightarrow n_e - n_m$ , the theory in this vacuum becomes even formally identical to the case  $u = \Lambda^2$  studied above. Thus the “dyon condensation” does not occur in this model, contrary to such a statement sometimes made in the literature. We may therefore limit ourselves here and below to the  $u = \Lambda^2$  vacuum without losing generality.

In view of the light spectrum found here let us compare the massive Seiberg-Witten effective low energy theory to a more naïve guess for the structure of the effective action, constructed by introducing just two chiral superfields  $U = \text{Tr } \Phi^2$  and  $S \equiv \frac{g^2}{32\pi} \text{Tr } WW$  and by using the so-called “integrating-in” procedure, which leads to the superpotential [3]

$$S \log \frac{U^2}{\Lambda^4} + mU. \quad (21)$$

Eq. (21) is such that the anomalous transformation of the action,  $4\alpha(g^2/32\pi)F_{\mu\nu}\tilde{F}^{\mu\nu}$  (under  $\Phi \rightarrow e^{i\alpha}\Phi$ ), is formally reproduced. Such an “effective action” yields upon minimization of the scalar potential, also the correct results,  $u = \langle \text{Tr } \Phi^2 \rangle = \pm\Lambda^2$  and the anomaly Eq. (9). Nonetheless, it is an incorrect low energy action in this theory, as it does not contain light particles with mass  $\sim m^{1/2}\Lambda^{1/2}$ . The reason for such a failure seems to lie in the fact that the global  $SU_R(2)$  symmetry (and its small induced breaking)

in the limit of  $m/\Lambda \rightarrow 0$  has not been taken into account properly.

It is of particular interest to study the way CP symmetry is realized in the massless ( $N = 2$ ) and in the massive ( $N = 1$ ) theories, and the relation thereof. Although one expects no dependence on the bare  $\theta$  parameter even in the presence of the  $\Phi$  mass term since in the original theory there is a massless charged fermion ( $\lambda$ ),<sup>7</sup> CP symmetry is realized at low energies in the Coulomb phase (massless case) and in the confining phase (massive case) in different ways. In the massless theory, the  $\theta$  independence is assured by the property of the exact Seiberg-Witten solution,

$$\begin{aligned} a &= \frac{\sqrt{2}\Lambda}{\pi} \int_{-1}^1 dx \frac{\sqrt{x-u/\Lambda^2}}{\sqrt{x^2-1}}, \\ a_D &= \frac{\sqrt{2}\Lambda}{\pi} \int_1^{u/\Lambda^2} dx \frac{\sqrt{x-u/\Lambda^2}}{\sqrt{x^2-1}}, \end{aligned} \quad (22)$$

in the following sense. A shift of  $\theta$  by  $\Delta\theta$  causes the change in  $\Lambda$  as

$$\Lambda \rightarrow e^{i\Delta\theta/4}\Lambda. \quad (23)$$

$\Lambda$  depends on  $\theta$  this way since it depends on the bare coupling constant and bare  $\theta$  parameter through the complex “coupling constant”  $\tau = \theta/2\pi + 4\pi i/g^2$ . [6] But if we now move to a different vacuum by

$$u \rightarrow e^{i\Delta\theta/2}u \quad (24)$$

the net change is

$$a \rightarrow e^{i\Delta\theta/4}a; \quad a_D \rightarrow e^{i\Delta\theta/4}a_D : \quad (25)$$

a common phase rotation of  $a$  and  $a_D$ .

Thus all physical properties of an appropriately shifted vacuum  $u$  with a new value of  $\theta$  are the same as in the original theory. In particular, the full spectrum [1],

$$M = \sqrt{2}|n_m a_D + n_e a|. \quad (26)$$

as well as the low energy coupling constant and vacuum parameter are seen to be unchanged, since

<sup>7</sup> We thank S. Hsu for pointing out an error in this regards which appeared in the original manuscript.

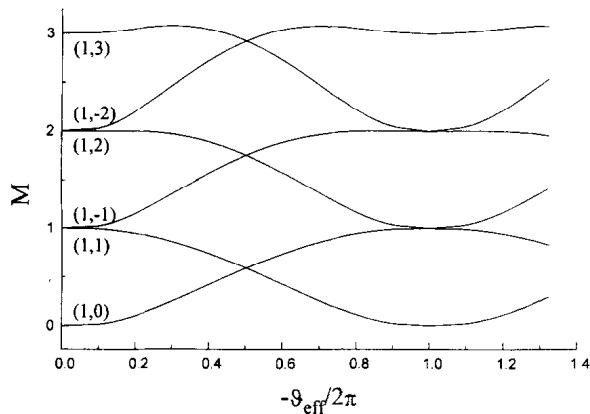


Fig. 1. Mass spectrum of some stable particles with magnetic charge in the  $N = 2$  theory as  $u$  varies as  $u = e^{i\alpha}\Lambda^2$ ,  $\alpha = 0 \rightarrow \pi$ . The numbers near each curve indicate  $(n_m, n_e)$ . The unit of mass is  $4|\Lambda|/\pi$ .

$$\tau_{\text{eff}} = \frac{\theta_{\text{eff}}}{2\pi} + \frac{4\pi i}{g_{\text{eff}}^2} = \frac{da_D}{da}. \quad (27)$$

In other words, the ensemble of theories represented by the points of QMS, taken together, is invariant under  $\theta \rightarrow \theta + \Delta\theta$ .

(The above argument may be inverted: the invariance of the theory under Eq. (23) and Eq. (24) can be interpreted as an indication that the anomalous chiral  $U(1)$  transformation property of the theory is indeed correctly incorporated in the low energy action.)

On the other hand, at fixed generic  $u$ , physics depends on  $\theta$  (as well as on the bare coupling constant  $g$ ) non trivially: this is so because the CP invariance is spontaneously broken by  $u = \langle \text{Tr } \Phi^2 \rangle \neq 0$ . Equivalently, at fixed given bare parameters there are theories with different value of  $u$  and with inequivalent physics. For instance, the spectrum of some stable particles depends on  $u$  (hence on the effective low energy coupling constant  $g_{\text{eff}}$  and the theta parameter  $\theta_{\text{eff}}$ ), if  $u$  is smoothly changed along a semicircular path  $u = e^{i\alpha}\Lambda^2$ ,  $\alpha = 0 \rightarrow \pi$ , as is illustrated in Fig. 1.<sup>8</sup> Note in particular the periodicity of the spectrum in  $\theta_{\text{eff}}$  with periodicity  $2\pi$ , in spite of a nontrivial spectral flow. Also, the theories at  $u = \pm\Lambda^2$  are charac-

terized by the fact that  $\theta_{\text{eff}} = 0$  (or  $\pm 2\pi$  which is the same as 0 because of the periodicity).

Consider now the massive theory. Upon turning on the mass term (even infinitesimal) the vacuum is fixed at  $A_D = 0$ ,  $u = \Lambda^2$  in which the effective low energy vacuum parameter  $\theta_{\text{eff}}$  takes the value zero. Nonetheless, since  $u \neq 0$  (as in other vacua in the Coulomb phase) and also  $\langle M \rangle \neq 0$ , one might wonder whether CP invariance is broken spontaneously at low energies. In fact this does not occur.

To see that the low energy theory has indeed an exact CP invariance, let us go back to the Yukawa Lagrangian Eq. (15). By using Eq. (12) and the fact that  $U'(0) = -2i\Lambda$ ;  $\Lambda = e^{i\theta/4}|\Lambda|$ , and  $U''(0) = -1/2$  one sees that the only nontrivial phases appear in  $m$  and  $\langle M \rangle$ ,

$$m = e^{i\alpha}|m|, \quad \langle M \rangle = e^{i\beta}|\langle M \rangle|; \quad \alpha \equiv \arg m, \\ \beta \equiv \pi/4 + \arg m/2 + \theta/8, \quad (28)$$

as well as through the expansion of  $U''(A_D) - U''(0)$  in powers of  $A_D/\Lambda$ . From the exact Seiberg-Witten formula for  $a_D(u)$  (Eq. (22)) one finds by a change of the integration variable that

$$\frac{a_D}{\Lambda} = \frac{i}{\pi} \sum_{n=0}^{\infty} \frac{(-)^n \Gamma^2(n+1/2) (u/\Lambda^2 - 1)^{n+1}}{2^{n+1} (n+1)! n!} \quad (29)$$

with an overall factor  $i$  on the right hand side. By inverting the series one gets

$$U(A_D) = \Lambda^2(1 + f(A_D/i\Lambda)); \\ U''(A_D) = -f''(A_D/i\Lambda) \quad (30)$$

where  $f(x) = 2x + (1/4)x^2 - (1/32)x^3 + \dots$  and  $f''(x) = 1/2 - (3/16)x + \dots$  are real functions of (possibly complex) variable  $x$ .

Now first make the phase rotations

$$\psi_D \rightarrow e^{-i\alpha/2}\psi_D; \quad \psi_M \rightarrow e^{-i(\beta-\alpha/2)}\psi_M; \\ \lambda_D \rightarrow e^{i(2\beta-\alpha/2)}\lambda_D; \quad M' \rightarrow e^{i\beta}M'. \quad (31)$$

( $\psi_{\tilde{M}}$  and  $\tilde{M}'$  transforms respectively as  $\psi_M$  and  $M'$ ). These transformations eliminate phases from all mass terms as well as from the Yukawa terms involving  $M'$ 's. On the other hand the Yukawa term  $A_D\psi_M\psi_{\tilde{M}}$  acquires a phase factor  $\exp -2i(\beta - \alpha/2) = \exp(-i\pi/2 - i\theta/4)$ . The final rotation

<sup>8</sup> It is important that such a path is taken outside the curve on which the ratio  $a_D/a$  is real and on which the spectrum changes discontinuously, some stable particle becoming unstable, etc. According to Ref [7] it is a near ellipse included entirely in the unit circle  $|u| = |\Lambda^2|$  except at the points  $u = \pm\Lambda^2$ .

$$A_D \rightarrow e^{i\pi/2+i\theta/4} A_D \quad (32)$$

however eliminates this phase from the Yukawa term and simultaneously transforms the argument of  $f''(x)$  as  $A_D/i\Lambda \rightarrow A_D/|\Lambda|$ , making the expansion coefficients of  $U''(A_D) - U''(0)$  in  $A_D$  all real.

Thus the low energy effective theory is indeed independent of the bare  $\theta$  parameter. More important, the spontaneous breakdown of CP invariance à la T.D. Lee [8] does not occur, all masses and Yukawa interaction coefficients being real. The low energy vacuum parameter  $\theta_{\text{eff}}$  is zero. This completes the proof of CP invariance in the low energy theory. Since the low energy physics does not depend on the phase  $\theta$ , no oblique confinement speculated by t'Hooft (condensation of (1,0)–(1,1) dyon pairs with opposite electric charges) [2] takes place in this model.

One might be tempted to conclude that, by using a similar argument as above, “spontaneous CP violation” does not occur a fortiori in the  $m = 0$  theory either, since  $L_Y$  is much simpler in this case (no explicit mass term, no magnetic monopole condensation). This is not so. First of all, at a generic vacuum  $u \neq \pm\Lambda^2$  there is nonzero  $\theta_{\text{eff}} F_{\mu\nu} \tilde{F}^{\mu\nu}$  term which breaks CP (this is also a spontaneous breaking since the nonzero  $\theta_{\text{eff}}$  is due to the vacuum expectation value of  $\text{Re } dA_D/dA$ ). It is true that one can transform away  $\theta_{\text{eff}}$  by an  $SL(2, R)$  transformation of the scalar  $A_D \rightarrow A_D + (\theta_{\text{eff}}/2\pi) A; A \rightarrow A$ , which leaves the rest of the effective Lagrangian invariant. Such a transformation however introduces the Yukawa term of the form

$$\sqrt{2}\{A_D + (\theta_{\text{eff}}/2\pi) A\} \psi_M \psi_{\bar{M}}. \quad (33)$$

Since the condensates  $\langle A_D \rangle$  and  $\langle A \rangle$  are in general relatively complex, no phase rotation can eliminate the phases completely from the Lagrangian: CP invariance is broken spontaneously in this case. It is interesting that the above shift of the dual scalar  $A_D$  transforms Witten's boundary effect Eq. (20) - the electric charge of the magnetic monopole,  $\theta_{\text{eff}}/2\pi$ , - into the standard (albeit mutually non-local) minimal couplings of  $M$  with  $A_{D\mu}$  and  $A_\mu$ , as can be seen from the  $N = 2$  supersymmetric completion of the Yukawa interaction, Eq. (33).

One thus reaches an amusing conclusion that the massless theory depends (in a given vacuum) on the  $\theta$  parameter, while the massive theory is independent of it!

The low energy CP invariance and non-renormalization of the  $\theta$  parameter in the massive case may be closely connected to the phenomenon of confinement. According to 't Hooft [2], the confinement is a sort of dual superconductivity, due to the condensation of (color) magnetic charges. Now if the dynamics of magnetic condensation were such that the magnetic monopoles must have rigorously zero electric charge to be able to condense, then it would follow that by Witten's formula Eq. (20) the *low energy*  $\theta$  parameter would have to be exactly zero. (For somewhat related ideas see Ref. [9].) Apparently this seems to be what happens in the massive Seiberg-Witten model. However, since the independence of the low energy theory on the bare  $\theta$  parameter is due to the presence of a massless fermion in the original model, an aspect probably not shared by the ordinary QCD, it is not clear whether the massive Seiberg-Witten model can be regarded as a good model of solution of the strong CP puzzle.

Let us conclude with a general comment. The bare  $\theta$  parameter (which can be set to zero) is renormalized in the infrared by multi-instanton effects (or loops of dyons in the dual variables) differently in various vacua, i.e., to a nonzero value in a generic vacuum of the  $N = 2$  theory, to zero in the confining phase ( $N = 1$  theory). Precisely these instanton effects are responsible for maintaining at any scale the duality relation  $\tau_D = -1/\tau$ ,  $\tau = \theta_{\text{eff}}/2\pi + 4\pi i/g_{\text{eff}}^2$ , which generalizes the Dirac's quantization condition [10]  $ge = 2\pi n$ ,  $n = 0, 1, 2, \dots$ . Note how an old puzzle related to the Dirac's quantization condition (*how to maintain the quantization condition for g and e which are both U(1) coupling constants hence which get renormalized smoothly in the same direction as the scale is slowly varied?*) [11] is solved in the Seiberg-Witten model. The “electric” coupling constant  $g_{\text{eff}}$  here is truly a non Abelian charge and gets renormalized in the opposite way compared to the  $U_D(1)$  magnetic charge  $g_D$ . This consideration seems to strengthen the idea that magnetic monopoles and dyons can appear in Nature only as composite, solitonic particles in the context of a non Abelian gauge theory, spontaneously broken (or gauge-projected) to a group involving  $U(1)$  subgroups.

One of the authors (K.K.) thanks LPTHE, Centre d'Orsay, Université de Paris-Sud where his study on

these problems started, for a warm hospitality, the organizer and participants of the Kyoto workshop on Supersymmetry (March 96) for an occasion to present and discuss some of the subjects treated here, and his colleagues A. Di Giacomo, T. Eguchi, Riccardo Guida, T. Kugo, G. Paffuti, P. Rossi, Hiroshi Suzuki, M. Testa, E. Tomboulis and Sung-Kil Yang for discussions.

## References

- [1] N. Seiberg and E. Witten, Nucl. Phys. B 426 (1994) 19.
- [2] G. 't Hooft, Nucl. Phys. B 190 [FS3] (1981) 455.
- [3] N. Seiberg and K. Intriligator, Nucl. Phys. B 431 (1994) 551.
- [4] K. Konishi, Phys. Lett. B 135 (1984) 439.
- [5] R.F. Dashen, Phys. Rev. 183 (1969) 1245; Phys. Rev. D 3 (1971) 1879.
- [6] M.A. Shifman and A.I. Vainshtein, Nucl. Phys. B 359 (1991) 571.
- [7] U. Lindström and M. Rocek, Phys. Lett. B 355 (1995) 492; A. Fayyazuddin, Mod. Phys. Lett. A 10 (1995) 2703; P. Argyres, A. Faraggi and A. Shapere, hep-th/9505190; M. Matone, hep-th/9506181; F. Ferrari and A. Bilal, hep-th/9602082;
- [8] T.D. Lee, Physics Reports C 9 (1974) 143.
- [9] G. Schierholz, Nucl. Phys. B (Proc. Suppl.) 37A (1994) 203.
- [10] P.A.M. Dirac, Proc. Roy. Soc. A 133 (1931) 60.
- [11] B. Zumino, Erice Lectures (1966), ed. A. Zichichi; S. Coleman, Erice Lectures (1977), ed. A. Zichichi.



ELSEVIER

Nuclear Physics B 534 (1998) 373–391

NUCLEAR  
PHYSICS B

# A lattice study of spectator effects in inclusive decays of $B$ -mesons

UKQCD Collaboration

M. Di Pierro, C.T. Sachrajda

*Department of Physics and Astronomy, University of Southampton, Southampton SO17 1BJ, UK*

Received 12 June 1998; accepted 14 August 1998

---

## Abstract

We compute the matrix elements of the operators which contribute to spectator effects in inclusive decays of  $B$ -mesons. The results agree well with estimates based on the vacuum saturation (factorization) hypothesis. For the ratio of lifetimes of charged and neutral mesons we find  $\tau(B^-)/\tau(B_d) = 1.03 \pm 0.02 \pm 0.03$ , where the first error represents the uncertainty in our evaluation of the matrix elements, and the second is an estimate of the uncertainty due to the fact that the Wilson coefficient functions have only been evaluated at tree-level in perturbation theory. This result is in agreement with the experimental measurement. We also discuss the implications of our results for the semileptonic branching ratio and the charm yield. © 1998 Elsevier Science B.V.

PACS: 12.38.Gc; 13.25.Hw; 14.40.Nd

Keywords: Inclusive beauty decays; Spectator effect; Lattice QCD

---

## 1. Introduction

Inclusive decays of heavy hadrons can be studied in the framework of the heavy-quark expansion, in which widths and lifetimes are computed as series in inverse powers of the mass of the  $b$ -quark [1–3] (for recent reviews and additional references see Refs. [4,5]). The leading term of this expansion corresponds to the decay of a free-quark and is universal, contributing equally to the lifetimes of all beauty hadrons. Remarkably there are no corrections of  $O(1/m_b)$ , and the first corrections are of  $O(1/m_b^2)$  [6,3]. “Spectator effects”, i.e. contributions from decays in which a light constituent quark also participates in the weak process, enter at third order in the heavy-quark expansion, i.e. at  $O(1/m_b^3)$ . However, as a result of the enhancement of the phase space for  $2 \rightarrow 2$

body reactions, relative to  $1 \rightarrow 3$  body decays, the spectator effects are likely to be larger than estimates based purely on power counting, and may well be significant. The need to evaluate the spectator effects is reinforced by the striking discrepancy between the experimental result for the ratio of lifetimes [7],

$$\frac{\tau(\Lambda_b)}{\tau(B_d)} = 0.78 \pm 0.04, \quad (1)$$

and the theoretical prediction,

$$\frac{\tau(\Lambda_b)}{\tau(B_d)} = 0.98 + O(1/m_b^3). \quad (2)$$

In order to explain this discrepancy in the conventional approach, the higher order terms in the heavy-quark expansion, and the spectator effects in particular, would have to be surprisingly large. In this paper we compute the matrix elements of the four-quark operators needed to evaluate these spectator effects for mesons; a computation of the effects for the  $\Lambda_b$  is in progress and the results will be reported in a future publication.

The experimental value of the ratio of lifetimes of the charged and neutral  $B$ -mesons is [7]

$$\frac{\tau(B^-)}{\tau(B_d^0)} = 1.06 \pm 0.04, \quad (3)$$

to be compared to the theoretical prediction

$$\frac{\tau(B^-)}{\tau(B_d^0)} = 1 + O(1/m_b^3). \quad (4)$$

Below we determine the contribution to the  $O(1/m_b^3)$  term on the right-hand side of Eq. (4) coming from spectator effects, which we believe to be the largest component. The same spectator effects also contribute to the semileptonic branching ratio of the  $B$ -mesons, and to the charm-yield ( $n_c$ , the average number of charmed particles in decays of  $B$ -mesons). This will be briefly discussed in Section 5.

At  $O(1/m_b^3)$ , the non-perturbative contributions to the spectator effects are contained in the matrix elements of the four-quark operators

$$O_{V-A}^q = \bar{b}_L \gamma_\mu q_L \bar{q}_L \gamma^\mu b_L, \quad (5)$$

$$O_{S-P}^q = \bar{b}_R q_L \bar{q}_L b_R, \quad (6)$$

$$T_{V-A}^q = \bar{b}_L \gamma_\mu T^a q_L \bar{q}_L \gamma^\mu T^a b_L, \quad (7)$$

$$T_{S-P}^q = \bar{b}_R T^a q_L \bar{q}_L T^a b_R, \quad (8)$$

where  $q$  represents the field of the light quark, and the  $T^a$ 's are the generators of the colour group. Throughout this paper we take these operators to be defined at a renormalization scale  $m_b$ . The subscripts  $L$  and  $R$  represent “left” and “right”, respectively. For mesons, following Ref. [9], we introduce the parameters  $B_{1,2}$  and  $\varepsilon_{1,2}$ :

$$\frac{1}{2m_{B_q}} \langle B_q | O_{V-A}^q | B_q \rangle \equiv \frac{f_{B_q}^2 m_{B_q}}{8} B_1, \quad (9)$$

$$\frac{1}{2m_{B_q}} \langle B_q | O_{S-P}^q | B_q \rangle \equiv \frac{f_{B_q}^2 m_{B_q}}{8} B_2, \quad (10)$$

$$\frac{1}{2m_{B_q}} \langle B_q | T_{V-A}^q | B_q \rangle \equiv \frac{f_{B_q}^2 m_{B_q}}{8} \varepsilon_1, \quad (11)$$

$$\frac{1}{2m_{B_q}} \langle B_q | T_{S-P}^q | B_q \rangle \equiv \frac{f_{B_q}^2 m_{B_q}}{8} \varepsilon_2, \quad (12)$$

where  $f_{B_q}$  is the leptonic decay constant of the meson  $B_q$  (in a normalization in which  $f_\pi \simeq 131$  MeV). The introduction of the parameters  $B_{1,2}$  and  $\varepsilon_{1,2}$  is motivated by the vacuum saturation (or factorization) approximation [10] in which the matrix elements of four-quark operators are evaluated by inserting the vacuum inside the current products. This leads to  $B_i = 1$  and  $\varepsilon_i = 0$  at some renormalization scale,  $\mu$ , for the operators. It may be argued that the scale at which the factorization approximation holds is different from our chosen renormalization scale  $m_b$ , specifically that if the approximation is valid at all then it should hold at a typical hadronic scale of about 1 GeV [11]. In QCD, in the limit of a large number of colours  $N_c$ ,

$$B_i = O(1) \text{ and } \varepsilon_i = O(1/N_c). \quad (13)$$

In this paper we evaluate the parameters  $B_i$  and  $\varepsilon_i$  in lattice simulations, thus testing the validity of the factorization hypothesis.

The results of our calculations indicate that the vacuum saturation hypothesis is (surprisingly?) well satisfied. In the  $\overline{\text{MS}}$  renormalization scheme we find

$$B_1(m_b) = 1.06(8), \quad B_2(m_b) = 1.01(6), \quad (14)$$

$$\varepsilon_1(m_b) = -0.01(3), \quad \varepsilon_2(m_b) = -0.01(2) \quad (15)$$

and

$$\frac{\tau(B^-)}{\tau(B_d)} = 1.03 \pm 0.02 \pm 0.03, \quad (16)$$

in good agreement with the experimental value in Eq. (3).

The present calculation is very similar to that of the  $B_B$  parameter of  $B-\bar{B}$  mixing, for which several recent simulations have been performed [12–14], including one using the same configurations used in this study [12]. We use the calculations of the  $B_B$  parameter as a comparison and check on our results, both for the perturbative matching coefficients and for the evaluation of the matrix elements. In performing this comparison we believe that we have found an error in the perturbative calculation. We also stress that the feature that the values of the matrix elements of the operators (5)–(8) are close to those expected from the vacuum saturation hypothesis is also present in the evaluation of the  $B_B$  parameter.

The plan of the remainder of this paper is as follows. In the next section we derive the relation, at one-loop order of perturbation theory, between the operators defined in Eqs. (5)–(8) and the bare lattice operators whose matrix elements are computed directly in lattice simulations. The technical details of the calculation are relegated to Appendix A. In Section 3 we present a description of the lattice computation, the results of this computation and a discussion of the implications. The comparison of our study to that of  $B^0$ – $\bar{B}^0$  mixing is presented in Section 4 (the discussion of the evaluation of the matching coefficients for  $B$ – $\bar{B}$  mixing is described in detail in Appendix B). Finally, in Section 5 we present our conclusions.

## 2. Perturbative matching

The Wilson coefficient functions of the operators (5)–(8) in the OPE for inclusive decay rates have been evaluated only at tree-level [9]. At this level of precision it is sufficient to compute the matrix elements in any “reasonable” renormalization scheme. In this paper we present the results for the matrix elements of the HQET operators in the  $\overline{\text{MS}}$  scheme at a renormalization scale  $m_b$ .<sup>1</sup> In order to obtain these from the matrix elements of bare operators in the lattice theory, with cut-off  $a^{-1}$  (where  $a$  is the lattice spacing), which we compute directly in our simulations, we require the corresponding matching coefficients. In this section we present these matching coefficients (at one-loop order in perturbation theory); we postpone a detailed description of their evaluation to Appendix A. In the following section we will use these coefficients and the computed values of the matrix elements of the bare operators to determine the coefficients  $B_i$  and  $\varepsilon_i$  in the  $\overline{\text{MS}}$  scheme.

The coefficients presented in the section were all obtained using local lattice operators defined in the tree-level improved SW action [15] defined in Eq. (21) below.

It is convenient to perform the matching in two steps:

- (i) The first step is the evaluation of the coefficients which relate the HQET operators in the continuum ( $\overline{\text{MS}}$ ) and lattice schemes, both defined at the scale  $a^{-1}$ ,

$$O_i^C(a^{-1}) = O_i^L(a^{-1}) + \frac{\alpha_s}{4\pi} D_{ij} O_j^L(a^{-1}), \quad (17)$$

where the lattice (continuum) operators are labelled by the superfix  $L$  ( $C$ ). The mixing of operators is such that it is necessary to compute the matrix elements of eight lattice operators  $O_j^L(a^{-1})$  (see Table 1, where the coefficients  $D_{ij}$  are presented).

- (ii) We then evolve the HQET operators in the continuum scheme from renormalization scale  $\mu = a^{-1}$  to scale  $\mu = m_b$ ,

$$O_i^C(m_b) = M_{ij}(a^{-1}, m_b) O_j^C(a^{-1}). \quad (18)$$

---

<sup>1</sup>From these it is possible to obtain the matrix elements in any other renormalization scheme using perturbation theory.

Table 1  
Coefficients  $D_{ij}$  for the four operators of Eqs. (5)–(8)

$D_{ij}$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$O_j^L$
$j = 1$	−21.64	−	2.06	0.54	$\bar{b}_L \gamma^\mu q_L \bar{q}_L \gamma^\mu b_L$
$j = 2$	−	−21.64	2.16	2.40	$\bar{b}_R q_L \bar{q}_L b_R$
$j = 3$	9.29	2.43	−10.80	2.83	$\bar{b}_L \gamma^\mu T^a q_L \bar{q}_L \gamma^\mu T^a b_L$
$j = 4$	9.72	10.79	11.34	−9.05	$\bar{b}_R T^a q_L \bar{q}_L T^a b_R$
$j = 5$	−18.37	−	−3.06	−	$\bar{b}_R \gamma^\mu q_R \bar{q}_L \gamma^\mu b_L$
$j = 6$	36.75	18.37	6.12	3.06	$\bar{b}_R q_L \bar{q}_R b_L$
$j = 7$	−13.78	−	6.89	−	$\bar{b}_R \gamma^\mu T^a q_R \bar{q}_L \gamma^\mu T^a b_L$
$j = 8$	27.56	13.78	−13.78	−6.89	$\bar{b}_R T^a q_L \bar{q}_R T^a b_L$

This evolution, sometimes known as “hybrid renormalization”, requires knowledge of the anomalous dimension matrix [16–18].

The matching procedure involves short-distance physics only, and so can be carried out in perturbation theory. Lattice perturbation theory generally converges very slowly, and therefore, where possible, the matching should be performed non-perturbatively so as to minimize these systematic errors. In this letter, however, we do not use a non-perturbative renormalization, but, when evaluating the matching coefficients, we do use a “boosted” lattice coupling constant in order to partially resum the large higher order contributions (e.g. those coming from tadpole graphs).

In order to obtain the parameters  $B_i$  and  $\varepsilon_i$  it is also necessary to determine the normalization of the axial current [19]. In this case there is a single coefficient  $Z_A$  defined by

$$\langle 0 | A_0^C(a^{-1}) | B \rangle = Z_A \langle 0 | A_0^L(a^{-1}) | B \rangle, \quad (19)$$

which at one-loop order in perturbation theory is given by

$$Z_A = 1 + \frac{\alpha_s(a^{-1})}{4\pi} \frac{4}{3} \left( \frac{5}{4} - \frac{1}{2}x_1 - x_2 + x_3 \right) \simeq 1 - 20.0 \frac{\alpha_s(a^{-1})}{4\pi}, \quad (20)$$

where the coefficients  $x_i$  are defined in Eq. (A.3) and tabulated in Table A.3 in Appendix A.  $A_0^C$  and  $A_0^L$  are the time components on the axial current in the HQET in the  $\overline{\text{MS}}$  and lattice schemes respectively, both defined at the scale  $a^{-1}$ . Using hybrid renormalization one can obtain the axial current in the  $\overline{\text{MS}}$  scheme at scale  $m_b$ . We stress again that the results presented for the  $B_i$ 's and  $\varepsilon_i$ 's below were obtained with both the four quark operators and the axial current defined in the HQET in the  $\overline{\text{MS}}$  scheme at the scale  $m_b$ .<sup>2</sup>

In the following section we combine the results of the matrix elements computed on the lattice with the perturbative coefficients presented in this section to obtain the  $B_i$ 's and  $\varepsilon_i$ 's.

<sup>2</sup> These differ at one-loop level from the corresponding operators in QCD.

### 3. Lattice computation and results

The non-perturbative strong interaction effects in spectator contributions to inclusive decays are contained in the matrix elements of the eight four-quark operators,  $O_j$ , given in Table 1. We evaluate these matrix elements in a quenched simulation on a  $24^3 \times 48$  lattice at  $\beta = 6.2$  using the SW tree-level improved action [15],

$$S^{\text{SW}} = S^{\text{gauge}} + S^{\text{Wilson}} - i \frac{\kappa}{2} \sum_{x,\mu,\nu} \bar{q}(x) F_{\mu\nu}(x) \sigma_{\mu\nu} q(x), \quad (21)$$

where  $S^{\text{gauge}}$  and  $S^{\text{Wilson}}$  are the Wilson gauge and quark actions, respectively. The use of this action reduces the errors due to the granularity of the lattice to ones of  $O(\alpha_s a)$ , where  $a$  is the lattice spacing. We use the 60 SU(3) gauge-field configurations, and the light quark propagators corresponding to hopping parameters  $\kappa = 0.14144, 0.14226$  and  $0.14262$  which have been used previously to obtain the  $B$ -parameter of  $B^0 - \bar{B}^0$  mixing [12] and other quantities required for studies of  $B$ -physics. The value of the hopping parameter in the chiral limit is given by  $\kappa_c = 0.14315$ . The calculation of the matrix elements of the operators in Table 1 is very similar to that of the  $\Delta B = 2$  operators from which the  $B$ -parameter is extracted, and we exploit the similarities to perform checks on our calculations (see Section 4 and Appendix B).

The evaluation of the matrix elements requires the computation of two- and three-point correlation functions,

$$C(t_x) \equiv \sum_x \langle 0 | J(x) J^\dagger(0) | 0 \rangle, \quad (22)$$

where we have assumed that  $t_x > 0$ , and

$$K(t_x, t_y) \equiv \sum_{x,y} \langle 0 | J(y) O_j^L(0) J^\dagger(x) | 0 \rangle, \quad (23)$$

where  $t_y > 0 > t_x$ . In Eqs. (22) and (23)  $J^\dagger$  and  $J$  are interpolating operators which can create or destroy a heavy pseudoscalar meson (containing a static heavy quark) and  $O_j$  is one of the operators whose matrix element we wish to evaluate. In practice we choose  $J$  to be the fourth component of an axial current. It is generally advantageous to “smear” the interpolating operators  $J$  and  $J^\dagger$  over the spatial coordinates, in order to enhance the overlap with the ground state. Following Ref. [12] we have used several methods of smearing and checked that the results for the matrix elements of  $O_j$  are independent of the choice of smearing. In this paper we present as our “best” results those obtained using a gauge-invariant smearing based on the Jacobi algorithm described in Ref. [20]. The smeared heavy-quark field at time  $t$ ,  $b^S(x, t)$ , is defined by

$$b^S(x) = \sum_{x'} M(x, x') b(x', t), \quad (24)$$

where

$$M(x, x') = \sum_{n=0}^N \kappa_s \Delta^n(x, x') \quad (25)$$

and  $\Delta$  is the three-dimensional version of the Laplace operator. The parameter  $\kappa_s$  and the number of iterations can be used to control the smearing radius, and here we present our results for  $\kappa_s = 0.25$  and  $N = 140$  which corresponds to an rms smearing radius  $r_0 = 6.4a$  [21]. The smeared interpolating operator is then chosen to be  $J^S(x) = \bar{b}^S(x)\gamma_5 q(x)$ , where  $q$  is the field of the light quark. The local interpolating operator is simply defined to be  $J^L(x) = \bar{b}(x)\gamma_5 q(x)$ .

The eight matrix elements  $\langle B | O_j^L(0) | B \rangle$  are determined by computing the ratios:

$$R_j(t_1, t_2) \equiv \frac{4K_j^{SS}(-t_1, t_2)}{C^{LS}(-t_1)C^{LS}(t_2)}, \quad (26)$$

where  $t_1$  and  $t_2$  are positive and sufficiently large to ensure that only the ground state contributes to the correlation functions. The indices  $S$  and  $L$  denote whether the interpolating operators are smeared or local. It is convenient to choose both  $J$  and  $J^\dagger$  to be smeared in the three-point function  $K_j$  and to evaluate the two-point functions with a local operator at the source and a smeared one at the sink. At large time separations  $t_1$  and  $t_2$ ,

$$R_j(t_1, t_2) \rightarrow \frac{2}{m_B Z_L^2} \langle B | O_j^L(0) | B \rangle, \quad (27)$$

where  $Z_L$  is given by

$$\langle 0 | J^L(0) | B \rangle \equiv \sqrt{2m_B} Z_L. \quad (28)$$

The  $R_j$  defined in Eq. (26) contribute directly to the  $B_i$ 's and the  $\varepsilon_i$ 's, apart from perturbative matching factors. To see this, note that the leptonic decay constant of the  $B$ -meson in the static limit (i.e. infinite mass limit for the  $b$ -quark) is given by

$$f_B^2 = \frac{2Z_L^2 Z_A^2}{m_B}, \quad (29)$$

so that

$$R_j(t_1, t_2) \rightarrow \frac{2}{m_B Z_L^2} \langle B | O_j^L(0) | B \rangle = \frac{4Z_A^2}{f_B^2 m_B^2} \langle B | O_j^L(0) | B \rangle, \quad (30)$$

which corresponds precisely to the normalization of the  $B_i$ 's and  $\varepsilon_i$ 's in Eqs. (9)–(12) (apart from the matching factors  $Z_A^2$  and that of the four-quark operator  $O_j$  described in Section 2).

In these computations it is particularly important to establish that the contribution from the ground-state has been isolated. The natural way to do this is to look for plateaus, i.e. for regions in  $t_1$  and  $t_2$  for which  $R_j(t_1, t_2)$  is independent of  $t_1$  and  $t_2$ . Since the statistical errors grow fairly quickly with  $t_{1,2}$  our ability to verify the existence of plateaus is limited. For example in Fig. 1 we present our results for  $R_j(t_1, t_2)$ , obtained with the

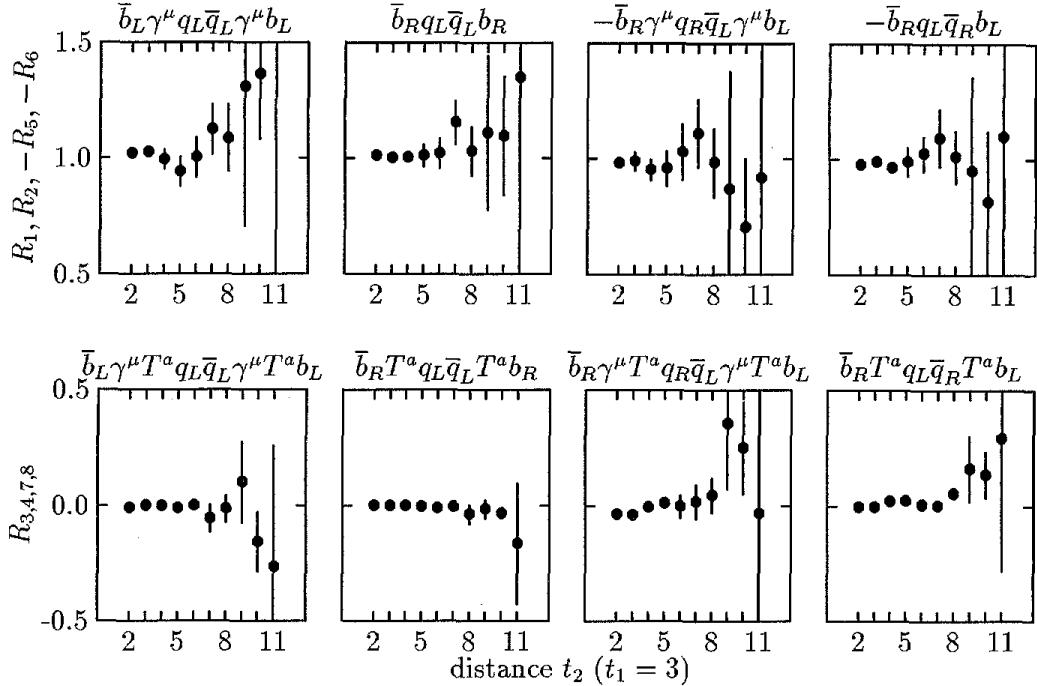


Fig. 1. Results for the  $R_j$ ,  $j = 1, \dots, 8$  as a function of  $t_2$  for  $t_1 = 3$ . The value of the quark mass corresponds to  $\kappa = 0.14226$ .

light quark mass corresponding to  $\kappa = 0.14226$  as a function of  $t_2$  for  $t_1 = 3$ . The results are consistent with being constant, but the errors are uncomfortably large for  $t_2 > 5$  or so. An alternative, and perhaps more convincing, way to confirm that the contribution from the ground-state has been isolated is to check that the values of the  $R_j$  are independent of the method of smearing used to define the smeared interpolating operators. Following Ref. [12], in addition to the gauge invariant prescription for smearing described above, we have defined the smeared field in four other ways (having transformed the fields to the Coulomb gauge). For these additional four cases,  $M(\mathbf{x}, \mathbf{x}')$  of Eq. (25) is replaced by the following:

$$\text{Exponential : } M(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|/r_0), \quad (31)$$

$$\text{Gaussian : } M(\mathbf{x}, \mathbf{x}') = \exp(-|\mathbf{x} - \mathbf{x}'|^2/r_0^2), \quad (32)$$

$$\text{Cube : } M(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^3 \Theta(r_0 - |x_i - x'_i|), \quad (33)$$

$$\text{DoubleCube : } M(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^3 \left(1 - \frac{|x_i - x'_i|}{2r_0}\right) \Theta(2r_0 - |x_i - x'_i|), \quad (34)$$

and we have chosen  $r_0 = 5$ . As an example, we present in Table 2 the results for the  $R_j(t_1, t_2)$  for  $t_1 = 3$  and  $t_2 = 3$ , again for the middle value of the three  $\kappa$ 's,  $\kappa = 0.14226$ .

We take as our best results those obtained with the gauge invariant smearing, and with  $t_1 = t_2 = 3$ . After extrapolating to the chiral limit we find the following results for

Table 2

Values of the  $R_j(3, 3)$  obtained with different smearing methods for a light quark with  $\kappa = 0.14226$ 

	Gauge Inv.	Exponential	Gaussian	Cube	Double cube
$R_1(3, 3)$	1.02(3)	1.03(2)	1.04(3)	1.03(2)	1.07(8)
$R_2(3, 3)$	1.00(2)	1.01(2)	1.01(3)	1.01(2)	1.00(5)
$R_3(3, 3)$	0.00(1)	-0.00(1)	-0.00(2)	-0.00(1)	-0.03(3)
$R_4(3, 3)$	0.00(1)	0.00(1)	0.01(1)	0.00(1)	0.01(2)
$R_5(3, 3)$	-0.99(4)	-1.01(3)	-1.01(4)	-1.00(3)	-1.01(7)
$R_6(3, 3)$	-0.99(3)	-1.00(2)	-1.01(3)	-1.00(3)	-1.01(6)
$R_7(3, 3)$	-0.04(2)	-0.03(2)	-0.04(2)	-0.03(2)	-0.09(5)
$R_8(3, 3)$	-0.00(1)	-0.00(1)	-0.01(1)	-0.01(1)	-0.02(2)

the  $R_j$ :

$$R_1(3, 3) = 1.04 \pm 0.04, \quad R_2(3, 3) = 1.00 \pm 0.03, \quad (35)$$

$$R_3(3, 3) = -0.01 \pm 0.02, \quad R_4(3, 3) = -0.00 \pm 0.01, \quad (36)$$

$$R_5(3, 3) = -0.97 \pm 0.06, \quad R_6(3, 3) = -0.98 \pm 0.05, \quad (37)$$

$$R_7(3, 3) = -0.03 \pm 0.03, \quad R_8(3, 3) = -0.01 \pm 0.01. \quad (38)$$

We now combine these results with the matching coefficients presented in Section 2 to determine the  $B_i$ 's and the  $\varepsilon_i$ 's. For the value of the lattice coupling constant we take one of the standard definitions of the boosted coupling:

$$\frac{\alpha_s(a^{-1})}{4\pi} = \frac{6(8\kappa_c)^4}{(4\pi)^2\beta} = 0.0105. \quad (39)$$

Using the coefficients from Section 2 we then obtain

$$B_1(a^{-1}) = (Z_A)^{-2} \left( 1 + \frac{\alpha_s(a^{-1})}{4\pi} D \right)_{1j} R_j = 0.98(8),$$

$$B_2(a^{-1}) = (Z_A)^{-2} \left( 1 + \frac{\alpha_s(a^{-1})}{4\pi} D \right)_{2j} R_j = 0.93(5),$$

$$\varepsilon_1(a^{-1}) = (Z_A)^{-2} \left( 1 + \frac{\alpha_s(a^{-1})}{4\pi} D \right)_{3j} R_j = 0.01(3),$$

$$\varepsilon_2(a^{-1}) = (Z_A)^{-2} \left( 1 + \frac{\alpha_s(a^{-1})}{4\pi} D \right)_{4j} R_j = 0.00(2).$$

We now evolve those coefficients to the scale  $m_b$ , using the relations

$$B(m_b) = \left[ 1 + \frac{2C_F\delta}{N_c} \right] B(a^{-1}) - \frac{2\delta}{N_c} \varepsilon(a^{-1}), \quad (40)$$

$$\varepsilon(m_b) = \left[ 1 + \frac{\delta}{N_c^2} \right] \varepsilon(a^{-1}) - \frac{C_F\delta}{N_c^2} B(a^{-1}), \quad (41)$$

where

$$\delta \equiv \left( \frac{\alpha_s(a^{-1})}{\alpha_s(m_b)} \right)^{9/2\beta_0} - 1 = 0.09(3). \quad (42)$$

In estimating  $\delta$  and its uncertainty we have allowed for a conservative variation of the parameters around the “central” values ( $\Lambda_{\text{QCD}} = 250$  MeV,  $a^{-1} = 2.9$  GeV,  $m_b = 4.5$  GeV and  $\beta_0 = 9$ ). We finally obtain

$$B_1(m_b) = 1.06(8), \quad B_2(m_b) = 1.01(6), \quad (43)$$

$$\varepsilon_1(m_b) = -0.01(3), \quad \varepsilon_2(m_b) = -0.01(2). \quad (44)$$

These matrix elements have also been evaluated using QCD sum-rules [22]. These authors find  $B_1(m_b) = 0.96(4)$ ,  $B_2(m_b) = 0.95(2)$ ,  $\varepsilon_1(m_b) = -0.14(1)$  and  $\varepsilon_2(m_b) = -0.08(1)$ , differing somewhat (particularly for the  $\varepsilon_i$ 's) from our values.

Using the results in Eqs. (43) and (44) we obtain the following value for the ratio of lifetimes for the neutral and charged  $B$ -mesons:

$$\frac{\tau(B^-)}{\tau(B_d)} = 1 + k_1 B_1 + k_2 B_2 + k_3 \varepsilon_1 + k_4 \varepsilon_2 = 1.03 \pm 0.02 \pm 0.03, \quad (45)$$

where the coefficients  $k_i$  are taken from Ref. [9]. The first error in Eq. (45) is from the uncertainty in the values of the matrix elements in Eqs. (43) and (44), whereas the second is an estimate of the uncertainty due to our ignorance of the one-loop contribution to the Wilson coefficient function in the OPE (this was estimated by varying the matching scale from  $m_b/2$  to  $2m_b$  using the procedure described in Ref. [9]). The value in Eq. (45) is in good agreement with the experimental results in Eq. (3).

There has been a considerable amount of discussion lately as to whether the theoretical predictions for the semileptonic branching ratio of the  $B$ -meson ( $B_{SL}$ ) and the charm yield are consistent with experimental measurements [23,24,9]. Here we limit our discussion to a comment on the implications of our results on these two physical quantities. The spectator contributions to  $B_{SL}$  and  $n_c$  take the form

$$\Delta B_{SL,\text{spec}} = b_1 B_1 + b_2 B_2 + b_3 \varepsilon_1 + b_4 \varepsilon_2, \quad (46)$$

$$\Delta n_{c,\text{spec}} = n_1 B_1 + n_2 B_2 + n_3 \varepsilon_1 + n_4 \varepsilon_2, \quad (47)$$

where the coefficients  $b_i$  and  $n_i$  at tree-level are presented explicitly in Ref. [9] (see also Refs. [11,25,16]). The uncertainties in the predictions for  $B_{SL}$  and  $n_c$  due to the errors in the matrix elements in Eqs. (43) and (44) are small (about 0.1% for both  $B_{SL}$  and  $n_c$ ). Larger uncertainties are due to our ignorance of the one-loop contribution to the Wilson coefficient function in the OPE. For example, following the procedure described in Ref. [9] with a matching scale  $\mu = m_b/2$  we find  $\Delta B_{SL,\text{spec}} = 0.3(1)\%$  and  $\Delta n_{c,\text{spec}} = 0.5(2)\%$ , whereas for a scale  $2m_B$  the corresponding results are  $\Delta B_{SL,\text{spec}} = -0.1(1)\%$  and  $\Delta n_{c,\text{spec}} = 0.0(1)\%$ . For the charm yield these corrections are negligible as expected, and for the semileptonic branching ratio they are small. Nevertheless, it would be useful to know the one-loop contribution to the coefficient functions in order to eliminate the variation obtained by changing the matching scale.

#### 4. $B-\bar{B}$ mixing

In this section we revisit the phenomenologically important process of  $B-\bar{B}$  mixing. The computation of the matrix elements of the relevant lattice operators on the same gauge configurations has already been presented in Ref. [12], and we do not add to these lattice computations. We do, however, wish to make two observations:

- (i) The matrix elements of the lattice  $\Delta B = 2$  operators relevant for  $B-\bar{B}$  mixing factorize with a similar precision to that found for the four-quark operators considered in Section 3. Because of the way in which lattice computations are usually organized, this property is not as readily manifest for the  $\Delta B = 2$  operators as it is for the spectator effects. However, by using colour and spin Fierz identities, we demonstrate that the values of the matrix elements of the lattice  $\Delta B = 2$  operators are very close to those expected using factorization. Some of our observations have already been noted in Ref. [14], where the Wilson formulation for the light quarks was used. We extend this investigation of factorization, and show that each contribution to the matrix elements (i.e. each Wick contraction) is close to the estimated value obtained using the factorization and vacuum saturation hypothesis.
- (ii) We believe that there is an error in the published value of the matching factors for the  $\Delta B = 2$  operators using the SW action for the light quarks. We discuss this in some detail in Appendix B; in this section we briefly comment on the consequences of this error.

We now consider these two observations in turn.

##### 4.1. Factorization

In order to determine the  $B$ -parameter of  $B-\bar{B}$  mixing we need to evaluate the matrix elements of the three lattice operators  $O_L$ ,  $O_R$  and  $O_N$  defined in Eqs. (B.1)–(B.3) of Appendix B, as well as that of

$$O_S = \bar{b}q_L \bar{b}q_L. \quad (48)$$

We now show that these matrix elements (and related ones) are reproduced remarkably accurately by assuming the factorization hypothesis and vacuum saturation. For each of these operators ( $O_j$ ) we define the ratio  $R_j(t_1, t_2)$  analogously to Eq. (26) as follows:

$$R_j(t_1, t_2) \equiv \frac{3}{8} \frac{K_j^{SS}(t_1, t_2)}{C^{LS}(-t_1)C^{LS}(t_2)}, \quad (49)$$

where the correlation functions  $C$  and  $K_j$  are defined in Eqs. (22) and (23), except that in the  $K_j$  the operators  $O_j$  are now  $\Delta B = 2$  operators, and the interpolating operators destroy a  $B$ -meson and create a  $\bar{B}$ -meson.

We start by explaining explicitly what we mean by factorization (combined with vacuum saturation). The evaluation of the  $B$ -parameter requires the computation of

three-point functions  $K_j(t_1, t_2)$ , which are of the form<sup>3</sup>

$$\langle 0 | \bar{q}(y) \gamma^5 b(y) \bar{b}(0) \Gamma q(0) \bar{b}(0) \tilde{\Gamma} q(0) \bar{q}(x) \gamma^5 b(x) | 0 \rangle. \quad (50)$$

There are four Wick contractions which contribute to the correlation function, and it is convenient to track these by introducing a fictitious quantum number, labelled by an integer suffix on each field, so that the correlation function is written in the form

$$\langle 0 | \bar{q}_2(y) \gamma^5 b_1(y) O_{\Gamma, \tilde{\Gamma}}(0) \bar{q}_4(x) \gamma^5 b_3(x) | 0 \rangle, \quad (51)$$

where

$$O_{\Gamma, \tilde{\Gamma}} \equiv \bar{b}_1 \Gamma q_2 \bar{b}_3 \tilde{\Gamma} q_4 + \bar{b}_3 \Gamma q_4 \bar{b}_1 \tilde{\Gamma} q_2 - \bar{b}_1 \Gamma q_4 \bar{b}_3 \tilde{\Gamma} q_2 - \bar{b}_3 \Gamma q_2 \bar{b}_1 \tilde{\Gamma} q_4, \quad (52)$$

and  $\Gamma, \tilde{\Gamma}$  are Dirac matrices. The contraction of spinor  $(\alpha, \beta)$  and colour  $(a)$  indices in each bilinear in Eq. (52) is implicit (e.g.  $\bar{b}_1 \Gamma q_2 = \bar{b}_{1,\alpha}^a \Gamma_{\alpha,\beta} q_{2,\beta}^a$ ). Only contractions between fields with the same suffix are allowed, and the four terms in Eq. (52) correspond to the four original Wick contractions. For all the operators of interest, by using colour and spin Fierz identities, it is possible to rewrite each of the four terms into sums of operators of the form  $(\bar{b}_1 \Gamma' q_2) (\bar{b}_3 \tilde{\Gamma}' q_4)$  and  $(\bar{b}_1 \Gamma' T^a q_2) (\bar{b}_3 \tilde{\Gamma}' T^a q_4)$ , for some  $\gamma$ -matrices  $\Gamma'$  and  $\tilde{\Gamma}'$ . In the factorization and vacuum saturation hypothesis

$$\langle \bar{B} | (\bar{b}_1 \Gamma' T^a q_2) (\bar{b}_3 \tilde{\Gamma}' T^a q_4) | B \rangle \simeq 0, \quad (53)$$

and

$$\langle \bar{B} | (\bar{b}_1 \Gamma' q_2) (\bar{b}_3 \tilde{\Gamma}' q_4) | B \rangle \simeq \langle \bar{B} | (\bar{b}_1 \Gamma' q_2) | 0 \rangle \langle 0 | (\bar{b}_3 \tilde{\Gamma}' q_4) | B \rangle. \quad (54)$$

Lorentz invariance implies that each of the matrix elements on the right-hand side of Eq. (54) vanishes or is proportional to the leptonic decay constant  $f_B$ .

We now consider each of the operators in turn:

- $O_L$  and  $O_R$ : The factor of 3/8 in Eq. (49) was chosen so that the factorization hypothesis gives  $R_L = R_R = 1$ . Numerically we find

$$\frac{R_L(3,3) + R_R(3,3)}{2} = 0.95 \pm 0.03. \quad (55)$$

- $O_S$ : Lorentz invariance implies that the matrix element  $\langle B | \bar{b} \sigma^{\mu\nu} (1 - \gamma^5) q | 0 \rangle$  vanishes. Using this fact we deduce that factorization implies that  $R_S \simeq 5/8$ . The numerical result for  $R_S$  is 0.60(3), in very good agreement with the estimate based on factorization.
- $O_N$ : Finally factorization implies that  $R_N \simeq 1$ , in good agreement with the numerical value 0.97(4).

In order to illustrate further that the numerical results quoted above are in agreement with expectations based on factorization and vacuum saturation we present separately in Fig. 2 the contributions to each of the ratios from the two independent contractions:

$$R^a = \langle \bar{B} | (\bar{b}_1 \Gamma' q_2) (\bar{b}_3 \tilde{\Gamma}' q_4) | B \rangle \quad (56)$$

---

<sup>3</sup>The extension of this discussion to include smeared interpolating operators is completely straightforward.

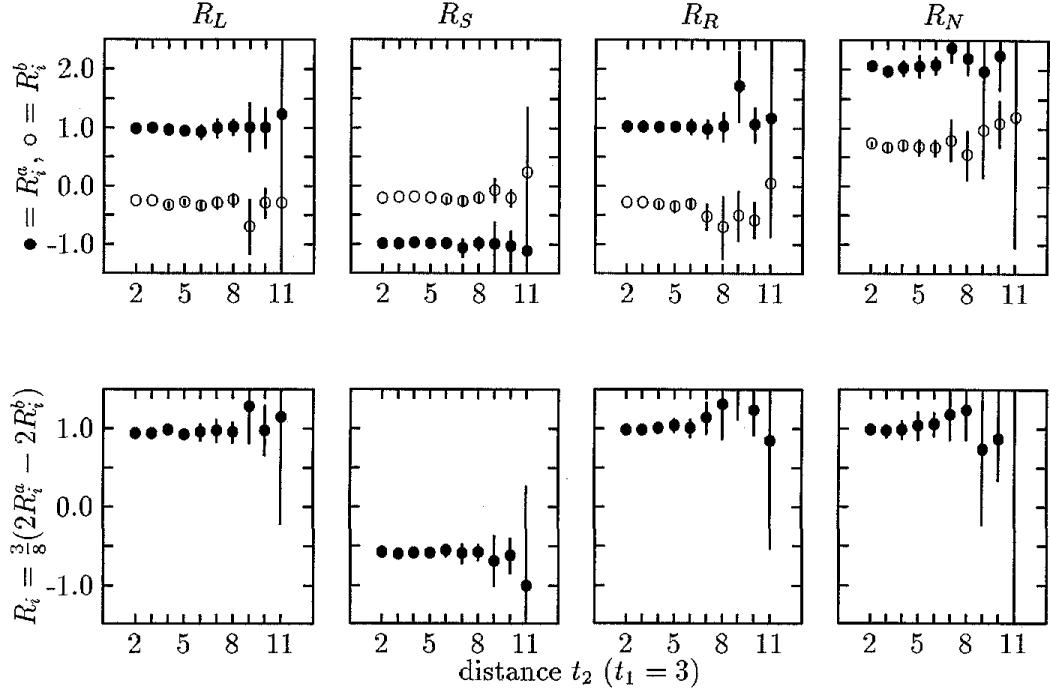


Fig. 2. Results for the  $R_j^a$ ,  $R_j^b$  and  $R_j$  ( $j = L, S, R, N$ ) as a function of  $t_2$  for  $t_1 = 3$ . The value of the quark mass corresponds to  $\kappa = 0.14226$ .

and

$$R^b = \langle \bar{B} | (\bar{b}_1 \Gamma' q_4) (\bar{b}_3 \tilde{\Gamma}' q_2) | B \rangle, \quad (57)$$

as well as the total combination

$$R = \frac{3}{8}(2R^a - 2R^b). \quad (58)$$

We see that not only are the values of the  $R$ 's quoted above in agreement with expectations from factorization, but the values of each of the  $R^a$  and  $R^b$  are also those we would expect on this assumption:

$$R_L^a \simeq 1, \quad R_L^b \simeq -\frac{1}{3}R_L^a, \quad R_S^a \simeq -1, \quad R_S^b \simeq \frac{1}{6}R_S^a, \quad (59)$$

$$R_R^a \simeq 1, \quad R_R^b \simeq -\frac{1}{3}R_R^a, \quad R_N^a \simeq 2, \quad R_N^b \simeq \frac{1}{3}R_N^a. \quad (60)$$

In this subsection we have demonstrated that the surprising precision of predictions for matrix elements obtained using the factorization hypothesis in inclusive decays, which was discussed in Section 3, also applies to existing results for  $B-\bar{B}$  mixing.

#### 4.2. Matching

In this subsection we discuss very briefly the implications of the error in the published value of the perturbative matching factors in  $B-\bar{B}$  mixing (see Appendix B). For example, we take method (a) of Ref. [12], in which each of the ratios  $R_i$  is fitted separately, and find

$$B_b(m_b) = 0.66(2), \quad B_b = \alpha_s^{-6/23} B_b(m_b) = 0.96(3), \quad (61)$$

where  $B_b(m_b)$  and  $B_b$  are the  $B$ -parameter in the  $\overline{\text{MS}}$  scheme at scale  $m_b$  and the renormalization group invariant  $B$ -parameter, respectively. For the comparison we only quote the statistical error. In order to see the effect of the error in the matching coefficient, the result in Eq. (61) should be compared with that obtained in Ref. [12] using an identical procedure (except for the values of the matching coefficients),

$$B_b(m_b) = 0.69(2), \quad B_b = \alpha_s^{-6/23} B_b(m_b) = 1.02(3). \quad (62)$$

Giménez and Martinelli had pointed out that the authors of Ref. [12] had used matching coefficient which did not include the contributions to the  $O(a^2)$  terms in the lattice operators which they had used [13]. As explained in Ref. [13], this leads to a negligible correction to the  $B$ -parameter. The differences in the values in Eqs. (61) and (62) is therefore largely due to the error discussed in Appendix B.

## 5. Conclusions

In this paper we have evaluated the matrix elements which contain the non-perturbative QCD effects in the spectator contributions to inclusive decays of  $B$ -mesons. For the  $\overline{\text{MS}}$  scheme our principal results are contained in Eqs. (43) and (44). The raw lattice results from which the matrix elements in any renormalization scheme can be determined are given in Eqs. (35)–(38).

The results for the matrix elements are very close to those which would be expected on the basis of factorization and the vacuum saturation hypothesis (particularly for the matrix elements of the bare lattice operators). We find the extent to which this hypothesis is satisfied to be surprising, but point out in Subsection 4.1 that this was also the case for the  $\Delta B = 2$  operators which contribute to  $B-\bar{B}$  mixing, whose matrix elements have been computed by several groups.

The calculation described in this paper is the first evaluation, using lattice simulations, of the matrix elements of the operators in Eqs. (5)–(8) between  $B$ -meson states. The errors in the results presented in Eqs. (35)–(38) are reasonably small, and until unquenched computations become possible, it is not phenomenologically necessary to invest a large effort to reduce the statistical errors in this quenched calculation. It is desirable, however, to establish, beyond any doubt, that the ground state meson has been isolated and to confirm explicitly the validity of the arguments presented in Section 3 that this is indeed the case. For this a similar calculation on a larger statistical sample will be required. It would also be very useful to evaluate the one-loop contributions to the Wilson coefficient functions in the OPE for inclusive decay rates. It is probable that these corrections currently represent the largest uncertainty in the inclusive rates.

A similar study of the matrix elements of the operators Eqs. (35)–(38) between  $A_b$  states is in progress. This is particularly important in view of the discrepancy between the experimental measurement in Eq. (1) and the theoretical prediction in Eq. (2) for the

ratio  $\tau(\Lambda_b)/\tau(B_d)$ . This calculation will help determine whether the discrepancy is due to baryonic matrix elements being larger than expected from quark models, or whether local quark-hadron duality, assumed in the phenomenology, is not valid. Technically the baryonic matrix elements of four-quark operators are more complicated to evaluate than mesonic ones, since it is not sufficient to generate the set of light-quark propagators from an arbitrary lattice point to the origin. The results will be presented in a following publication.

## Appendix A. Evaluation of the $D_{ij}$

In this appendix we discuss the evaluation of the coefficients  $\{D_{ij}\}$  of Eq. (17) for a generic four-quark operator  $O_i$ . This requires the evaluation of one-loop corrections to the matrix elements of the operators  $\{O_i\}$  in both the continuum and lattice schemes,

$$D_{ij} = C_{ij}^C - C_{ij}^L, \quad (\text{A.1})$$

where the superscripts  $C$  and  $L$  refer to the continuum renormalization scheme ( $\overline{\text{MS}}$ ) and the lattice regularization, respectively. We evaluate the matrix elements between four-quark states at zero momentum, regulating the infrared divergences by giving the gluon a small mass  $\lambda$ . The coefficients  $\{D_{ij}\}$  do not depend on the infra-red regulator, although each of  $C_{ij}^C$  and  $C_{ij}^L$  are separately infrared divergent.

We distinguish five categories of one-loop corrections to the generic operator

$$O_i \equiv \bar{b} \Gamma_i^A q \bar{q} \tilde{\Gamma}_i^A b, \quad (\text{A.2})$$

where  $A$  represents both colour and spinor indices:

- (1) Corrections to the external lines. These are proportional to  $C_F$ , the eigenvalue of the quadratic Casimir operator in the fundamental representation ( $C_F = 4/3$  for the  $SU(3)$  colour group), and are independent of  $\Gamma_i$  and  $\tilde{\Gamma}_i$ .
- (2) Vertex corrections to each of the quark bilinears.
- (3) Corrections in which the gluon couples to both heavy-quark propagators.
- (4) Corrections in which the gluon couples to one light-quark propagator and one heavy-quark propagator (at the other vertex).
- (5) Corrections in which the gluon couples to both light-quark propagators.

In Table A.1 we present the values of the contributions to the  $C_{ij}^C$  in the  $\overline{\text{MS}}$  scheme. For the operators  $O_1$  and  $O_3$ ,  $\Omega_1 = -\frac{1}{2}$  and  $\Omega_2 = 0$ , whereas for  $O_2$  and  $O_4$   $\Omega_1 = 1$  and  $\Omega_2 = 0$ . For a general choice of  $O_i$ ,  $\Omega_{1,2}$  depend on the precise definition of the operator  $\bar{b}(\Gamma_i^A \sigma^{\mu\nu}) q \bar{q}(\sigma^{\nu\mu} \tilde{\Gamma}_i^A) b$  and on the choice of basis for the  $\gamma$ -matrices in  $D$ -dimensions.

The coefficients  $\{D_{ij}\}$  themselves are presented in Table A.2, where the  $x_i$ 's are defined as

$$x_i \equiv c_i + c_i^I, \quad (\text{A.3})$$

and the numerical values of the  $c_i$  and  $c_i^I$  are tabulated in Table A.3. The contributions proportional to the  $x_i$ 's come from the diagrams in the lattice regularization. The com-

Table A.1  
Values of the contributions to  $C_{ij}^C$

(Category)	$C_{ij}^C$	$O_j^L$
(1)	$-\log(\lambda^2 a^2) + \frac{1}{2}$	$C_F \bar{b} \Gamma_i^A q \bar{q} \tilde{\Gamma}_i^A b$
(2)	$-\log(\lambda^2 a^2) + 1$	$\bar{b}(T^a \Gamma_i^A T^a) q \bar{q} \tilde{\Gamma}_i^A b + \bar{b} \Gamma_i^A q \bar{q} (T^a \tilde{\Gamma}_i^A T^a) b$
(3, odd)	$2 \log(\lambda^2 a^2)$	$\bar{b}(T^a \Gamma_i^A) q \bar{q} (\tilde{\Gamma}_i^A T^a) b$
(4, odd)	$\log(\lambda^2 a^2) - 1$	$\bar{b}(\Gamma_i^A T^a) q \bar{q} (\tilde{\Gamma}_i^A T^a) b + \bar{b}(T^a \Gamma_i^A) q \bar{q} (T^a \tilde{\Gamma}_i^A) b$
(5, odd)	$-\log(\lambda^2 a^2) + \Omega_1$	$\bar{b}(\Gamma_i^A T^a) q \bar{q} (T^a \tilde{\Gamma}_i^A) b$
(5, odd)	$-3 \log(\lambda^2 a^2) + \Omega_2$	$\frac{1}{12} \bar{b}(\Gamma_i^A \sigma^{\mu\nu} T^a) q \bar{q} (T^a \sigma^{\nu\mu} \tilde{\Gamma}_i^A) b$

Table A.2  
Values of the contributions to  $D_{ij}$

$j$ (Category)	$D_{ij}$	$O_j^L$
1 (1, 2)	$\frac{1}{2} - x_1$	$C_F \bar{b} \Gamma_i^A q \bar{q} \tilde{\Gamma}_i^A b$
2 (2)	$1 - x_2$	$\bar{b}(T^a \Gamma_i^A T^a) q \bar{q} \tilde{\Gamma}_i^A b + \bar{b} \Gamma_i^A q \bar{q} (T^a \tilde{\Gamma}_i^A T^a) b$
3 (2)	$-x_3$	$\bar{b}(T^a \gamma^0 \Gamma_i^A \gamma^0 T^a) q \bar{q} \tilde{\Gamma}_i^A b + \bar{b} \Gamma_i^A q \bar{q} (T^a \gamma^0 \tilde{\Gamma}_i^A \gamma^0 T^a) b$
4 (3, odd)	$-x_4$	$\bar{b}(T^a \Gamma_i^A) q \bar{q} (\tilde{\Gamma}_i^A T^a) b$
5 (4, odd)	$-1 - x_5$	$\bar{b}(\Gamma_i^A T^a) q \bar{q} (\tilde{\Gamma}_i^A T^a) b + \bar{b}(T^a \Gamma_i^A) q \bar{q} (T^a \tilde{\Gamma}_i^A) b$
6 (4)	$-x_6$	$\bar{b}(\Gamma_i^A \gamma^0 T^a) q \bar{q} (\tilde{\Gamma}_i^A \gamma^0 T^a) b + \bar{b}(T^a \gamma^0 \Gamma_i^A) q \bar{q} (T^a \gamma^0 \tilde{\Gamma}_i^A) b$
7 (5, odd)	$\Omega_1 - x_7$	$\bar{b}(\Gamma_i^A T^a) q \bar{q} (T^a \tilde{\Gamma}_i^A) b$
8 (5, odd)	$\Omega_2 - x_8$	$\frac{1}{12} \bar{b}(\Gamma_i^A \sigma^{\mu\nu} T^a) q \bar{q} (T^a \sigma^{\nu\mu} \tilde{\Gamma}_i^A) b$
9 (5)	$-x_9$	$\frac{1}{4} \bar{b}(\Gamma_i^A \gamma^\mu T^a) q \bar{q} (T^a \gamma^\mu \tilde{\Gamma}_i^A) b$

ponents proportional to the  $c_i$  would be the results if the Wilson formulation of the quark action had been used, and those proportional to  $c_i^I$  are the additional contributions which result from the use of the SW-improved action.<sup>4</sup>

<sup>4</sup> In the notation of Ref. [19], the parameters  $\delta_v$  and  $\delta_s$  are defined by

$$\delta_v \equiv \frac{1}{\pi^2} \int_{-\pi}^{+\pi} d^4 k \left( \frac{\Delta_4(4 - \Delta_1) - (\Delta_4 - \Delta_5)}{4\Delta_1\Delta_2^2} - 3 \frac{\theta(1 - k^2)}{k^4} \right),$$

$$\delta_s \equiv \frac{r^2}{4\pi^2} \int_{-\pi}^{+\pi} d^4 k \frac{4\Delta_1(\Delta_4 - \Delta_5)(2 + r^2\Delta_1) - \Delta_4^2(2 + r^2\Delta_1)}{4\Delta_1\Delta_2^2}.$$

Table A.3

Values of the coefficients  $c_i$  and  $c_i^I$ . Their expressions are given in terms of the variables defined in Refs. [19] and [26] and two new variables  $\delta_v$ ,  $\delta_s$ . The numerical integrals have been recomputed. Some of them differ slightly from the results of [19].

$c_1 = f + e = 17.89$	$c_1^I = f^I - 2(l + m) = -11.53$
$c_2 = d_1 = 5.46$	$c_2^I = n = 0.73$
$c_3 = d_2 = -7.22$	$c_3^I = h - 2d^I - q = 0.33$
$c_4 = -c = -4.53$	$c_4^I = 0$
$c_5 = -d_1 = -5.46$	$c_5^I = -n = -0.73$
$c_6 = d_2 = -7.22$	$c_6^I = c_3^I = 0.33$
$c_7 = -v - \delta_v = 4.85$	$c_7^I = \delta_s = 0.27$
$c_8 = \delta_v = 2.07$	$c_8^I = -v^I + s - \delta_s + 3\delta_L = 10.46$
$c_9 = 4w = -4.84$	$c_9^I = 4w^I - 2l - 2m + s + 3\delta_R = -4.88$

## Appendix B. $\Delta B = 2$ operators

There are many parallels between the calculations of spectator effects in inclusive decays, which is the main subject of this paper, and that of the matrix elements of the  $\Delta B = 2$  operators which contribute, for example, to the important process of  $B^0-\bar{B}^0$  mixing. Indeed, we have recomputed the matrix elements of the  $\Delta B = 2$  operators and compared the results to those in Ref. [12] as a check on our procedures and programs. The calculation of the matching factors are also similar in the two cases, and we have exploited this fact as a check on our perturbative calculation. Since we disagree with one of the terms in the results of Ref. [19], we briefly discuss the evaluation of the matching factors in this appendix.

We consider a generic  $\Delta B = 2$  operator of the form  $\bar{b}\Gamma_i^A q \bar{b}\tilde{\Gamma}_i^A q$ . The evaluation of the various contributions to the  $D_{ij}$ 's for  $\Delta B = 2$  operators parallels that of the operator  $O_i$  for spectator effects in inclusive decays defined in Eq. (A.2); specifically, as explicitly marked in Table A.2, the coefficients corresponding to  $j = 4, 5, 7$  and 8 change sign, whilst the remaining coefficients are the same. Thus from Table A.2 for spectator effects we deduce that the  $D_{ij}$ 's for  $\Delta B = 2$  operators are as given in Table B.1. The  $x_i$ 's are defined in Eq. (A.3) and tabulated in Table A.3.

We are particularly interested in the operator whose matrix elements contains the non-perturbative QCD effects for  $B^0-\bar{B}^0$  mixing:

$$\bar{b}\Gamma_i^A q \bar{b}\tilde{\Gamma}_i^A q \equiv O_L = \bar{b}\gamma^\rho q_L \bar{b}\gamma_\rho q_L, \quad (\text{B.1})$$

where the label  $L$  denotes “left”. In this case the different operators  $O_j$  of Table B.1 reduce to three independent ones,

$$O_R = \bar{b}\gamma^\rho q_R \bar{b}\gamma_\rho q_R, \quad (\text{B.2})$$

$$O_N = 2\bar{b}q_L \bar{b}q_R + 2\bar{b}q_R \bar{b}q_L + \bar{b}\gamma^\rho q_L \bar{b}\gamma_\rho q_R + \bar{b}\gamma^\rho q_R \bar{b}\gamma_\rho q_L, \quad (\text{B.3})$$

as well as  $O_L$  itself. Step (i) of the matching (see Section 2) is now given by the relation

Table B.1  
Values of the contributions to the  $D_{ij}$ 's for  $\Delta B = 2$  operators

$j$ (category)	$D_{ij}^{\Delta B=2}$	$O_j^L$
1 (1, 2)	$\frac{1}{2} - x_1$	$C_F \bar{b}(\Gamma_i^A) q \bar{b}(\tilde{\Gamma}_i^A) q$
2 (2)	$1 - x_2$	$\bar{b}(t^a \Gamma_i^A t^a) q \bar{b}(\tilde{\Gamma}_i^A) q + \bar{b}(\Gamma_i^A) q \bar{b}(t^a \tilde{\Gamma}_i^A t^a) q$
3 (2)	$-x_3$	$\bar{b}(t^a \gamma^0 \Gamma_i^A \gamma^0 t^a) q \bar{b}(\tilde{\Gamma}_i^A) q + \bar{b}(\Gamma_i^A) q \bar{b}(t^a \gamma^0 \tilde{\Gamma}_i^A \gamma^0 t^a) q$
4 (3, odd)	$x_4$	$\bar{b}(t^a \Gamma_i^A) q \bar{b}(t^a \tilde{\Gamma}_i^A) q$
5 (4, odd)	$1 + x_5$	$\bar{b}(\Gamma_i^A t^a) q \bar{b}(t^a \tilde{\Gamma}_i^A) q + \bar{b}(t^a \Gamma_i^A) q \bar{b}(\tilde{\Gamma}_i^A t^a) q$
6 (4)	$-x_6$	$\bar{b}(\Gamma_i^A \gamma^0 t^a) q \bar{b}(t^a \gamma^0 \tilde{\Gamma}_i^A) q + \bar{b}(t^a \gamma^0 \Gamma_i^A) q \bar{b}(\tilde{\Gamma}_i^A \gamma^0 t^a) q$
7 (5, odd)	$-\Omega_1 + x_7$	$\bar{b}(\Gamma_i^A t^a) q \bar{b}(\tilde{\Gamma}_i^A t^a) q$
8 (5, odd)	$-\Omega_2 + x_8$	$\frac{1}{12} \bar{b}(\Gamma_i^A \sigma^{\mu\nu} t^a) q \bar{b}(\tilde{\Gamma}_i^A \sigma^{\mu\nu} t^a) q$
9 (5)	$-x_9$	$\frac{1}{4} \bar{b}(\Gamma_i^A \gamma^\mu t^a) q \bar{b}(\tilde{\Gamma}_i^A \gamma^\mu t^a) q$

$$O_L^C = \left(1 + \frac{\alpha_s}{4\pi} D_L\right) O_L^L + \frac{\alpha_s}{4\pi} D_N O_N^L + \frac{\alpha_s}{4\pi} D_R O_R^L, \quad (\text{B.4})$$

where

$$D_L = -22.4, \quad D_N = -13.8 \quad \text{and} \quad D_R = -3.2. \quad (\text{B.5})$$

For the continuum renormalization scheme we have used  $\overline{\text{MS}}$ . In  $4 - 2\epsilon$  dimensions

$$\bar{b} \gamma^\rho \sigma^{\mu\nu} q_L \bar{b} \gamma_\rho \sigma_{\mu\nu} q_L = (12 - 2\epsilon) \bar{b} \gamma^\rho q_L \bar{b} \gamma_\rho q_L, \quad (\text{B.6})$$

from which we derive that  $\Omega_1 + \Omega_2 = 5$  in this scheme. In considering the crossing relations between inclusive decays and mixing, we note that in the latter process, one of the  $\bar{b}$  fields destroys a quark (so that  $\bar{b} \gamma^0 = \bar{b}$ ) and the other creates an antiquark (so that  $\bar{b} \gamma^0 = -\bar{b}$ ).

The results for  $D_L$  and  $D_N$  in Eq. (B.5) agree with those in the literature [26], whereas that for  $D_R$  does not (a similar conclusion was reached independently by Giménez [27]). In Ref. [26] the quoted result is  $D_R = -5.4$ .<sup>5</sup>

<sup>5</sup> We believe that the reason is that in Eq. (B.16) of Ref. [19] there should be a correction:

$$(\dots) + \frac{g^2}{16\pi^2} \frac{4}{3} (\omega + \omega') \rightarrow (\dots) - \frac{g^2}{16\pi^2} \frac{4}{3} (\omega + \omega') O_R^{\text{latt}},$$

Eq. (B.26) should be replaced by

$$D_R^I = \frac{1}{3} [s + 4\omega^I - 2(l + m)]$$

and that in Table A.1

$$D_R^I = -0.38 \quad \text{for } r = 1.$$

These errors are carried forward into successive papers [26,12,13]. Moreover in the same paper, in the definition of  $v^I$  (Eq. (B.17)), the term  $\Delta_2$  at the denominator should be replaced by  $\Delta_2^2$ .

## Acknowledgements

We thank Vicente Giménez and Juan Reyes for detailed correspondence concerning the matching factors and for pointing out some errors in the earlier version of the paper. We also thank Carlotta Pittori, Giulia De Divitiis, Luigi Del Debbio and Jonathan Flynn for many helpful discussions. We gratefully acknowledge Hartmut Wittig and other colleagues from the UKQCD collaboration for enabling us to use the gauge configurations and quark propagators.

This work was supported by PPARC grants GR/L29927 and GR/L56329, and EPSRC grant GR/K41663.

## References

- [1] J. Chay, H. Georgi and B. Grinstein, Phys. Lett. B 247 (1990) 399.
- [2] I.I. Bigi, M.A. Shifman, N.G. Uraltsev and A.I. Vainstein, Phys. Rev. Lett. 71 (1993) 496.
- [3] A.V. Manohar and M.B. Wise, Phys. Rev. D 49 (1994) 1310.
- [4] I. Bigi, M. Shifman and N. Uralstev, Ann. Rev. Nucl. Part. Sci. 47 (1997) 591.
- [5] M. Neubert, Int. J. Mod. Phys. A 11 (1996) 4173.
- [6] I.I. Bigi, N.G. Uraltsev and A.I. Vainstein, Phys. Lett. B 293 (1992) 430; B 297 (1993) 477 (E);
- [7] T. Junk, presented at the 2nd Int. Conf. on *B*-Physics and *CP*-Violation, Honolulu, Hawaii, March 1997, quoted in Ref. [8]
- [8] M. Neubert, Invited talk at the Int. Europhysics Conf. on High-Energy Physics (HEP 97), Jerusalem, Israel, August 1997; hep-ph/9801269.
- [9] M. Neubert and C.T. Sachrajda, Nucl. Phys. B 483 (1997) 339.
- [10] M.A. Shifman, A.I. Vainshtein and V.I. Zakharov, Nucl. Phys. B 147 (1979) 385, 448.
- [11] B. Blok and M. Shifman, Proc. of the 3rd Workshop on the Tau-Charm Factory, Marbella, Spain, June 1993, ed. J. Kirby and R. Kirby (editions Frontières, 1994) (hep-ph/9311331);  
I.I. Bigi et al., in *B-Decays*, ed. S. Stone, 2nd edition (World Scientific, Singapore, 1994) p. 134;  
I.I. Bigi, hep-ph/9508408.
- [12] UKQCD Collaboration, A.K. Ewing et al., Phys. Rev. D 54 (1996) 3526.
- [13] V. Giménez and G. Martinelli, Phys. Lett. B 398 (1997) 135.
- [14] J. Christensen, T. Draper and C. McNeile, Phys. Rev. D 56 (1997) 6993.
- [15] B. Sheikholeslami and R. Wohlert, Nucl. Phys. B 259 (1985) 572.
- [16] M.A. Shifman and M.B. Voloshin, Sov. J. Nucl. Phys. 41 (1985) 120; JETP 64 (1996) 698.
- [17] M.A. Shifman and M.B. Voloshin, Sov. J. Nucl. Phys. 45 (1987) 292.
- [18] H.D. Politzer and M.B. Wise, Phys. Lett. B 206 (1988) 681; B 208 (1988) 504.
- [19] A. Borelli and C. Pittori, Nucl. Phys. B 385 (1992) 502.
- [20] UKQCD Collaboration, C.R. Allton et al., Phys. Rev. D 47 (1993) 5128.
- [21] UKQCD Collaboration, R.M. Baxter et al., Phys. Rev. D 49 (1994) 1594.
- [22] H. Cheng and K. Yang, hep-ph/9805222.
- [23] G. Altarelli and S. Petrarca, Phys. Lett. B 261, (1991) 303.
- [24] I. Bigi, B. Block, M.A. Shifman and A. Vainshtein, Phys. Lett. B 323 (1994) 408.
- [25] B. Guberina, S. Nussinov, R. Peccei and R. Rückl, Phys. Lett. B 89 (1979) 111;  
N. Bilic, B. Guberina and J. Trampetic, Nucl. Phys. B 248 (1984) 261;  
B. Guberina, R. Rückl and J. Trampetic, Z. Phys. C 33 (1986) 297.
- [26] A. Borelli et al., Nucl. Phys. B 409 (1993) 382.
- [27] V. Giménez, Private Communication.

## Towards a lattice determination of the $B^*B\pi$ coupling

This content has been downloaded from IOPscience. Please scroll down to see the full text.

JHEP10(1998)010

(<http://iopscience.iop.org/1126-6708/1998/10/010>)

View the [table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 140.192.180.252

This content was downloaded on 16/09/2014 at 15:07

Please note that [terms and conditions apply](#).

# Towards a lattice determination of the $B^*B\pi$ coupling

---

**UKQCD Collaboration****Giulia M. de Divitiis, Luigi Del Debbio, Massimo Di Pierro  
and Jonathan M. Flynn**

*Dept. of Physics and Astronomy, Univ. of Southampton,  
Southampton SO17 1BJ, UK  
E-mail: giulia@hep.phys.soton.ac.uk, ldd@hep.phys.soton.ac.uk,  
mdp@hep.phys.soton.ac.uk, j.flynn@hep.phys.soton.ac.uk*

**Chris Michael and Janne Peisa\***

*DAMTP, Univ. of Liverpool, Liverpool L69 3BX, UK  
E-mail: cmi@liv.ac.uk, J.J.Peisa@swansea.ac.uk*

**ABSTRACT:** The coupling  $g_{B^*B\pi}$  is related to the form factor at zero momentum of the axial current between  $B^*$ - and  $B$ -states. This form factor is evaluated on the lattice using static heavy quarks and light quark propagators determined by a stochastic inversion of the fermionic bilinear. The  $g_{B^*B\pi}$  coupling is related to the coupling  $g$  between heavy mesons and low-momentum pions in the effective heavy meson chiral lagrangian. The coupling of the effective theory can therefore be computed by numerical simulations. We find the value  $g = 0.42(4)(8)$ . Besides its theoretical interest, the phenomenological implications of such a determination are discussed.

**KEYWORDS:** Lattice Gauge Field Theories, B-Physics.

---

\*Present address: Dept. of Physics, Univ. of Wales Swansea, Singleton Park, Swansea SA2 8PP, UK.

---

## Contents

<b>1. Introduction</b>	<b>1</b>
<b>2. Pion reduction</b>	<b>5</b>
<b>3. Lattice results</b>	<b>6</b>
3.1. Stochastic propagators	6
3.2. Lattice computation	8
<b>4. Renormalisation constants</b>	<b>14</b>
<b>5. Phenomenological implications</b>	<b>17</b>
<b>6. Conclusions</b>	<b>20</b>

---

## 1. Introduction

The precise determination of the Cabibbo-Kobayashi-Maskawa (CKM) matrix will provide a stringent consistency test of the Standard Model (SM), together with new handles to understand CP violation and search for hints of new physics. As far as the heavy flavour sector is concerned, a large amount of experimental data is expected from  $B$ -factories and CLEO in the near future. Non-perturbative QCD effects are the main source of systematic error in the extraction of fundamental parameters from experimental data: reliable results depend on some way to tame the large-distance dynamics.

Lattice simulations provide a powerful tool to investigate non-perturbative dynamics from first principles, but systematic errors are introduced by the finite lattice spacing and the restricted range of quark masses which may be simulated. A different approach is based on exploiting exact or approximate symmetries of the theory to write effective lagrangians describing the large distance behaviour in terms of effective meson fields. The chiral symmetry of strong interactions in the limit where  $m_q \rightarrow 0$  constrains the terms appearing in the chiral lagrangian. The couplings in this lagrangian are phenomenological inputs, which need to be taken either from experimental data or from some other source. On the other hand, when the quark masses tend towards infinity, heavy quark effective theory (HQET) has proved to be a powerful tool to study heavy-flavour physics. However, HQET is less powerful when applied to heavy-to-light transitions, like  $B \rightarrow \pi$ , where the normalisation of matrix elements is not

fixed by the symmetry. A combination of these two symmetries has been used in recent years to develop the heavy meson chiral lagrangian ( $\text{HM}\chi$ ), describing the interactions of low-momentum pions with mesons containing a single heavy quark [1] (for reviews see [2, 3]).

The definitions and notations used throughout this paper are conveniently introduced by a succint description of the building blocks of this effective theory.

For the heavy degrees of freedom, heavy quark symmetry predicts that the wave function of a heavy meson is independent of the flavour and spin of the heavy quark, leading to a covariant representation of heavy mesonic states [4]. The angular momentum  $j$  and the parity  $P$  of the light degrees of freedom determine a degenerate doublet of states with spin-parity  $J^P = (j \pm 1/2)^P$ . The pseudoscalar and vector mesons (e.g.  $B$  and  $B^*$ ) correspond to the  $j = 1/2$  case and are described by a  $4 \times 4$  Dirac matrix  $H$  with two spinor indices, one for the heavy quark and one for the light degrees of freedom. In terms of the effective meson fields:

$$H = \frac{1 + \not{v}}{2} [B_\mu^* \gamma^\mu - B \gamma_5], \quad \overline{H} = \gamma^0 H^\dagger \gamma^0, \quad (1.1)$$

where  $v$  is the velocity of the meson and  $B$  and  $B^*$  are the annihilation operators for particles containing a  $b$  quark in the initial state. These meson fields are used in the effective lagrangian to describe the heavy mesons. The light mesons are treated as an octet of pseudo-Goldstone bosons according to the usual chiral lagrangian formalism. At low-momentum the strong interactions of  $B$  and  $B^*$  mesons with light pseudoscalars are described by the couplings in the effective lagrangian; the lowest order interaction is given by [1, 3]

$$\mathcal{L}_{\text{HM}\chi}^{\text{int}} = g \text{Tr}(\overline{H}_a H_b \mathcal{A}_\mu^{ba} \gamma^\mu \gamma^5), \quad (1.2)$$

where

$$\mathcal{A}_\mu = \frac{i}{2} (\xi^\dagger \partial^\mu \xi - \xi \partial^\mu \xi^\dagger) \quad (1.3)$$

with  $\xi = \exp(i\mathcal{M}/f)$ .  $\mathcal{M}$  is a  $3 \times 3$  matrix of  $\pi$ ,  $\eta$  and  $K$  fields

$$\mathcal{M} = \begin{pmatrix} \pi^0/\sqrt{2} + \eta/\sqrt{6} & \pi^+ & K^+ \\ \pi^- & -\pi^0/\sqrt{2} + \eta/\sqrt{6} & K^0 \\ K^- & \bar{K}^0 & -\sqrt{2/3}\eta \end{pmatrix} \quad (1.4)$$

and the trace is over Dirac indices. At tree level  $f$  can be set equal to  $f_\pi$  (the definition of the decay constant used here sets  $f_\pi = 132$  MeV). The roman indices  $a$  and  $b$  denote light quark flavour and repeated indices are summed over 1, 2, 3. The expansion of  $\mathcal{A}$  in terms of pion fields begins with a linear term,

$$\mathcal{A}_\mu = -\frac{1}{f} \partial_\mu \mathcal{M} + \dots \quad (1.5)$$

The coupling  $g$  in eq. (1.2) can easily be related to the  $B^* B \pi$  coupling defined as [5, 6]

$$\langle B^0(p) \pi^+(q) | B^{*+}(p') \rangle = -g_{B^* B \pi}(q^2) q_\mu \eta^\mu(p') (2\pi)^4 \delta(p' - p - q), \quad (1.6)$$

where  $\eta^\mu$  is the polarisation vector of the  $B^*$  and the physical states are relativistically normalised:

$$\langle B(p)|B(p')\rangle = 2p^0(2\pi)^3\delta^{(3)}(\mathbf{p} - \mathbf{p}') . \quad (1.7)$$

The physical coupling  $g_{B^*B\pi}$  is given by the value of the above form factor for an on-shell pion:

$$g_{B^*B\pi} = \lim_{q^2 \rightarrow m_\pi^2} g_{B^*B\pi}(q^2) . \quad (1.8)$$

At tree level in the heavy meson chiral lagrangian, the above matrix element is

$$\langle B^0(p)\pi^+(q)|B^{*+}(p')\rangle = -\frac{2m_B}{f_\pi} g q_\mu \eta^\mu(p') (2\pi)^4 \delta(p' - p - q) , \quad (1.9)$$

which therefore yields

$$g_{B^*B\pi} = \frac{2m_B}{f_\pi} g . \quad (1.10)$$

As a result,  $g$  and  $g_{B^*B\pi}$  are considered as equivalent throughout this paper. The above relation can be extended to take into account higher-order terms in the HM $\chi$  lagrangian [7, 8, 9].

Starting from eq. (1.6) and performing an LSZ reduction of the pion field, the coupling  $g$  is related to the form factor at zero momentum-transfer of the axial current between hadronic states. Such a relation is the analog, in the  $B\pi$  system, of the Goldberger-Treiman relation, relating the nucleon electromagnetic form factor to the nucleon-nucleon-pion coupling. An important consequence of the Goldberger-Treiman relation, for our purposes, is that it allows a numerical evaluation of the  $g_{B^*B\pi}$  coupling, as the form factors of the axial current can be evaluated by a lattice simulation. The details of the pion reduction are presented in sect. 2.

The interest of such a computation is twofold. From a theoretical point-of-view, it is interesting *per se* to be able to fix, from lattice QCD, the coupling appearing in the heavy meson chiral lagrangian. On the other hand, it is important to stress that this determination also has phenomenological motivations. Assuming vector dominance in the  $B \rightarrow \pi l\nu$  decay, the coupling  $g_{B^*B\pi}$  fixes the normalisation of the form factors used to parametrise the matrix element of the weak vector current,  $V^\mu = \bar{u}\gamma^\mu b$ , between hadronic states. Defining the form factors by

$$\begin{aligned} \langle \pi^+(p')|V^\mu|\bar{B}(p)\rangle &= f_1(q^2)(p + p' - \frac{m_B^2 - m_\pi^2}{q^2}q)^\mu + f_0(q^2)\frac{m_B^2 - m_\pi^2}{q^2}q^\mu = \\ &= f_+(q^2)(p + p')^\mu + f_-(q^2)q^\mu , \end{aligned} \quad (1.11)$$

where  $q = p - p'$  is the transferred momentum, the contribution from the  $B^*$  channel is easily evaluated:

$$f_1(q^2) = \frac{g_{B^*B\pi}}{2f_{B^*}} \frac{1}{1 - q^2/m_{B^*}^2} . \quad (1.12)$$

The normalisation of the form factor therefore depends on the  $B^*B\pi$  coupling and the decay constant of the vector meson, defined as

$$\langle 0 | V^\mu | \bar{B}^*(p) \rangle = i \frac{m_{B^*}^2}{f_{B^*}} \eta^\mu(p). \quad (1.13)$$

Heavy quark symmetry and chiral symmetry justify this pole form for  $f_1$  when  $q^2$  is close to  $q_{\max}^2 = (m_B - m_\pi)^2$  [1, 7, 8, 9, 10, 11]. For  $q^2$  far from  $q_{\max}^2$ , the pole form may be taken as a phenomenological ansatz. However, we note that the functional dependence of the form factor in eq. (1.12) cannot be simultaneously consistent with heavy quark symmetry at large  $q^2$ , which demands that  $f_1(q_{\max}^2) \sim m_B^{1/2}$  and the light-cone sum rule scaling relation at  $q^2 = 0$ , which states  $f_1(q^2=0) \sim m_B^{-3/2}$  [12].<sup>1</sup> Nonetheless, by fitting lattice results, which are available in the high  $q^2$  region where the pole form is justified, we can determine the parameters in eq. (1.12).

It is interesting to remark that the interplay of the effective lagrangian approach and lattice simulations provides another determination of the form factors for the heavy-to-light  $B$  decays and therefore sheds further light on the theoretical determination of the non-perturbative effects mentioned at the beginning. This result can be compared with direct computations of the same quantities obtained by fitting lattice data [14], using unitarity bounds [15] and sum rules [6, 16].

In the work described here, the matrix element of the light quark axial current between the heavy mesons is computed in a quenched lattice simulation of QCD in the static heavy quark limit, using stochastic methods to compute the desired light quark propagators, as described in sect. 3. The discussion of systematic errors is an important issue in any lattice calculation and plays a crucial part in estimating the error on the final result. In this respect, it is important to stress here that we are presenting an exploratory study. Our main concern is therefore to test the possibility of extracting the coupling defined above, rather than presenting its best lattice determination. Such a task would require a more extensive simulation and is left for further studies.

Renormalisation constants are needed in order to connect lattice results with continuum physical observables. Those relevant for the action and the quantities considered in this paper are summarised in sect. 4.

The best estimate we obtain for  $g$  is

$$g = 0.42(4)(8). \quad (1.14)$$

The phenomenological implications of this result are discussed in sect 5.

---

<sup>1</sup>Light-cone sum rules have also been applied at large  $q^2$  and reproduce the  $m_B^{1/2}$  scaling behaviour of  $f_1$  as demanded by heavy quark symmetry [13].

## 2. Pion reduction

An LSZ reduction of the pion in the definition of  $g_{B^*B\pi}$ , eq. (1.6), yields

$$\langle B^0(p)\pi^+(q)|B^{*+}(p+q)\rangle = i(m_\pi^2 - q^2) \int_x e^{iq\cdot x} \langle \bar{B}(p)|\pi(x)|B^*(p+q)\rangle. \quad (2.1)$$

Using the PCAC relation

$$\pi(x) = \frac{1}{m_\pi^2 f_\pi} \partial^\mu A_\mu(x), \quad (2.2)$$

where  $A_\mu$  is, as usual, the QCD axial current, eq. (2.1) becomes

$$\langle B^0(p)\pi^+(q)|B^{*+}(p+q)\rangle = q^\mu \frac{1}{f_\pi} \frac{m_\pi^2 - q^2}{m_\pi^2} \int_x e^{iq\cdot x} \langle \bar{B}(p)|A_\mu(x)|B^*(p+q)\rangle. \quad (2.3)$$

The matrix element of the axial current is parametrised in terms of three form factors

$$\begin{aligned} \langle B^0(p)|A_\mu(0)|B^{*+}(p+q)\rangle &= \eta_\mu F_1(q^2) + (\eta \cdot q)(2p+q)_\mu F_2(q^2) + \\ &\quad + (\eta \cdot q) q_\mu F_3(q^2) \end{aligned} \quad (2.4)$$

yielding for the  $B^*B\pi$  coupling

$$g_{B^*B\pi}(q^2) = -\frac{1}{f_\pi} \frac{m_\pi^2 - q^2}{m_\pi^2} \left[ F_1(q^2) + (m_{B^*}^2 - m_B^2) F_2(q^2) + q^2 F_3(q^2) \right]. \quad (2.5)$$

In the static limit in which our simulation is performed, the  $B$  and  $B^*$  mesons are degenerate, so that the form factor  $F_2$  can be discarded.

Analytical continuation of eq. (2.5) towards the soft-pion limit ( $q^2 \rightarrow 0$ ), leads to

$$g_{B^*B\pi}(0) = -\frac{1}{f_\pi} F_1(0). \quad (2.6)$$

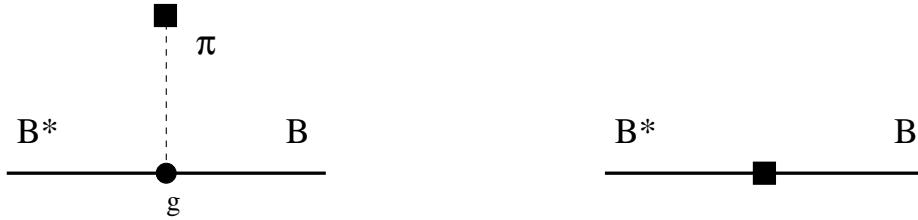
It is commonly assumed, when deriving the Goldberger-Treiman relation, that  $g_{B^*B\pi}$  is a smooth function of  $q^2$ , and, therefore, that the physical coupling can be approximated by

$$g_{B^*B\pi} = g_{B^*B\pi}(m_\pi^2) \approx g_{B^*B\pi}(0). \quad (2.7)$$

The above equation explicitly shows that, in the soft-pion limit, the  $B^*B\pi$  coupling is related to the form factor of the axial current between  $B$  and  $B^*$  states. If one were working in the chiral limit from the very beginning, the same Goldberger-Treiman relation, eq. (2.6), would be obtained from the conservation of the axial current.

The relation between  $g_{B^*B\pi}$  and  $g$  mentioned earlier can be rederived by comparing the matrix element of the Noether current associated to chiral symmetry both in HM $\chi$  and QCD. In the chiral limit, the Noether currents associated with chiral symmetry, in QCD and in HM $\chi$ , are respectively

$$j_{\text{QCD } \mu}^{5 ab} = \bar{q}^a \gamma_\mu \gamma_5 q^b \quad (2.8)$$



**Figure 1:** Tree-level diagrams needed to compute the matrix element of the axial current in HM $\chi$ . The dot ( $\bullet$ ) represents the  $g$ -vertex in the HM $\chi$  lagrangian; the squares are insertions of the axial current in eq. (2.9).

and

$$j_{\text{HM}\chi\mu}^{5ab} = f_\pi \partial_\mu \mathcal{M}^{ab} - 2g \left( B_\mu^{*a\dagger} B^b + B^a\dagger B_\mu^{*b} \right) + \dots, \quad (2.9)$$

where the ellipsis denotes terms with more than one pion or terms containing both heavy mesons and pions.

The form factors of the axial current in QCD are related to the coupling of the heavy meson chiral lagrangian by matching the two theories at tree level. The diagrams needed at tree level to evaluate the matrix element in eq. (2.4) for the HM $\chi$  current are depicted in fig. 1. A straightforward computation leads to

$$\begin{aligned} F_1(q^2) &= -2g m_B, \\ F_3(q^2) &= \frac{2g}{q^2} m_B, \end{aligned}$$

which, together with the Golberger-Treiman relation, reproduces eq. (1.10). One may also work away from the chiral and heavy quark limits and include corrections for finite mass pions and heavy quarks in this result.

The determination of  $g$  is therefore reduced to the computation of the matrix element of the light-light axial vector current between hadronic states. Such an evaluation can be performed using three-point correlation functions on the lattice. The details of the calculation are reported in the next section.

### 3. Lattice results

#### 3.1. Stochastic propagators

The numerical analysis is carried out on 20 quenched gauge configurations, generated on a  $12^3 \times 24$  lattice at  $\beta = 5.7$ , corresponding to  $a^{-1} = 1.10$  GeV. The heavy quark propagators are evaluated in the static limit [18]. Stochastic propagators [19, 20] are used to invert the fermionic matrix for the light quarks. They can be used in place of light quark propagators calculated with the usual deterministic algorithm. The stochastic inversion is based on the relation

$$S_{ij} = M_{ij}^{-1} = \frac{1}{Z} \int \mathcal{D}\phi (M_{jk}\phi_k)^* \phi_i \exp(-\phi_i^*(M^\dagger M)_{ij}\phi_j), \quad (3.1)$$

where, in our case,  $M$  is the tadpole improved SW fermionic operator and the indices  $i, j, k$  represent simultaneously the space-time coordinates, the spinor and colour indices. Two values of  $\kappa$  are considered,  $\kappa_1 = 0.13843$  and  $\kappa_2 = 0.14077$ , with  $c_{\text{sw}} = 1.57$ . The lighter value  $\kappa_2$  corresponds to a bare mass of the light quark around the strange mass. The chiral limit corresponds to  $\kappa_c = 0.14351$  [21]. For every gauge configuration, an ensemble of 24 independent fields  $\phi_i$  is generated with gaussian probability

$$P[\phi] = \frac{1}{Z} \exp \left( -\phi_i^* (M^\dagger M)_{ij} \phi_j \right). \quad (3.2)$$

All light propagators are computed as averages over the pseudo-fermionic samples:

$$S_{ij} = \begin{cases} \langle (M\phi)_j^* \phi_i \rangle \\ \text{or} \\ \langle (\gamma_5 \phi^*)_j (M\phi \gamma_5)_i \rangle \end{cases}, \quad (3.3)$$

where the two expressions are related by  $S = \gamma_5 S^\dagger \gamma_5$ . Moreover, the maximal variance reduction method is applied in order to minimise the statistical noise [19]. Maximal variance reduction involves dividing the lattice into two boxes ( $0 < t < T/2$  and  $T/2 < t < T$ ) and solving the equation of motion numerically within each box, keeping the spinor field  $\phi$  on the boundary fixed. According to the maximal reduction method, the fields which enter the correlation functions must be either the original fields  $\phi$  or solutions of the equation of motion in disconnected regions. The stochastic propagator is therefore defined from each point in one box to every point in the other box or on the boundary. For this reason, when computing a three-point correlation function

$$\langle 0 | J(t_1, x) \mathcal{O}(t_0, y) J^\dagger(t_2, z) | 0 \rangle, \quad (3.4)$$

one operator —  $\mathcal{O}$  in the present work — is forced to be on the boundary ( $t_0 = 0$  or  $T/2$ ) and the other two must be in different boxes, while the spatial coordinates are not constrained. If  $j$  is a point of the boundary, not all the terms in  $(M\phi)_j$  lie on the boundary because the operator  $M$  involves first neighbours in all directions. Hence, whenever a propagator  $S_{ij}$  is needed with one of the points on the boundary, we use whichever of the two expressions in eq. 3.3 has the spinor  $M\phi$  computed away from the boundary.

In smearing the hadronic interpolating operators, spatial fussed links are used. Following the prescription in [19, 22], to which the interested reader should refer for details, the fuzzed links are defined iteratively as

$$U_{\text{new}} = \mathcal{P} \left( f U_{\text{old}} + \sum_{i=1}^4 U_{\text{bend},i} \right), \quad (3.5)$$

where  $\mathcal{P}$  is a projector over  $SU(3)$  and  $U_{\text{bend},i}$  are the staples attached to the link in the spatial directions. We take  $f = 2.5$  and use two iterations with fuzzed links of length one. The smeared fermionic fields are defined following [22].

### 3.2. Lattice computation

In order to extract  $g$ , the three-point correlation function  $C_3$  and the two-point correlation function  $C_2$  for local ( $L$ ) and fussed ( $F$ ) sources are computed. The  $FF$  three-point function is defined as

$$C_{3\mu\nu}^{FF}(\mathbf{x}; t_1, t_2) = \frac{1}{V} \int d^3y \langle 0 | J_\nu^{B^*}(\mathbf{y}, -t_1) A^\mu(\mathbf{x} + \mathbf{y}, 0) J^B{}^\dagger(\mathbf{y}, t_2) | 0 \rangle, \quad (3.6)$$

where  $J_\nu^{B^*}$  and  $J^B$  are fussed operators with the quantum numbers corresponding respectively to the  $B^*$  and  $B$  states. Analogous definitions hold for the  $FL$  and  $LL$  cases. For time separations that are large enough to isolate the lowest energy states, the three-point function is related to the axial current matrix element:

$$\begin{aligned} C_{3\mu\nu}^{FF}(\mathbf{x}; t_1, t_2) &\rightarrow \langle 0 | J_\nu^{B^*} | B_r^* \rangle \frac{\langle B_r^* | A^\mu(\mathbf{x}) | B \rangle}{2m_B 2m_B} \langle B | J^B{}^\dagger | 0 \rangle e^{-M_B(t_1+t_2)} = \\ &= Z^F \eta_\nu^r(0) \frac{\langle B_r^* | A^\mu(\mathbf{x}) | B \rangle}{2m_B} Z^F e^{-M_B(t_1+t_2)}, \end{aligned} \quad (3.7)$$

where  $r$  is the polarisation label of the vector particle, and the sum over polarisations is omitted. In the static limit considered in this paper, the slope of the exponential time-decay is not the physical mass of the mesons; it can be interpreted as a binding energy. Moreover, the two-point functions for the vector and pseudoscalar particles are degenerate, leading to the same binding energies and  $Z$  factors for both the  $B$  and  $B^*$ . In order to avoid confusion, the physical mass is denoted by  $m_B$  and the binding energy by  $M_B$ .  $Z^F$  is the overlap of the interpolating operator with the physical state, defined from the two-point functions:

$$\begin{aligned} C_2^{FF}(t) &= \frac{1}{V} \int_V d^3y \langle 0 | J^B(\mathbf{y}, 0) J^B{}^\dagger(\mathbf{y}, t) | 0 \rangle \\ &\rightarrow (Z^F)^2 e^{-M_B t}. \end{aligned} \quad (3.8)$$

Integrating the three-point function over  $\mathbf{x}$  gives the matrix element for zero momentum transfer. In this limit, the latter can be expressed in terms of the form factor  $F_1(q^2)$  in eq. (2.4).

The sum over polarisations in eq. (3.7) yields

$$\int d^3x C_{3\mu\nu}^{FF}(\mathbf{x}; t_1, t_2) \rightarrow (Z^F)^2 (g_{\mu\nu} - \frac{p_\mu p_\nu}{m_B^2}) F_1(0) e^{-M_B(t_1+t_2)}. \quad (3.9)$$

The last equation shows that the three-point functions with  $\mu \neq \nu$  and  $\mu = \nu = 0$  should vanish. Therefore, only the correlators with  $\mu = \nu = 1, 2, 3$  are henceforth considered.

Moreover, taking rotational symmetry into account,  $C_3^{FF}(\mathbf{x}; t_1, t_2)$  is expected to be a function of the distance  $r$  only, up to cut-off effects. The three-point functions

measured on the lattice are averaged over equivalent  $\mathbf{x}$  positions.<sup>2</sup> The desired matrix element is obtained from the ratio

$$E_\mu(r, t) \stackrel{\text{def}}{=} (Z^F)^2 \overline{\sum}_r^x \frac{C_{3\mu\mu}^{FF}(\mathbf{x}, t, t)}{C_2^{FF}(t) C_2^{FF}(t)} \rightarrow \frac{\langle B^* | A_\mu(r) | B \rangle}{2m_B} \eta_\mu, \quad (3.10)$$

where the time-dependence cancels when the three-point function is divided by the product of two-point functions.

The coefficients  $Z^F$  and  $Z^L$  are extracted from the exponential fit of the two point correlation functions  $C_2$ . The data are reported in fig. 2. As one can see from the plots, the two-point functions already exhibit a single-exponential behaviour at moderate time separations. The main sources of error in this computation are expected to stem from the determination of the three-point function and from the value of the light quark masses, which are far from the chiral limit. Thus, a single exponential fit of the two-point functions turns out to be accurate enough for the scope of this study. The value of  $Z^F$  is extracted from a direct fit of  $C_2^{FF}$ , while  $Z^L$  is obtained from  $C_2^{FF}$  and  $C_2^{FL}$ . The results of the fit are shown directly on the plot. It is worth remarking that, in all the channels considered, single-exponential fits yield reasonable values for the reduced  $\chi^2$ .

The  $B$  meson decay constant, in the static approximation, can be extracted from:

$$f_B^{\text{static}} = Z_A^{\text{static}} \sqrt{\frac{2}{m_B}} Z^L a^{-3/2}, \quad (3.11)$$

where  $Z_A^{\text{static}}$  is the renormalisation constant for the axial current in the static theory, which is discussed in the following section and defined in eq. (4.11). The aim here is not a precise determination of the pseudoscalar decay constant,  $f_B^{\text{static}}$ . Rather, the result is presented to allow an estimate of the systematic errors in our computation of the  $\text{HM}\chi$  coupling,  $g$ .

The results of the fits, together with the values for  $f_B^{\text{static}}$  are summarised in tab. 1. The  $B$  meson binding energies obtained from the fits of  $C_2^{FF}$  and  $C_2^{FL}$  are consistent with each other. However, our determination appears to be slightly different from the

<sup>2</sup>The symbol

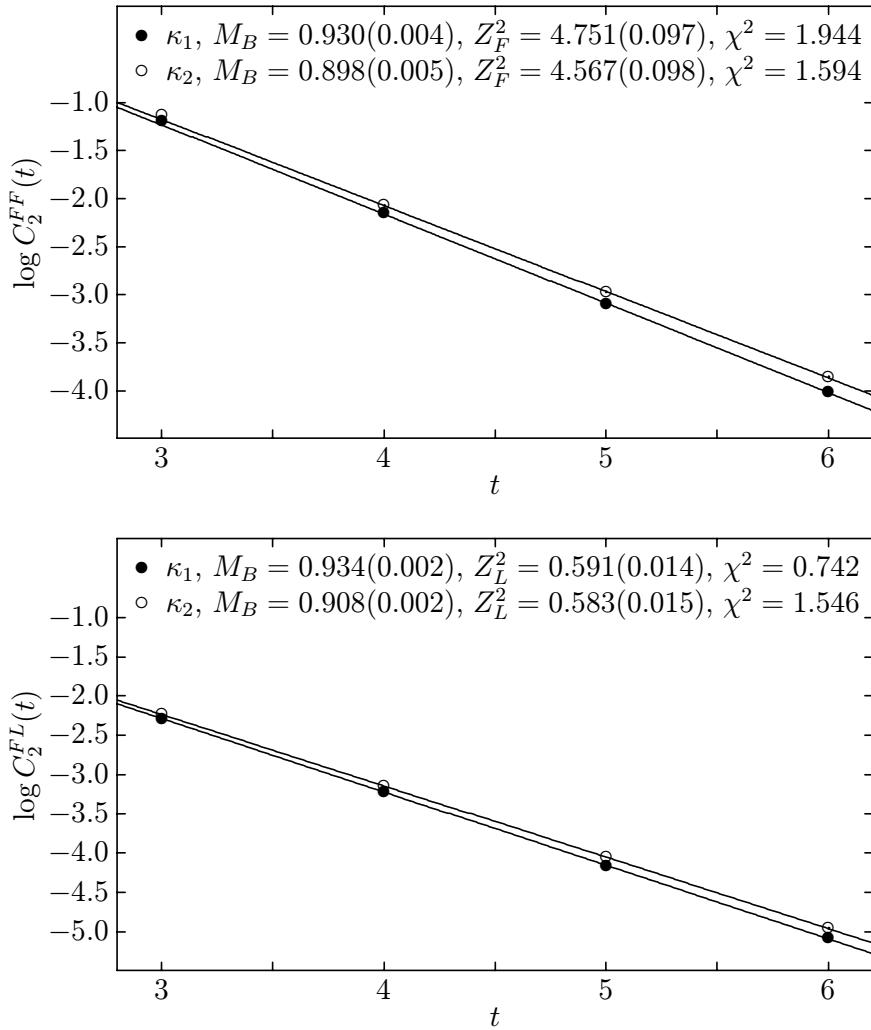
$$\overline{\sum}_r^x$$

indicates the average on all the spatial positions on the lattice compatible with the constraint  $|x| = r$ . On a finite lattice  $V = L^3$ , only some distances are allowed

$$r = \sqrt{x_1^2 + x_2^2 + x_3^2}$$

because each  $x_i$  must be an integer between 0 and  $L/2$ . For each allowed distance  $r$ , a given number of terms  $N_r$  appears in the above sum, yielding a relative error on each point  $\delta E_\mu(r, t) \sim N_r^{-1/2}$ , e.g.

$$N_0 = 1; N_1 = 6; N_{\sqrt{2}} = 12; N_{\sqrt{3}} = 6; \dots; N_{\sqrt{108}} = 1$$



**Figure 2:** Logarithmic plots of  $C_2^{FF}(t)$  and  $C_2^{FL}(t)$  for both values of  $\kappa$ . The quoted values refer to the reduced  $\chi^2$ .

one published in [19]. The discrepancy can be explained as a contribution from excited states, which is subtracted in [19] where a multi-exponential fit is performed. The 1-state fit yields a value of  $Z^L$  approximately 20% higher than the one obtained from the 3-state fit. Hence the value obtained for the static  $B$  decay constant lies above other lattice calculations of this quantity [17]. It is striking that the actual number does not depend on the value of the hopping parameter  $\kappa$ . However the variation, as the bare mass of the quark goes from  $m_s$  towards the chiral limit, is also expected to be about 20% and could be obscured by the contamination from excited states.

This is a first, exploratory, direct lattice determination of the coupling  $g$ , so the discrepancies noted above are perhaps expected and could easily arise from various lattice artefacts. Our value of  $\beta$  is far from the continuum limit, while our action and operators are not fully  $O(a)$  improved. We have not tried to optimise the smearing or

	$\kappa_1$	$\kappa_2$	$\kappa_c$
$M_B a$	0.930(4)	0.898(5)	0.862(7)
$(Z^F)^2$	4.75(10)	4.57(10)	4.37(15)
$(Z^L)^2$	0.59(1)	0.58(2)	0.57(3)
$f_B^{\text{static}}(\text{GeV})$	0.43(1)	0.43(1)	0.42(2)

**Table 1:** Values for  $Z^F$ ,  $Z^L$ ,  $M_B$  and  $f_B^{\text{static}}$  obtained from fitting the two-point functions.

fitting procedures. Our ensemble of gauge configurations and the collection of spinor configurations on each gauge sample are quite small. The calculation is also performed in the quenched approximation. All of these issues could be addressed in a further simulation, but here we will keep in mind that the uncertainties will propagate as systematic errors to our final result for the  $B^*B\pi$  coupling.

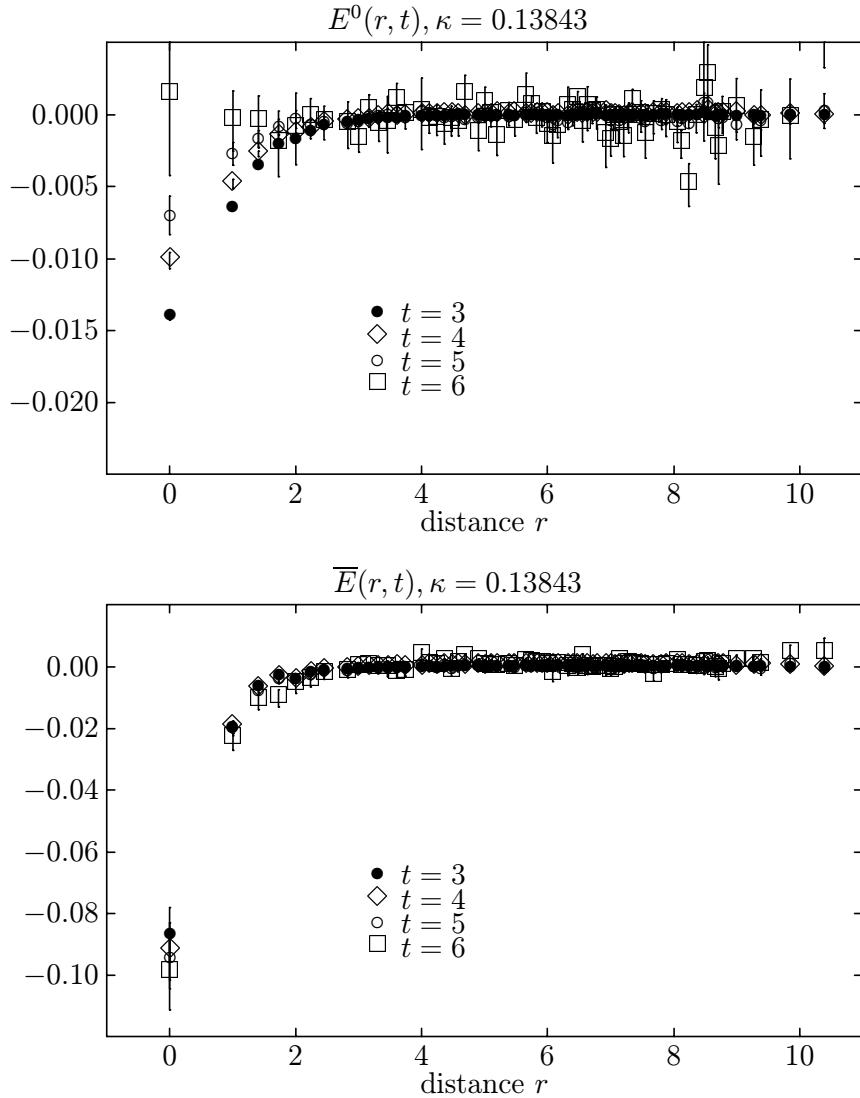
The generation of stochastic propagators for the light quark and the static approximation for the heavy quark have proved to be useful tools in this investigation. However, neither is strictly necessary to calculate the axial current matrix element of interest. One could combine static heavy quarks with light quark propagators determined by standard deterministic methods. A full  $O(a)$ -improved simulation with heavy quark masses around the charm mass would allow one to go beyond the static approximation and study the dependence of  $g$  on the heavy mass. The freedom allowed in lattice calculations to tune quark masses would also allow the light quark mass dependence, noted above as strikingly absent, to be investigated in more detail.

Returning to the results of the present simulation, the quantity  $E_0(x, t)$ , which is expected to vanish, is measured as a further consistency check. The data reported in figs. 3 and 4 show a much smaller signal than the one obtained for  $E_i(x, t)$ . As  $t$  is increased, the fictitious peak at zero distance decreases, while the noise increases.

Using rotational invariance, the average of the three spatial components of  $E_i(r, t)$

$$\overline{E}(r, t) = \frac{1}{3}(E_1(r, t) + E_2(r, t) + E_3(r, t)) \quad (3.12)$$

is measured. The results are reported in figs. 3 and 4, for the two values of  $\kappa$  used in this simulation. At fixed  $r$ ,  $\overline{E}(r, t)$  is expected to be independent of  $t$ . As this behaviour is confirmed by the data, the signal can be improved by averaging the values at different times, each weighted with its error, yielding a function of the spatial distance  $\overline{E}(r)$ , which needs to be integrated over the three-dimensional volume. Since this is an exploratory calculation we have neglected the effects of correlations between neighbouring time slices on the error in the average, which would increase the statistical error in our result. However, we note that the systematic errors considered below in any case dominate the uncertainty in the values at each time. The time-slices considered in the average are  $t = 4, 5, 6$ . Logarithmic plots of  $\overline{E}(r)$  are displayed in figs. 5 and 6 for both values of  $\kappa$ , suggesting that the data are consistent with an exponential decay.



**Figure 3:** Plots of  $E^0(r, t)$  and  $\bar{E}(r, t)$  as functions of  $r$  for different values of  $t$  ( $= 3, 4, 5, 6$ ) for  $\kappa = 0.13843$ .

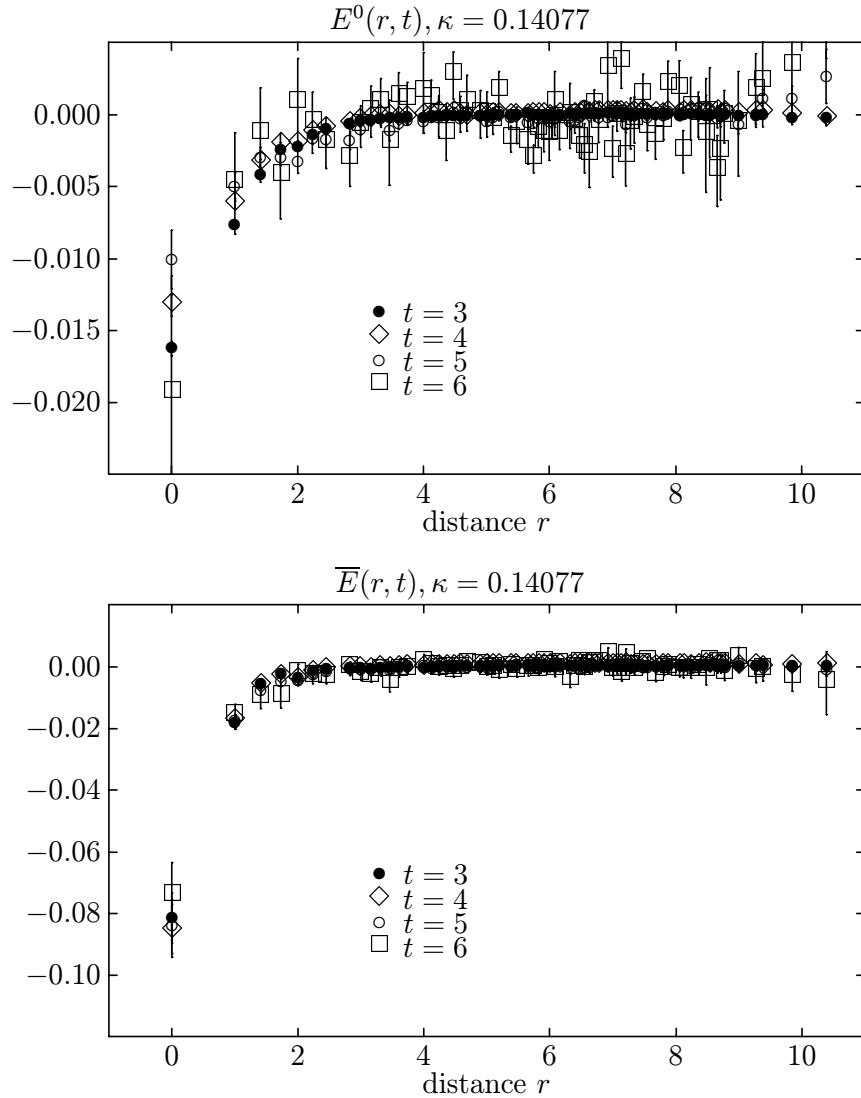
To evaluate the volume integral,  $\bar{E}(r)$  is fitted with a two-parameter exponential for each value of  $\kappa$ ,

$$f(r) = S e^{-r/r_0}. \quad (3.13)$$

The results of the fit and the values of the reduced  $\chi^2$  are recorded on the figures. The coupling  $g$  is finally obtained by integrating analytically the fitted function:

$$g^L = - \int_0^\infty 4\pi r^2 \bar{E}(r) dr = - \int_0^\infty 4\pi r^2 f(r) dr. \quad (3.14)$$

The superscript L indicates that these numbers are defined on the lattice using operators renormalised at the lattice energy scale  $a^{-1} = 1.10$  GeV. As for the two-point functions, the value of  $g^L$  does not appear to depend on the mass of quarks. It is not



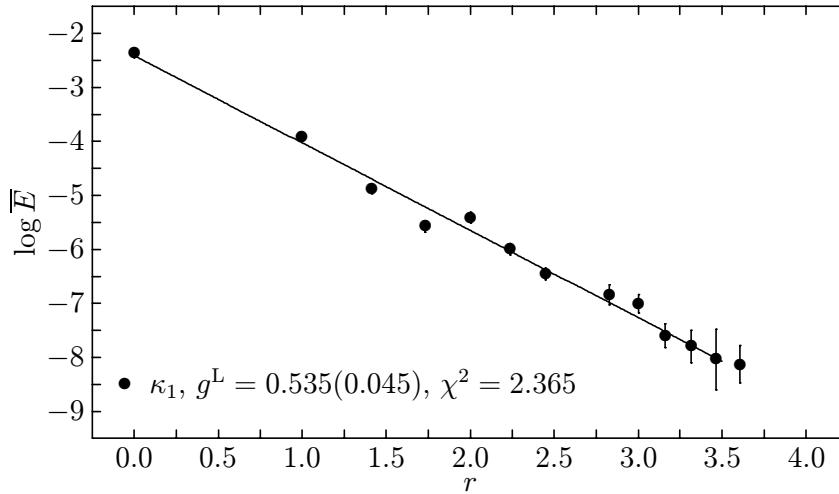
**Figure 4:** Plots of  $E^0(r, t)$  and  $\bar{E}(r, t)$  as functions of  $r$  for different values of  $t$  ( $= 3, 4, 5, 6$ ) for  $\kappa = 0.14077$ .

clear from this simulation, whether this is a genuine physical feature or, as explained earlier, an artefact of the lattice used here. Further studies should aim to clarify this point.

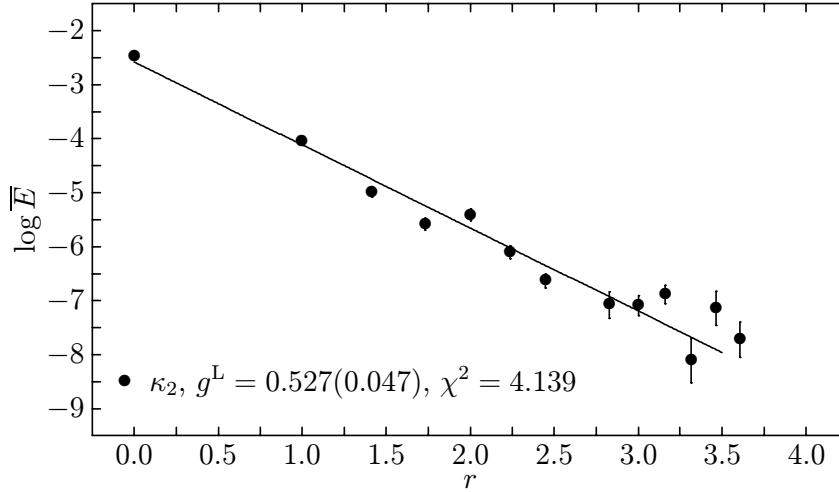
The best estimate for  $g^L$  at  $\kappa = \kappa_c$  is thus obtained from a weighted average of the results at the two kappas used in our simulations:

$$g^L = 0.52(5)(10). \quad (3.15)$$

The first error is statistical; the second is a conservative estimate of the systematic error of 20%, based on the systematic error observed in the estimate of  $f_B$ .



**Figure 5:** Plot of  $\overline{E}(r)$  as a function of  $r$  for  $\kappa = 0.13843$ .



**Figure 6:** Plot of  $\overline{E}(r)$  as a function of  $r$  for  $\kappa = 0.14077$ .

#### 4. Renormalisation constants

Quantities evaluated on the lattice are connected to their continuum counterparts by a finite renormalisation. In order to extract physical information, the matrix element of a bilinear quark operator defined on the lattice,  $\mathcal{O}^L(a)$ , has to be multiplied by the corresponding renormalisation constant

$$\mathcal{O}(\mu) = Z_{\mathcal{O}}(a\mu, g) \mathcal{O}^L(a). \quad (4.1)$$

which in general depends on the renormalisation scale  $\mu$  and the details of the lattice discretization (a single continuum operator may also match onto a set of lattice operators). For partially conserved currents, the  $\mu$  dependence disappears in the above

definition, the associated anomalous dimension being equal to zero. This is the case for the QCD light-light axial current considered here. It is also the case for the heavy-light current in full QCD, but not for the static-light current onto which it matches in the effective theory: hence the renormalisation constant  $Z_A^{\text{static}}$  in eq. (4.11) below, which converts the lattice matrix element to the physical  $f_B^{\text{static}}$ , includes the effect of running between different scales in the effective theory.

In our simulation, we use the standard gluon action and the tadpole-improved [23] Sheikholeslami-Wohlert [24] fermion action for the light quarks:

$$S = \frac{a^4}{2\kappa} \sum_{xy} \bar{\psi}(x) \left\{ \left[ 1 - \frac{i\tilde{\kappa}a}{2u_0^4} \sigma_{\mu\nu} F_{\mu\nu}(x) \right] \delta_{xy} - \frac{\tilde{\kappa}}{au_0} \left[ (1 - \gamma_\mu) U_\mu(x) \delta_{x+\hat{\mu},y} + (1 + \gamma_\mu) U_\mu^\dagger(y) \delta_{x-\hat{\mu},y} \right] \right\} \psi(y), \quad (4.2)$$

where as usual  $F_{\mu\nu}(x)$  is a lattice definition of the gluon field strength tensor,  $\tilde{\kappa} = \kappa u_0$  and we define  $u_0$  from the average plaquette in infinite volume,  $u_0^4 = \langle \frac{1}{3} \text{Tr } U_\square \rangle$ . The redefinition of the parameters in the action is done in order to absorb the large renormalisation coming from lattice tadpole graphs [23]. The standard definition of  $c_{\text{sw}}$ , the coefficient of the  $\sigma_{\mu\nu} F_{\mu\nu}$  term, in this case gives  $c_{\text{sw}} = 1/u_0^3$ . For our lattice,  $u_0 = 0.86081$  from the average plaquette, so that  $c_{\text{sw}} = 1.57$ . For the  $b$  quarks we use the standard static quark action. Here we summarise the required renormalisation constants for the light-light and static-light axial currents with the action and parameters defined above.

The matrix element of the light-light axial current between an initial state  $i$  and a final state  $f$  in the continuum can be written as:

$$\langle f | A_\mu | i \rangle = Z_A \langle f | A_\mu^L | i \rangle = Z_A^{1\text{-loop}} \frac{u_0}{u_0^{1\text{-loop}}} \langle f | A_\mu^L | i \rangle. \quad (4.3)$$

The factor  $u_0$ , a measure of the average link variable, can be interpreted as a non-perturbative rescaling of the quark fields in the tadpole improvement prescription. The 1-loop perturbative renormalisation constant must then be redefined by dividing out the 1-loop expression,  $u_0^{1\text{-loop}}$ , corresponding to the chosen definition of  $u_0$ ,

$$Z_A^{\text{tadpole}} = Z_A^{1\text{-loop}} / u_0^{1\text{-loop}}. \quad (4.4)$$

The overall renormalisation constant is given by the whole factor

$$Z_A = u_0 Z_A^{\text{tadpole}}. \quad (4.5)$$

The perturbative part reads

$$\frac{Z_A^{1\text{-loop}}}{u_0^{1\text{-loop}}} = \frac{1 + \frac{\alpha C_F}{4\pi} \zeta_A}{1 + \frac{\alpha C_F}{4\pi} \lambda} \simeq 1 + \frac{\alpha C_F}{4\pi} (\zeta_A - \lambda), \quad (4.6)$$

where  $\lambda = -\pi^2$  for our plaquette definition of  $u_0$  and  $\zeta_A = -13.8$  [26, 27, 28].

We note here that the removal of tree-level  $O(a)$  discretisation errors is achieved by combining the Sheikholeslami-Wohlert action for  $c_{\text{sw}} = 1$  with improved operators, found by redefining or “rotating” the quark fields [29]. In our simulation, the rotation has not been applied to the quarks, so the perturbative coefficient  $\zeta$  above is calculated using the improved action only. Moreover, in perturbation theory,  $c_{\text{sw}} = 1 + O(\alpha)$ , so we quote  $\zeta$  above calculated for  $c_{\text{sw}} = 1$ , without rotated light quark fields. In [30] the light fermion bilinear operator renormalisations are given as functions of  $c_{\text{sw}}$  and an arbitrary amount of field rotation: using our actual value of  $c_{\text{sw}} = 1.57$ , with no field rotation, in those results would increase  $Z_A$  by 5%.

The mean-field improved perturbative expansion is performed in terms of a boosted coupling constant,  $\tilde{\alpha}$ , which in our case is chosen as  $\tilde{\alpha} = \alpha_0/u_0^4$ , where  $\alpha_0$  is the bare lattice coupling constant. We use  $\tilde{\alpha}$  for  $\alpha$  in eq. (4.6). For our lattice, with  $\beta = 5.7$  and  $u_0 = 0.86081$  from the average plaquette, this leads to

$$Z_A = u_0 Z_A^{\text{tadpole}} = 0.806. \quad (4.7)$$

For the heavy-light axial current computed in the static limit, the renormalisation constant  $Z_A^{\text{static}}$  is found by a three step procedure combining the matching between full QCD and the continuum static theory at a scale of order  $m_b$  [31], the continuum static theory running from  $m_b$  to a lattice scale  $q^*$  [32, 33, 34] and finally the matching between the continuum static and the lattice static theories [35, 36]. Since we have performed a quenched lattice simulation we encounter the standard problem of matching a zero-flavour result to a continuum result with active flavours. We adopt a standard procedure of matching quenched to continuum results at  $q^*$  and then performing the continuum running with the appropriate number of flavours.<sup>3</sup> We extract from the calculation of Borrelli and Pittori [36] the appropriate matching factor corresponding to our case of an improved action, but without field rotations on the light quarks. In the notation of [36] the number we need is

$$C_F(5/4 - A_{\gamma_0\gamma_5} - d^I - f^I/2 + (e - e^{\text{red}})/2) = -19.21. \quad (4.8)$$

This appears in eq. (4.11) below and contrasts with the values  $-27.2$  for the Wilson action and  $-20.0$  for the improved action with  $c_{\text{sw}} = 1$  and field rotations included (We have recomputed the numerical integrals so these values differ in the last two figures from those quoted in [36]. The difference is, however, less than 1%).

In the matching to the lattice we will adopt a Lepage-Mackenzie choice [23] of the scale  $q^*$  at which to perform the matching. This scale is determined from the expectation value of  $\ln(qa)^2$  in the one-loop lattice perturbation theory integrals for the corrections to the renormalisation constant, including the perturbative tadpole

---

<sup>3</sup>One could decide to match the lattice to the continuum result at a different scale, then perform the continuum running with the appropriate number of flavours from that scale to  $m_b$ . However, for any choice of the intermediate scale between  $q^*$  and  $m_b$ , the difference in the final result is of the order of 2%, which is negligible compared to the systematic error due to the quenched approximation.

improvement corrections for the chosen definition of  $u_0$ . For the improved,  $c_{\text{sw}} = 1$ , action and plaquette definition of  $u_0$ , Giménez and Reyes [37] quote  $q^*a = 2.29$ . We will use this value.

We will also adopt a plaquette definition of the lattice coupling constant according to (for zero flavours):

$$-\ln(\langle \text{Tr } U_{\square} \rangle / 3) = \frac{4\pi}{3} \alpha_V (3.41/a)(1 - 1.19\alpha_V). \quad (4.9)$$

Once  $\alpha_V(3.41/a)$  is determined, we can use the equation for the running of  $\alpha_V$ ,

$$\alpha_V(q) = \frac{4\pi}{2b_0 \ln(q/\Lambda_V) + b_1 \ln(2 \ln(q/\Lambda_V))/b_0} \quad (4.10)$$

to determine  $\alpha_V(q^*)$ . In the quenched theory,  $b_0 = 11$  and  $b_1 = 102$ . We find  $a\Lambda_V = 0.294$  and  $\alpha_V(q^*) = 0.216$ .

Since  $q^* = 2.52$  GeV is between the charm and  $b$  quark thresholds we perform the continuum running with four active flavours. From the Particle Data Group (PDG) [38], we take  $\Lambda_{\overline{\text{MS}}}^{(5)} = 237_{-24}^{+26}$  MeV using two-loop running, and find  $\Lambda_{\overline{\text{MS}}}^{(4)}$  using continuity of the strong coupling at the  $b$  quark threshold. This threshold value is given by  $m_b$  satisfying  $m_b \overline{\text{MS}}(m_b) = m_b$ . We take  $m_b = 4.25$  GeV, using the average of the range, 4.1–4.4 GeV, quoted by the PDG [38].

The overall renormalisation constant is thus given by:

$$Z_A^{\text{static}} = \left( \frac{\alpha(m_b)}{\alpha(q^*)} \right)^d \left\{ 1 - c \frac{\alpha(m_b)}{4\pi} + \frac{\alpha(q^*) - \alpha(m_b)}{4\pi} J_A \right\} \times \\ \times \sqrt{u_0} \left( 1 + \frac{\alpha_V(q^*)}{4\pi} [4 \ln(q^*a) - 19.36 - C_F \lambda/2] \right), \quad (4.11)$$

where  $c = 8/3$ ,  $J_A = 0.91$  and  $d = -6/25$ . Using the inputs given above, we find

$$Z_A^{\text{static}} = 0.78. \quad (4.12)$$

Independent variations of  $\Lambda^{(5)}$  and  $m_b$  to the ends of the ranges quoted above and  $\pm 10\%$  variation in  $a^{-1}$  change this value by 1.3% or less. Changing  $q^*$  to  $1/a$  or  $\pi/a$  reduces  $Z_A^{\text{static}}$  by 13% or raises it by 1.3% respectively.

We close this section by noting that both  $Z_A$  and  $Z_A^{\text{static}}$  are evaluated in the chiral limit and so do not depend on the light quark mass used in the simulation.

## 5. Phenomenological implications

We now combine the lattice matrix element with its renormalisation factor to find the continuum value for  $g$ . Since we observe no light quark mass dependence in  $g^L$  and evaluated the renormalisation constant in the chiral limit, we multiply the lattice matrix element in eq. (3.15) by the renormalisation constant in eq. (4.7), to obtain:

$$g = Z_A g^L = 0.42(4)(8) \quad (5.1)$$

for the coupling of the heavy mesons with the Goldstone bosons.

Reference	$g$	Reference	$g$
[6]	0.32(2)	[41]	0.7
[42]	0.39	[5]	0.75-1.0
[3] <sup>a</sup>	0.44 (16)	[43, 44]	0.4-0.7
		[45]	0.38

<sup>a</sup>Combined sum rule + lattice results

**Table 2:** Recent determinations of the coupling constant  $g$

The direct decay  $B^* \rightarrow B\pi$  is kinematically forbidden. However, the corresponding reaction occurs in the  $D$  system, where the coupling  $g_{D^*D\pi}$  is also related to  $g$  by an expression analogous to eq. (1.10), although the  $1/m_Q$  corrections are expected to be larger in the charm case. A recent analysis [39], incorporating chiral symmetry breaking corrections plus  $1/m_c$  corrections in the  $\text{HM}\chi$  lagrangian and fitting to the branching ratios for  $D^{*0} \rightarrow D^0\pi^0$ ,  $D^{*+} \rightarrow D^+\pi^0$  and  $D_s^* \rightarrow D_s\pi^0$  (together with radiative decay rates for the same  $D^*$  mesons), gives

$$g = \begin{cases} 0.27(\frac{4}{2})(\frac{5}{2}) \\ 0.76(3)(\frac{2}{1}) \end{cases} . \quad (5.2)$$

The two fold ambiguity can be resolved by imposing the experimental limit  $\Gamma(D^{*+}) < 0.13 \text{ MeV}$  [40], which gives  $g < 0.52$  to a good approximation [39].

Other estimates of  $g$  are derived using constituent quark models and sum rules. Starting from the non-relativistic result  $g = 1$ , quark models can be improved using more sophisticated assumptions to describe the quark dynamics inside the meson. Independent estimates are obtained from computing QCD correlation functions in the framework of sum rules. To give an idea of the range spanned by different determinations, some recent results are listed in tab. 2, while our value together with the average of other determinations is displayed in fig. 7.

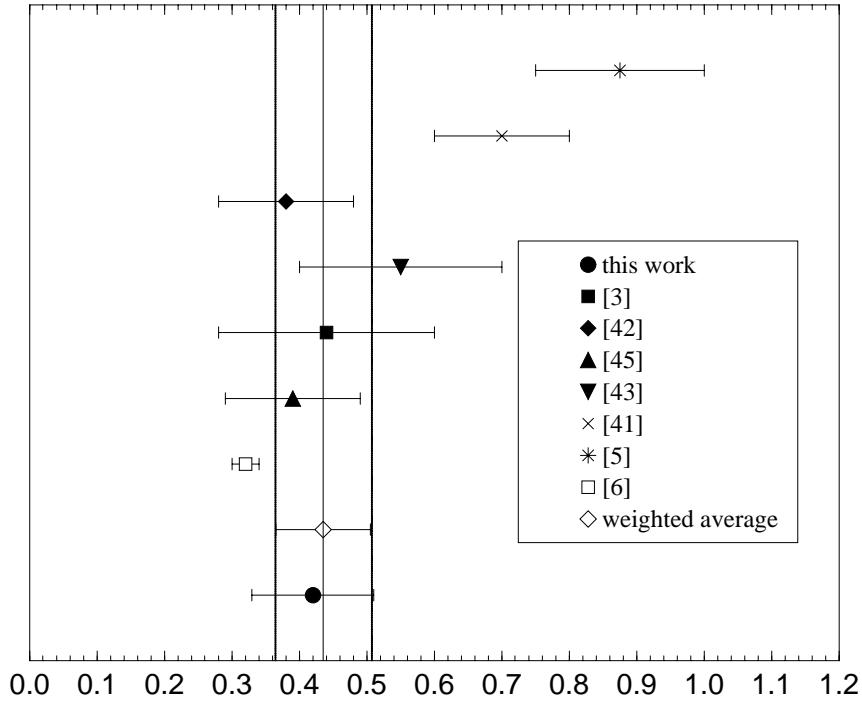
The best estimate from a global analysis of available results, quoted in the review [3], is

$$g \simeq 0.38 \quad (5.3)$$

with a total uncertainty of 20%; the latter being comparable with the estimated systematic error from our lattice computation. Not only is the result presented in this paper compatible with the previous estimates; the systematic error is also competitive when compared to the above results.

The agreement with other previous estimates is pleasing, but the value of the coupling  $g$  can also be used to check the consistency of the heavy quark symmetry predictions in the soft pion limit for the lattice results.

The form factor  $f_1$  of the semileptonic decay  $B \rightarrow \pi l\nu$  is predicted to have the expression given earlier in eq. (1.12). This behaviour for  $f_1(q^2)$  is expected to be valid near zero recoil ( $q^2 \rightarrow q_{\max}^2$ ), where the closest vector resonance dominates, even beyond



**Figure 7:** Comparison of various estimates of the coupling  $g$ .

the leading approximation in  $1/m_b$  in HQET [11]. One also obtains this behaviour from the heavy meson chiral lagrangian [1, 7, 8, 9], valid for a low momentum pion. It is perhaps surprising that such a behaviour is found by sum rules to hold reasonably, at least for the  $D$  meson, also at  $q^2 \rightarrow 0$  [6]. In this case the coupling  $g$  would fix the normalisation of the form factor  $f_1$ . The extension of the validity of the vector pole dominance for general values of  $q^2$  has no simple theoretical justification. It can be argued that the contributions from different resonances can lead to a single pole behaviour; however, in this case, the relation between  $g$  and  $f_1(0)$  would be spoilt.

Lattice data for the semi-leptonic  $B$  decay form factors have been fitted assuming a pole behaviour for  $f_1$  [14], yielding  $f_1(0) = 0.44(3)$ , to be compared with

$$f_1(0)|_{\text{VMD}} = \frac{m_B}{f_\pi f_{B^*}} g \quad (5.4)$$

from eq. (1.12). Our determination of  $g$  gives  $f_1(0) = 0.50(5)(10)$ , in good agreement with the direct fit. Such an agreement should not come as a surprise: the lattice data, after chiral extrapolation, turn out to be close to the end-point of the phase space kinematically allowed for  $B \rightarrow \pi$  decays ( $q^2 \simeq q_{max}^2$ ). The lattice normalisation of the form factor comes therefore from a fit in a region where vector dominance does hold. Hence, the above agreement should be seen as a consistency check of the two lattice computations. It is nevertheless important to see that the two results are not contradictory.

Assuming a pole form for  $f_1$ , an independent bound on the value of the residue is obtained by enforcing the theory to satisfy unitarity [46]. The bound quoted in [46] is

$$f_1(0)m_{B^*}^2 \leq 10 \text{ GeV}^2, \quad (5.5)$$

which translates into

$$f_1(0) \leq 0.4. \quad (5.6)$$

This, again, agrees reasonably with the determination obtained from VMD, using our value for  $g$ , and the one coming from the direct fit of the lattice form factors.

As the lattice determinations will improve in the future, the comparison of the three results summarised here could become an effective way to constrain the residue at the  $B^*$  pole.

## 6. Conclusions

We have shown that the coupling  $g_{B^*B\pi}$ , or equivalently the coupling  $g$  in the HM $\chi$  lagrangian, may be evaluated on the lattice from the matrix element of the light quark axial current between  $B$  and  $B^*$  states. The value of  $g$  enters many phenomenological quantities of interest calculated in heavy meson chiral perturbation theory, including the form factors for semileptonic  $B \rightarrow \pi$  decays mentioned here, as well as  $B^* \rightarrow \pi$  decays and other quantities such as ratios of heavy meson nonleptonic decay constants, heavy meson mass splittings and radiative decays.

Even the relatively crude estimate obtained here should be interesting for heavy meson phenomenology, but we believe future more precise lattice computations would be valuable.

## Acknowledgments

It is a pleasure to acknowledge enlightening discussions with C.T. Sachrajda. We would also like to thank L. Lellouch, R. Petronzio, G.C. Rossi and H. Wittig for discussing some of the issues presented here.

Numerical simulations were performed on a CRAY J90 computer at RAL, under grant GR/L55056. GdD is supported by a University of Southampton Annual Grant Award, LDD, JMF, GdD and MDP by PPARC under grants GR/L56329 and GR/L29927, and by EPSRC grant GR/K41663. JMF thanks the CERN Theory Division for hospitality during the completion of this work.

## References

- [1] M.B. Wise, *Phys. Rev. D* **45** (1992) 2188.
- [2] B. Grinstein, *An Introduction to Heavy Mesons*, 6th Mexican School of Particles and Fields, Villahermosa, Tabasco, October 1994, University of California San Diego preprint UCSD-PTH-95-05, [hep-ph/9508227](#).

- [3] R. Casalbuoni, A. Deandrea, N. Di Bartolomeo, R. Gatto, F. Feruglio and G. Nardulli, *Phys. Rep.* **281** (1997) 145 [[hep-ph/9605342](#)] and references therein.
- [4] M. Neubert, *Phys. Rep.* **245** (1994) 259 [[hep-ph/9306320](#)].
- [5] T.-M. Yan, H.-Y. Cheng, C.-Y. Cheung, G.-L. Lin, Y.C. Lin and H.-L. Yu, *Phys. Rev. D* **46** (1992) 1148.
- [6] V.M. Belyaev, V.M. Braun, A. Khodjamirian and R. Rückl, *Phys. Rev. D* **51** (1995) 6177 [[hep-ph/9410280](#)].
- [7] H.-Y. Cheng et al., *Phys. Rev. D* **49** (1994) 2490 [[hep-ph/9308283](#)].
- [8] R. Fleischer, *Phys. Lett. B* **303** (1993) 147.
- [9] A.F. Falk and B. Grinstein, *Nucl. Phys. B* **416** (1994) 771 [[hep-ph/9306310](#)].
- [10] C.G. Boyd and B. Grinstein, *Nucl. Phys. B* **442** (1995) 205 [[hep-ph/9402340](#)].
- [11] G. Burdman, Z. Ligeti, M. Neubert and Y. Nir, *Phys. Rev. D* **49** (1994) 2331 [[hep-ph/9309272](#)].
- [12] For a review, see: V.M. Braun, *Light-cone sum rules*, to appear in Proc. 4th Int. Workshop on Progress in Heavy Quark Physics, Rostock, Germany, September 1997, [hep-ph/9801222](#).
- [13] A. Khodjamirian, R. Rückl and C.W. Winhart, *Phys. Rev. D* **58** (1998) 054013 [[hep-ph/9802412](#)].
- [14] UKQCD Collaboration, L. Del Debbio, J.M. Flynn, L. Lellouch and J. Nieves, *Phys. Lett. B* **416** (1998) 392 [[hep-lat/9708008](#)].
- [15] L. Lellouch, *Nucl. Phys. B* **479** (1996) 353 [[hep-ph/9509358](#)].
- [16] P. Ball and V.M. Braun, *Phys. Rev. D* **55** (1997) 5561 [[hep-ph/9701238](#)];  
V.M. Belyaev, A. Khodjamirian and R. Rückl, *Z. Physik C* **60** (1993) 349 [[hep-ph/9305348](#)].
- [17] H. Wittig, *Int. J. Mod. Phys. A* **12** (1997) 4477 [[hep-lat/9705034](#)].
- [18] E. Eichten, *Nucl. Phys. 4 (Proc. Suppl.)* (1988) 147 [[hep-ph/9801258](#)];  
E. Eichten and B. Hill, *Phys. Lett. B* **232** (1989) 113.
- [19] UKQCD Collaboration, C. Michael and J. Peisa, University of Liverpool preprint LTH-420, *Phys. Rev. D* **58** (1998) 034506 [[hep-lat/9802015](#)].
- [20] G.M. de Divitiis, R. Frezzotti, M. Masetti and R. Petronzio, *Phys. Lett. B* **382** (1996) 393 [[hep-lat/9603020](#)].
- [21] UKQCD Collaboration, H. Shanahan et al., *Phys. Rev. D* **55** (1997) 1548 [[hep-lat/9608063](#)].

- [22] UKQCD Collaboration, P. Lacock, A. McKerrell, C. Michael, I.M. Stopher and P.W. Stephenson, *Phys. Rev.* **D 51** (1995) 6403 [[hep-lat/9412079](#)]
- [23] G.P. Lepage and P.B. Mackenzie, *Phys. Rev.* **D 48** (1993) 225.
- [24] B. Sheikholeslami and R. Wohlert, *Nucl. Phys.* **B 259** (1985) 572.
- [25] UKQCD Collaboration, A.K. Ewing et al., *Phys. Rev.* **D 54** (1996) 3526 [[hep-lat/9508030](#)].
- [26] C.T. Sachrajda, private communication.
- [27] M. Lüscher, S. Sint, R. Sommer and H. Wittig, *Nucl. Phys.* **B 491** (1997) 344 [[hep-lat/9611015](#)].
- [28] E. Gabrielli, G. Martinelli, C. Pittori, G. Heatlie and C.T. Sachrajda, *Nucl. Phys.* **B 362** (1991) 475.
- [29] G. Heatlie et al., *Nucl. Phys.* **B 352** (1991) 266.
- [30] M. Göckeler et al., Proceedings Lattice 96: 14th International Symposium on Lattice Field Theory, St. Louis, June 1996, *Nucl. Phys.* **53 (Proc. Suppl.)** (1997) 896 [[hep-lat/9608033](#)].
- [31] E. Eichten and B. Hill, *Phys. Lett.* **B 234** (1990) 511.
- [32] X. Ji and M.J. Musolf, *Phys. Lett.* **B 257** (1991) 409.
- [33] D.J. Broadhurst and A.G. Grozin, *Phys. Lett.* **B 267** (1991) 105; *ibid.* **274** (1991) 421.
- [34] V. Giménez, *Nucl. Phys.* **B 375** (1992) 582.
- [35] E. Eichten and B. Hill, *Phys. Lett.* **B 240** (1990) 193.
- [36] A. Borrelli and C. Pittori, *Nucl. Phys.* **B 385** (1992) 502.
- [37] V. Giménez and J. Reyes, IFIC Valencia preprint, FTUV 98/2, IFIC 98/2, [hep-lat/9806023](#).
- [38] Particle Data Group, C. Caso et al., *Euro. Phys. J.* **C 3** (1998) 1.
- [39] I.W. Stewart, *Nucl. Phys.* **B 529** (1998) 62 [[hep-ph/9803227 v2](#)].
- [40] ACCMOR Collaboration, S. Barlag et al., *Phys. Lett.* **B 278** (1992) 480.
- [41] S. Nussinov and W. Wetzel, *Phys. Rev.* **D 36** (1987) 130.
- [42] P. Colangelo, G. Nardulli, A. Deandrea, N. Di Bartolomeo, R. Gatto and F. Feruglio, *Phys. Lett.* **B 339** (1994) 151 [[hep-ph/9406295](#)].
- [43] P. Cho and H. Georgi, *Phys. Lett.* **B 296** (1992) 408 [[hep-ph/9209239](#)].
- [44] J.F. Amundson et al., *Phys. Lett.* **B 296** (1992) 415 [[hep-ph/9209241](#)].

- [45] P. Colangelo, F. De Fazio and G. Nardulli, *Phys. Lett.* **B 334** (1994) 175 [[hep-ph/9406320](#)].
- [46] C.G. Boyd, B. Grinstein and R.F. Lebed, *Phys. Rev. Lett.* **74** (1995) 4603 [[hep-ph/9412324](#)]. This paper uses a different convention to us for the vector decay constant,  $f_{B^*}$ , and their  $g_{B^*B\pi}$  is half as big as ours.

# An exploratory lattice study of spectator effects in inclusive decays of the $\Lambda_b$ baryon

UKQCD Collaboration

Massimo Di Pierro <sup>a</sup>, Chris T. Sachrajda <sup>a</sup>, Chris Michael <sup>b</sup>

<sup>a</sup> Department of Physics and Astronomy, University of Southampton, SO17 1BJ, UK

<sup>b</sup> Theoretical Physics Division, Dept. of Math. Sciences University of Liverpool, L69 3BX, UK

Received 2 July 1999; received in revised form 30 September 1999; accepted 6 October 1999

Editor: P.V. Landshoff

---

## Abstract

A possible explanation of the apparent discrepancy between the theoretical prediction and experimental measurement of the ratio of lifetimes  $\tau(\Lambda_b)/\tau(B_d)$  is that “spectator effects”, which appear at  $O(1/m_b^3)$  in the heavy quark expansion, contribute significantly. We investigate this possibility by computing the corresponding operator matrix elements in a lattice simulation. We find that spectator effects are indeed significant, but do not appear to be sufficiently large to account for the full discrepancy. We stress, however, that this is an exploratory study, and it is important to check our conclusions on a larger lattice and using a larger sample of gluon configurations. © 1999 Published by Elsevier Science B.V. All rights reserved.

PACS: 12.15.-y; 12.38.Gc; 12.39.Hg

---

## 1. Introduction

At leading order in the heavy-quark expansion the decay rate of the heavy quark is independent of its parent hadron. In this letter we present the results of an exploratory study, in which we attempt to gain some understanding of why the experimental result for the ratio of lifetimes [1]

$$\frac{\tau(\Lambda_b)}{\tau(B_d)} = 0.78 \pm 0.04 \quad (1)$$

differs so markedly from 1. In the heavy-quark expansion (see Refs. [2–7] and references therein) this ratio is equal to 1 at leading order, has no corrections of  $O(1/m_b)$  (where  $m_b$  is the mass of

the  $b$ -quark) [4,8] and the  $O(1/m_b^2)$  corrections can be evaluated to give [9],

$$\frac{\tau(\Lambda_b)}{\tau(B_d)} = 0.98 + O(1/m_b^3). \quad (2)$$

Here we present a lattice computation of the contributions to the  $O(1/m_b^3)$  term on the right-hand side of Eq. (2) which come from “spectator effects”, i.e. from decays in which two quark or antiquark constituents of the beauty hadron participate in the weak decay. These effects may be larger than estimates based purely on power counting would indicate as a result of the enhancement of the phase space for  $2 \rightarrow 2$  body reactions relative to  $1 \rightarrow 3$  body decays.

A comparison of our results with estimates obtained using other techniques is presented at the end of Section 3.

Denoting the  $O(1/m_b^3)$  contribution from spectator effects by  $\Delta$  and following the analysis of Ref. [9] we find that<sup>1</sup>

$$\Delta = -0.173\epsilon_1 + 0.195\epsilon_2 + 0.030L_1 - 0.252L_2, \quad (5)$$

where

$$\begin{aligned} \epsilon_1(m_b) &= \frac{8}{f_B^2 m_B} \frac{\langle B_q | \bar{b}\gamma^\mu Lt^a q \bar{q}\gamma^\mu Lt^a b | B_q \rangle}{2m_B} \\ &= -0.01 \pm 0.03, \end{aligned} \quad (6)$$

$$\begin{aligned} \epsilon_2(m_b) &= \frac{8}{f_B^2 m_B} \frac{\langle B_q | \bar{b}Lt^a q \bar{q}Rt^a b | B_q \rangle}{2m_B} \\ &= -0.02 \pm 0.02 \end{aligned} \quad (7)$$

have been computed, using lattice simulations, in Ref. [10]. The two new variables introduced here are

$$L_1(m_b) = \frac{8}{f_B^2 m_B} \frac{\langle \Lambda_b | \bar{b}\gamma^\mu Lq \bar{q}\gamma^\mu Lb | \Lambda_b \rangle}{2m_{\Lambda_b}}, \quad (8)$$

$$L_2(m_b) = \frac{8}{f_B^2 m_B} \frac{\langle \Lambda_b | \bar{b}\gamma^\mu Lt^a q \bar{q}\gamma^\mu Lt^a b | \Lambda_b \rangle}{2m_{\Lambda_b}}. \quad (9)$$

Heavy-quark symmetry implies that there are only two matrix elements which need to be considered for the  $\Lambda_b$ , in contrast to the four for  $B$ -mesons [9]<sup>2</sup>.

We stress that this is an exploratory study. It is the first calculation of the matrix elements  $L_{1,2}$  and provides a preliminary indication of whether spectator effects can reconcile the experimental measurements and theoretical predictions for the ratios of lifetimes in Eqs. (1) and (2). We perform the calculations with a static  $b$ -quark and two (rather large)

values of the mass of the light-quark and do not attempt to extrapolate the results to the chiral limit. Our results indicate that spectator effects are not negligible, although they do not appear to be sufficiently large to account fully for the discrepancy. Specifically we find:

$$L_1(m_b) = \begin{cases} -0.31(3) & \text{for } am_\pi = 0.52(3) \\ -0.22(4) & \text{for } am_\pi = 0.74(4), \end{cases} \quad (10)$$

$$L_2(m_b) = \begin{cases} 0.23(2) & \text{for } am_\pi = 0.52(3) \\ 0.17(2) & \text{for } am_\pi = 0.74(4), \end{cases} \quad (11)$$

with  $a^{-1} \approx 1.1$  GeV. The corresponding results for the ratio of lifetimes are

$$\frac{\tau(\Lambda_b)}{\tau(B_d)} = \begin{cases} 0.91(1) & \text{for } am_\pi = 0.52(3) \\ 0.93(1) & \text{for } am_\pi = 0.74(4). \end{cases} \quad (12)$$

## 2. Perturbative matching

In this section we briefly discuss the matching factors which are required to obtain the matrix elements of the continuum four-quark operators renormalised at a scale  $\mu$  from those of the bare lattice operators computed in lattice simulations at a cut-off  $a^{-1}$ . The details of the calculation are presented in Ref. [10]. Here we simply summarise the main points required for the evaluation of the matrix elements  $L_1$  and  $L_2$  in the  $\overline{\text{MS}}$  scheme.

We start by using the renormalisation group to relate the matrix elements  $L_1$  and  $L_2$ , defined in the  $\overline{\text{MS}}$  scheme at two different renormalisation scales,  $\mu = m_b$  and  $\mu = a^{-1}$ . Since the Wilson coefficient functions in the OPE expansion (5) have been evaluated at tree level only [9], we keep just the leading logarithms in the evolution equations so that [11,12]

$$\begin{aligned} \begin{pmatrix} L_1(m_b) \\ L_2(m_b) \end{pmatrix} &= \begin{pmatrix} 1 + \frac{2C_F\delta}{N_c} & -\frac{2\delta}{N_c} \\ -\frac{C_F\delta}{N_c^2} & 1 + \frac{\delta}{N_c^2} \end{pmatrix} \\ &\times \begin{pmatrix} L_1(a^{-1}) \\ L_2(a^{-1}) \end{pmatrix}, \end{aligned} \quad (13)$$

<sup>1</sup> Eq. (5) can be derived from Eq. (39) of Ref. [9] by the substitution

$$\tilde{B} = -6L_1, \quad (3)$$

$$r = -2\frac{L_2}{L_1} - \frac{1}{3}. \quad (4)$$

<sup>2</sup> The coefficients of two of the matrix elements for  $B$ -mesons are so small that we don't include them in Eq. (5).

where

$$\delta = \left( \frac{\alpha_s^{\overline{\text{MS}}}(a^{-1})}{\alpha_s^{\overline{\text{MS}}}(m_b)} \right)^{9/2\beta_0} - 1 = 0.40 \pm 0.04. \quad (14)$$

In estimating  $\delta$  we have used  $\Lambda_{\text{QCD}} = 250$  MeV,  $a^{-1} = 1.10$  GeV,  $m_b = 4.5$  GeV and  $\beta_0 = 9$ . The error in  $\delta$  is evaluated includes a 20% uncertainty for  $\Lambda_{\text{QCD}}$ .

In the second step of the matching we relate the matrix elements renormalised in the continuum to those regularized on lattice, both at the same scale,  $a^{-1}$ . Although this involves corrections of  $O(\alpha_s)$ , which are, in principle, beyond the precision which we require, we nevertheless include them because the perturbative coefficients in lattice perturbation theory are generally large. The computation for the most general four quark operator involving one heavy quark appears in the appendix of Ref. [10]. For  $L_1$  and  $L_2$  the relevant relations are:

$$L_1(a^{-1}) = \frac{8}{f_B^2 m_B} [h_{11} M_1 + h_{12} M_2 + h_{13} M_3 + h_{14} M_4], \quad (15)$$

$$L_2(a^{-1}) = \frac{8}{f_B^2 m_B} [h_{21} M_1 + h_{22} M_2 + h_{23} M_3 + h_{24} M_4], \quad (16)$$

where

$$M_1 = \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu Lq)(\bar{q}\gamma^\mu Lb) | \Lambda_b \rangle}{2m_{\Lambda_b}}, \quad (17)$$

$$M_2 = \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu L\gamma^0 q)(\bar{q}\gamma^\mu Lb) | \Lambda_b \rangle}{2m_{\Lambda_b}} + \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu Lq)(\bar{q}\gamma^0\gamma^\mu Lb) | \Lambda_b \rangle}{2m_{\Lambda_b}}, \quad (18)$$

$$M_3 = \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu Lt^a q)(\bar{q}\gamma^\mu Lt^a b) | \Lambda_b \rangle}{2m_{\Lambda_b}}, \quad (19)$$

$$M_4 = \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu L\gamma^0 t^a q)(\bar{q}\gamma^\mu Lt^a b) | \Lambda_b \rangle}{2m_{\Lambda_b}} + \frac{\langle \Lambda_b | (\bar{b}\gamma^\mu Lt^a q)(\bar{q}\gamma^0\gamma^\mu Lt^a b) | \Lambda_b \rangle}{2m_{\Lambda_b}} \quad (20)$$

are the matrix elements of the bare lattice operators regularised at  $a^{-1}$ . The coefficients  $h_{ij}$  are listed in Table 1 where for the lattice coupling constant we have used a boosted coupling equal to

$$\frac{\alpha_s^{\text{latt}}(a^{-1})}{4\pi} = \frac{6(8\kappa_{\text{crit}})^4}{(4\pi)^2 \beta} \simeq 0.01216. \quad (21)$$

Readers who prefer to use other choices of the lattice coupling constant can combine the coefficients in Table 1 with their choice of coupling.

A consequence of the Heavy Quark Effective Theory, and the fact that the light quarks in the  $\Lambda_b$  are in a spin zero combination, is that the number of lattice operators whose matrix elements have to be evaluated is four rather than 8 (which is the case for heavy mesons [10]).

In order to obtain the factor  $f_B^2 m_B$  it is also necessary to determine the normalization of the axial current,  $Z_A$ ,

$$f_B m_B = \langle 0 | A_0(a^{-1}) | B \rangle = Z_A \langle 0 | A_0^{\text{latt}}(a^{-1}) | B \rangle = \sqrt{2m_B} Z_A Z_L, \quad (22)$$

where  $A_0^{\text{latt}}$  is the time component of the axial current defined on the lattice and  $Z_L$  is obtained from the matrix element determined in numerical simulations At our value of the lattice spacing

$$Z_A = 1 - 20.0 \frac{\alpha_s^{\text{latt}}(a^{-1})}{4\pi} \simeq 0.75. \quad (23)$$

In the following section we combine the results for the matrix elements computed on the lattice ( $M_i$  and  $Z_L$ ) with the perturbative coefficients presented in this section ( $h_{ij}$  and  $Z_A$ ), to obtain the values of  $L_1(m_b)$  and  $L_2(m_b)$ .

### 3. Lattice computation and results

The non-perturbative strong interaction effects in spectator contributions to inclusive decays are contained in the matrix elements  $M_i$  in Eqs. (17)–(20). They have been evaluated in a quenched simulation on a  $12^3 \times 24$  lattice at  $\beta = 5.7$  using the SW improved action [13],

$$S^{\text{SW}} = S^{\text{gauge}} + S^{\text{Wilson}} - \frac{ic_{\text{SW}}}{2} \sum_{x,\mu,\nu} \bar{q}(x) F_{\mu\nu}(x) q(x), \quad (24)$$

where  $S^{\text{gauge}}$  and  $S^{\text{Wilson}}$  are the Wilson gauge and quark actions, respectively. We use 20 gauge-field configurations and the light quark propagators are computed using a stochastic inversion based on the exact relation

$$(A^{-1})_{ij} = \frac{1}{Z} \int [d\phi] (A_{jk} \phi_k)^* \phi_i \times \exp(-\phi_l^* (A^\dagger A)_{lm} \phi_m), \quad (25)$$

where the  $\phi$  are auxiliary bosonic fields, introduced in order to perform the inversion of the matrix  $A$ , which in our case is the fermionic matrix. To reduce the statistical noise the technique of maximal variance reduction has been used [14]. The use of this stochastic inversion technique makes it possible to compute a light-quark propagator from each point in the half of the lattice with  $0 < t \leq 12$  (we call this region box I) to each point in the other half where  $12 < t \leq 24$  (box II). This increases considerably the effective statistics in the computation of the matrix elements.

We have performed the calculations at  $c_{\text{SW}} = 1.57$ , which is the numerical value of  $1/u_0^3$ , with  $u_0$  being the average value of a link variable as defined from the trace of the plaquette. The calculation is therefore “tadpole-improved” and hence the perturbative coefficients in Table 1 are the same as those which would be obtained with a tree-level improved action ( $c_{\text{SW}} = 1$ ). We have evaluated the matrix elements with two values of the light-quark mass, corresponding to  $\kappa_1 = 0.13847$  (for which  $m_\pi a \approx 0.74(4)$ ) and  $\kappa_2 = 0.14077$  (for which  $m_\pi a \approx 0.52(3)$ ) [15]. For this value of  $c_{\text{SW}}$   $\kappa_{\text{crit}} \approx 0.14351$  [15]. The same

lattice has been used to compute, with satisfactory results, the wave function of a  $B$ -meson and the effective coupling constant for the decay  $B^* \rightarrow B + \pi$  (in the Heavy Meson Chiral Lagrangian). This computation is reported in Ref. [16] and the values which we use for  $f_B^2 m_B$  are extracted from this paper.

The evaluation of the matrix elements requires the computation of two- and three-point correlation functions of the form,

$$C_2(t_x) = \sum_x \langle 0 | J(x) J^\dagger(0) | 0 \rangle, \quad (26)$$

where we have assumed  $t_x > 0$ , and

$$C_3(\mathcal{O}, t_x, t_y) = \sum_{x,y} \langle 0 | J(y) \mathcal{O}(0) J^\dagger(x) | 0 \rangle, \quad (27)$$

where  $t_y > 0 > t_x$ . In Eqs. (26) and (27)  $J$  and  $J^\dagger$  are interpolating operators which can destroy or create the  $\Lambda_b$  baryon, for which we take

$$J_\gamma^\dagger = \epsilon_{abc} (\bar{u}_\alpha^a (\gamma^5 C)_{\alpha\beta} \bar{d}_\beta^b) \bar{b}_\gamma^c, \quad (28)$$

where  $\bar{u}$ ,  $\bar{d}$ ,  $\bar{b}$  are the quark fields. In Eq. (28)  $a, b, c$  are colour labels,  $\alpha, \beta, \gamma$  are spinor labels and a sum over repeated indices is implied. We define  $Z_A$  by

$$Z_A u_\gamma^{(s)}(\mathbf{0}) = \frac{\langle \Lambda_b, s | J^\dagger(0) \gamma | 0 \rangle}{\sqrt{2m_A}}, \quad (29)$$

where  $s$  represents the spin state (up or down) of the baryon.

In order to enhance the contribution of the ground state to the correlation functions, it is useful to

Table 1

Matching coefficients for the matrix elements  $M_1$  and  $M_2$ , renormalised on lattice at an energy scale  $a^{-1} = 1.10$  GeV. The values of the integrals  $x_i$  are reported in the appendices of Ref. [10]

Coeff.	Expression	Value
$h_{11}$	$1 + \frac{\alpha}{4\pi} \left[ \frac{10}{3} - \frac{4}{3}x_1 - \frac{8}{3}x_2 \right] \simeq 1 - 21.65 \frac{\alpha}{4\pi}$	0.737
$h_{12}$	$\frac{\alpha}{4\pi} \left[ -\frac{4}{3}x_3 \right] \simeq 9.19 \frac{\alpha}{4\pi}$	0.112
$h_{13}$	$\frac{\alpha}{4\pi} \left[ -\frac{5}{2} - x_4 - 2x_5 - x_7 \right] \simeq 9.29 \frac{\alpha}{4\pi}$	0.113
$h_{14}$	$\frac{\alpha}{4\pi} \left[ -x_6 \right] \simeq 6.89 \frac{\alpha}{4\pi}$	0.084
$h_{21}$	$\frac{\alpha}{4\pi} \frac{2}{9} \left[ -\frac{5}{2} - x_4 - 2x_5 - x_7 \right] \simeq 2.06 \frac{\alpha}{4\pi}$	0.025
$h_{22}$	$\frac{\alpha}{4\pi} \left[ -\frac{2}{9}x_6 \right] \simeq 1.53 \frac{\alpha}{4\pi}$	0.019
$h_{23}$	$1 + \frac{\alpha}{4\pi} \left[ \frac{5}{12} - \frac{4}{3}x_1 + \frac{1}{3}x_2 - \frac{7}{6}x_4 + \frac{2}{3}x_5 - \frac{7}{6}x_7 \right] \simeq 1 - 10.82 \frac{\alpha}{4\pi}$	0.869
$h_{24}$	$\frac{\alpha}{4\pi} \left[ \frac{1}{2}x_6 \right] \simeq -3.45 \frac{\alpha}{4\pi}$	-0.042

“smear” the interpolating operators  $J$  and  $J^\dagger$ . In this paper we will follow Ref. [14] and adopt the type of smearing known as “fuzzing”. This technique consists in replacing light quark field  $q(x)$ , by a “fuzzed” field

$$\begin{aligned} q^F(x) = & \sum_{i=1,2,3} U_i^F(x) q(x + \hat{i}) \\ & + U_{-i}^F(x) q(x - \hat{i}), \end{aligned} \quad (30)$$

where  $U_{\pm i}^F(x)$  are defined by the recursive relations

$$\begin{aligned} U_i^F(x) = & \mathcal{P}_{SU(2)} \left[ \zeta U_i^F(x) \right. \\ & + \sum_{j \neq i} U_j^F(x) U_i^F(x + \hat{j}) U_{-j}^F(x + \hat{i} + \hat{j}) \\ & \left. + U_{-j}^F(x) U_i^F(x - \hat{j}) U_j^F(x + \hat{i} - \hat{j}) \right], \end{aligned} \quad (31)$$

$$\begin{aligned} U_{-i}^F(x) = & \mathcal{P}_{SU(2)} \left[ \zeta U_{-i}^F(x) \right. \\ & + \sum_{j \neq i} U_j^F(x) U_{-i}^F(x + \hat{j}) U_{-j}^F(x - \hat{i} + \hat{j}) \\ & \left. + U_{-j}^F(x) U_{-i}^F(x - \hat{j}) U_j^F(x - \hat{i} - \hat{j}) \right], \end{aligned} \quad (32)$$

starting with initial values  $U_i^F(x) = U_i(x)$  and  $U_{-i}^F(x) = U_i^\dagger(x - \hat{i})$ .  $\mathcal{P}_{SU(2)}$  is a projector on  $SU(2)$ , implemented as in the Cabibbo-Marinari cooling algorithm, and  $\zeta = 2.5$  is a constant value. The recursive procedure for  $U_{\pm i}^F(x)$  has been applied twice.

We introduce two superscripts on each correlation function, each of which can be either “ $F$ ” or “ $L$ ”, which indicate whether the interpolating operators  $J$  and  $J^\dagger$  are fuzzed or local.

The standard technique to extract hadronic matrix elements of the type  $\langle \Lambda_b | \mathcal{O} | \Lambda_b \rangle$  is to look for plateaus in the ratios

$$R(\mathcal{O}, t_1, t_2) = Z_A^2 \frac{C_3^{FF}(\mathcal{O}, t_1, t_2)}{C_2^{LF}(t_1) C_2^{LF}(t_2)}. \quad (33)$$

In our analysis, however, even with the use of fuzzed interpolating operators  $J$  and  $J^\dagger$ , we cannot eliminate the effects of excited states from the three-point correlation functions  $C_3^{FF}(\mathcal{O}, t_1, t_2)$  satisfactorily. On the other hand we do find that the ground state dominates the two-point correlation function  $C_2^{FF}(t)$  for  $t > 3$ , and the masses we obtain in this way agree, within errors, with those found previously on the same lattice [14] using a 3-mass correlated fit for a number of smeared correlators. In order to obtain the matrix elements  $M_i$  of Eqs. (17)–(20) we therefore need to subtract the effects of the excited states.

We have followed the following procedure to extract the matrix elements  $\langle \Lambda_b | \mathcal{O} | \Lambda_b \rangle$ :

- For each value of the light-quark mass we start by fitting the two-point correlation function for  $t > 3$  with a single exponential

$$C_2^{FF}(t) = (Z_A^F)^2 \exp(-m_A t), \quad (34)$$

thus obtaining the mass of the ground state,  $m_A$ . Within errors, the masses of the ground state which we obtain are in agreement with those obtained from the same lattice using a more sophisticated fitting procedure in Ref. [14].

- We model the contribution of the excited states by a second exponential and now fit  $C_2^{FF}(t)$  for  $t > 1$  by

$$C_2^{FF}(t) = (Z_A^F)^2 e^{-m_A t} + (Z_{A_1}^F)^2 e^{-m_{A_1} t}, \quad (35)$$

keeping  $m_A$  and  $Z_A^F$  fixed at the values obtained from the single exponential fit above. We find that  $C_2^{FF}$  is well represented by the two exponentials.

- For each operator  $\mathcal{O}$  we then fit the three-point correlation function  $C_3^{FF}(\mathcal{O}, t_1, t_2)$  to

$$\begin{aligned} C_3^{FF}(\mathcal{O}, t_1, t_2) &= \frac{\langle \Lambda_b | \mathcal{O} | \Lambda_b \rangle}{2m_A} (Z_A^F)^2 e^{-m_A(|t_1| + |t_2|)} \\ &+ C [e^{-m_A|t_1| - m_{A_1}t_2} + e^{-m_A|t_1| - m_{A_1}t_2}] \end{aligned} \quad (36)$$

obtaining values for the two unknown parameters  $\langle \Lambda_b | \mathcal{O} | \Lambda_b \rangle$  and the constant  $C$ , which encodes the contribution from excited states.

This procedure has been repeated for each of the 4 relevant operators, and for the linear combinations corresponding to  $L_1$  and  $L_2$  on 40 jackknife samples to extract the statistical errors.

In order to control the contributions from the excited states more effectively it will be necessary to carry out a simulation with considerably improved statistics. It is, however, possible to check the consistency of our approach *a posteriori*. We subtract the contributions from the excited states obtained above from the two- and three-point correlation functions, and look for plateaus in the ratios:

$$R(\mathcal{O}, t_1, t_2) = Z_A^2 \frac{\tilde{C}_3^{FF}(\mathcal{O}, t_1, t_2)}{\tilde{C}_2^{LF}(t_1)\tilde{C}_2^{LF}(t_2)}, \quad (37)$$

where the tilde indicates that contribution from excited states has been subtracted from the correlation function. The subtracted two-point correlation function is reported in Fig. 1. Fig. 2 shows the plateaus for the ratios  $R$  corresponding to the operators in  $L_1$  and  $L_2$  (with the appropriate normalization factor  $\frac{8}{f_B^2 m_B}$ ). The plateaus in Fig. 2 give us confidence in our treatment of the subtraction of the contribution of the excited states. The results for the matrix elements obtained from Eqs. (36) and (37) agree to within 1%.

Our results for the matrix elements at each of the two values of  $\kappa$  are reported in Table 2. Combining them with Eq. (2) we obtain

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = \begin{cases} 0.91(1) & \text{for } am_\pi = 0.52(5) \\ 0.93(1) & \text{for } am_\pi = 0.74(4). \end{cases} \quad (38)$$

From Eq. (38) we see that, although they are of  $O(1/m_b^3)$ , spectator effects are indeed significant (compare Eqs. (38) and (2)). Estimates of the parameter  $r$  defined in Eq. (4), using the non-relativistic quark model or the bag model [17,18] or QCD Sum Rules [19] are typically in the range 0.1–0.5. On the

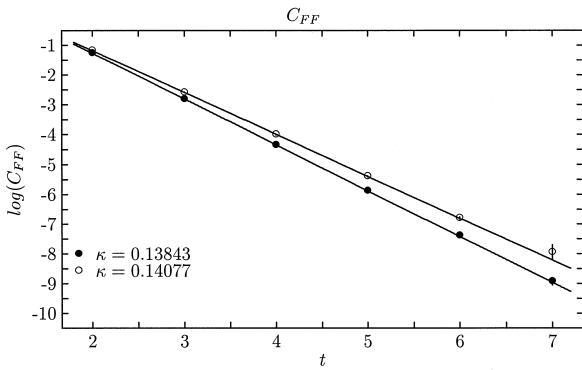


Fig. 1. Plot of  $C_2^{FF}$  after the subtraction for the contribution of excited states.

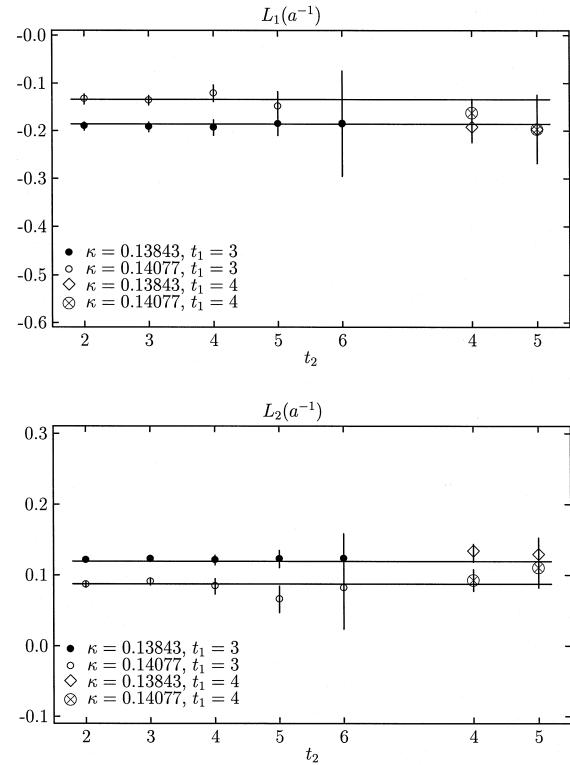


Fig. 2. Plot of the matrix elements  $L_1(a^{-1})$  and  $L_2(a^{-1})$  computed after the subtraction of excited states.

other hand, Rosner has estimated  $r$  from the spin splitting between  $\Sigma_Q$  and  $\Sigma_Q^*$  baryons ( $Q = c, b$ ) and finds  $r \approx 1$  (2) from charmed (beauty) baryons [20]. A recent reanalysis of this problem using QCD Sum Rules in which more condensates are introduced, finds a range of possible values for the ratio of lifetimes (including ones close to the experimental value in Eq. (1)), depending on the (unknown) values of various condensates [21]. At the two values of the light quark mass at which we do our computations we find  $r \approx 1.2 \pm 0.2$ , which is at the high end of expectations. The large values which we find for  $r$  and consequently the significant effect on the prediction for the lifetime ratios, make it important to improve the precision of the lattice simulations.

In quark models and related pictures, the parameter  $\tilde{B} = 1$ . In our simulations we find larger values,  $\tilde{B} = 1.9 \pm 0.2$  ( $1.3 \pm 0.2$ ) at  $\kappa_1$  ( $\kappa_2$ ).

#### 4. Conclusions

In this paper we have evaluated the matrix elements which contain the non-perturbative QCD ef-

Table 2

Lattice results for the matrix elements computed on lattice,  $M_i$ , the combined matrix elements at two different scales,  $L_i$ , and the physical ratio of lifetimes

Expression	$\kappa_1$	$\kappa_2$
$Z_A^2 Z_L^2 = f_B^2 m_B / 2$	0.33(1)	0.33(1)
$M_1$	−0.026(3)	−0.019(3)
$M_2$	0.045(4)	0.039(5)
$M_3$	0.018(2)	0.013(2)
$M_4$	−0.040(4)	−0.031(4)
$L_1(a^{-1})$	−0.18(2)	−0.13(3)
$L_2(a^{-1})$	0.21(2)	0.16(2)
$L_1(m_b)$	−0.31(3)	−0.22(4)
$L_2(m_b)$	0.23(2)	0.17(2)

fects in the spectator contribution to inclusive decays of the  $\Lambda_b$  baryon. Our principal results (in the  $\overline{\text{MS}}$  scheme) are presented in Table 2. The results indicate that spectator effects are important, accounting for a significant fraction of the discrepancy between the theoretical prediction for  $\tau(\Lambda_b)/\tau(B_d)$  in Eq. (2) and the experimental result in Eq. (1). It also appears that not all of the discrepancy can be accounted for by spectator effects.

The calculation described in this paper is the first evaluation, using lattice simulations, of the matrix elements in Eq. (8) and Eq. (9) between  $|\Lambda_b\rangle$  states. Having established that spectator effects are significant, it is now necessary to improve the precision, both statistical and systematic. This requires a high-statistics simulation at a smaller value of the lattice spacing (to decrease the errors due to discretisation) and with more values of the light quark mass (to enable a reliable extrapolation to the chiral limit).

In this study we have used static  $b$ -quarks. It would also be valuable, as a control of the systematic errors, to repeat the calculation with propagating  $b$ -quarks, for which these uncertainties are different.

## Acknowledgements

We wish to acknowledge Giulia De Divitiis, Luigi Del Debbio, Jonathan Flynn, Hartmut Wittig and other colleagues from the UKQCD collaboration for helpful discussions. Our simulations have been per-

formed on a Cray J90 at the Rutherford Appleton Laboratory (Oxford).

This work was supported by PPARC grants GR/L29927 and GR/L56329, and EPSRC grant GR/K41663.

## References

- [1] T. Junk, presented at the 2nd International Conference on  $B$ -Physics and  $CP$ -Violation, Honolulu, Hawaii, March 1997, quoted in Ref. [22]
- [2] J. Chay, H. Georgi, B. Grinstein, Phys. Lett. B 247 (1990) 399.
- [3] I.I. Bigi et al., hep-ph/9212227, Proc. 1992 Meeting of the Division of Particles and Fields of the APS (DPF92), Batavia, IL, November 1992, p. 610.
- [4] I.I. Bigi, M.A. Shifman, N.G. Uraltsev, A.I. Vainstein, Phys. Rev. Lett. 71 (1993) 496.
- [5] A.V. Manohar, M.B. Wise, Phys. Rev. D 49 (1994) 1310.
- [6] I. Bigi, M. Shifman, N. Uraltsev, Ann. Rev. Nucl. Part. Sci. 47 (1997) 591.
- [7] M. Neubert, Int. J. Mod. Phys. A 11 (1996) 4173.
- [8] I.I. Bigi, N.G. Uraltsev, A.I. Vainshtein, Phys. Lett. B 293 (1992) 430; B 297 (1993) 477 (E).
- [9] M. Neubert, C.T. Sachrajda, Nucl. Phys. B 483 (1997) 339.
- [10] M. Di Pierro, C.T. Sachrajda, Nucl. Phys. B 534 (1998) 373.
- [11] M.A. Shifman, M.B. Voloshin, Sov. J. Nucl. Phys. 41 (1985) 120; JETP 64 (1996) 698.
- [12] M.A. Shifman, M.B. Voloshin, Sov. J. Nucl. Phys. 45 (1987) 292.
- [13] B. Sheikholeslami, R. Wohlert, Nucl. Phys. B 259 (1985) 572.
- [14] C. Michael, J. Peisa, Phys. Rev. D 58 (1998) 034506.
- [15] UKQCD Collaboration: H.P. Shanahan et al., Phys. Rev. D 55 (1997) 1548.
- [16] G. De Divitiis, L. Del Debbio, M. Di Pierro, J. Flynn, C. Michael, J. Peisa, JHEP 10 (1998) 010.
- [17] B. Guberina, S. Nussinov, R. Peccei, R. Rückl, Phys. Lett. B 89 (1979) 111; N. Bilic, B. Guberina, J. Trampetic, Nucl. Phys. B 248 (1984) 261; B. Guberina, R. Rückl, J. Trampetic, Z. Phys. C 33 (1986) 297.
- [18] B. Blok, M. Shifman, in: J. Kirby, R. Kirby (Eds.), Proc. 3rd Workshop on the Tau-Charm Factory, Marbella, Spain, June 1993, Editions Frontières, 1994, hep-ph/9311331; I.I. Bigi et al., in: S. Stone (Ed.),  $B$ -Decays, second edition, World Scientific, Singapore, 1994, p. 134; I.I. Bigi, hep-ph/9508408.
- [19] P. Colangelo, F. De Fazio, Phys. Lett. B 387 (1986) 371.
- [20] J. Rosner, Phys. Lett. B 379 (1986) 267.
- [21] C-S. Huang, C. Liu, S-L. Zhu, hep-ph/9906300.
- [22] M. Neubert, invited talk at the International Europhysics Conference on High-Energy Physics (HEP 97), Jerusalem, Israel, August 1997; hep-ph/9801269.



ELSEVIER

## Spectator Effects in Inclusive Decays of Beauty Hadrons \*

**UKQCD Collaboration:** Massimo Di Pierro <sup>a</sup> and Chris T. Sachrajda<sup>a</sup><sup>a</sup>Department of Physics and Astronomy, University of Southampton, SO17 1BJ, United Kingdom

We evaluate the matrix elements of the four-quark operators which contribute to the lifetimes of  $B$ -mesons and the  $\Lambda_b$ -baryon. We find that the spectator effects are not responsible for the discrepancy between the theoretical prediction and experimental measurement of the ratio of lifetimes  $\tau(\Lambda_b)/\tau(B)$ .

## 1. INTRODUCTION

Inclusive decays of heavy hadrons can be studied in the framework of the heavy quark expansion, in which, for example, lifetimes are computed as series in inverse powers of the mass of the  $b$ -quark [1]. For an arbitrary hadron  $H$

$$\tau^{-1}(H) = \frac{G_F^2 m_b^5}{192\pi^3} \frac{|V_{cb}|^2}{2m_H} \sum_{i \geq 0} c_i m_b^{-i} \quad (1)$$

where

- $c_0$  corresponds to the decay of a free-quark and is universal.
- $c_1$  is zero because the operators of dimension four can be eliminated using the equations of motion.
- $c_2$  can be estimated and is found to be small.
- $c_3$  contains a contribution proportional to

$$\langle H | \bar{b}\Gamma q \bar{q}\tilde{\Gamma} b | H \rangle \quad (2)$$

and is therefore the first term in the expansion to which the interaction between the heavy and the light quark(s) contribute. Although this is an  $O(m_b^{-3})$  correction, it may be significant since it contains a phase-space enhancement.

The aim of our lattice simulation is to compute  $c_3$  for  $B$ -mesons and the  $\Lambda_b$ -baryon, in order to

check whether spectator effects contribute significantly to the ratios of lifetimes for which the experimental values are:

$$\frac{\tau(B^-)}{\tau(B^0)} = 1.06 \pm 0.04 \quad (3)$$

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = 0.78 \pm 0.04. \quad (4)$$

The discrepancy between the experimental value in eq. (4) and the theoretical prediction of  $\tau(\Lambda_b)/\tau(B^0) = 0.98$  (based on the Operator Product Expansion in eq. (1) including terms in the sum up to those of  $O(m_b^{-2})$ ) is a major puzzle. It is therefore particularly important to compute the  $O(m_b^{-3})$  spectator contributions to this ratio.

The ratios in eqs. (3) and (4) can be expressed in terms of 6 matrix elements:

$$\frac{\tau(B^-)}{\tau(B^0)} = a_0 + a_1 \varepsilon_1 + a_2 \varepsilon_2 + a_3 B_1 + a_4 B_2 \quad (5)$$

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = b_0 + b_1 \varepsilon_1 + b_2 \varepsilon_2 + b_3 L_1 + b_4 L_2 \quad (6)$$

where<sup>2</sup>

$$B_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b}\gamma^\mu L q \bar{q}\gamma_\mu L b | B \rangle}{2m_B} \quad (7)$$

$$B_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b}L q \bar{q}R b | B \rangle}{2m_B} \quad (8)$$

$$\varepsilon_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b}\gamma^\mu L t^a q \bar{q}\gamma_\mu L t^a b | B \rangle}{2m_B} \quad (9)$$

<sup>2</sup>In terms of the parameters  $\tilde{B}$  and  $r$  introduced in ref. [2]

$$r = -6L_1$$

$$\tilde{B} = -2L_2/L_1 - 1/3$$

\*talk presented by Massimo Di Pierro

$$\varepsilon_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b} L t^a q \bar{q} R t^a b | B \rangle}{2m_B} \quad (10)$$

$$L_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle \Lambda | \bar{b} \gamma^\mu L q \bar{q} \gamma_\mu L b | \Lambda \rangle}{2m_\Lambda} \quad (11)$$

$$L_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle \Lambda | \bar{b} \gamma^\mu L t^a q \bar{q} \gamma_\mu L t^a b | \Lambda \rangle}{2m_\Lambda} \quad (12)$$

and the coefficients  $a_i$  and  $b_i$  are given by

	value	value	
$a_0$	+1.00	$b_0$	+0.98
$a_1$	-0.697	$b_1$	-0.173
$a_2$	+0.195	$b_2$	+0.195
$a_3$	+0.020	$b_3$	+0.030
$a_4$	+0.004	$b_4$	-0.252

The values in the table correspond to operators renormalized in a continuum renormalization scheme at the scale  $\mu = m_B$ . Their matrix elements are obtained from those in the lattice regularization by perturbative matching (see the appendix).

## 2. B DECAY

The matrix elements  $B_1, B_2, \varepsilon_1, \varepsilon_2$  are computed on a  $24^3 \times 48$  lattice at  $\beta = 6.2$  (corresponding to a lattice spacing  $a^{-1} = 2.9(1)$  GeV) using the tree-level improved SW action for three values of  $\kappa = 0.14144, 0.14226, 0.14262$  and are then extrapolated to the chiral limit ( $\kappa_c = 0.14315$ ) [3]. We find

$$B_1 = 1.06 \pm 0.08 \quad (13)$$

$$B_2 = 1.01 \pm 0.06 \quad (14)$$

$$\varepsilon_1 = -0.01 \pm 0.03 \quad (15)$$

$$\varepsilon_2 = -0.02 \pm 0.02 \quad (16)$$

which implies that

$$\frac{\tau(B^-)}{\tau(B^0)} = 1.03 \pm 0.02 \pm 0.03 \quad (17)$$

in agreement with the experimental value (3).

## 3. Λ DECAY

The computation the baryonic matrix elements  $L_1$  and  $L_2$  is a little more difficult. We have performed an exploratory study in which the light

quark propagators are computed using a stochastic method [4] based on the relation

$$M_{ij}^{-1} = \int [d\phi] (M_{jk} \phi_k)^* \phi_i e^{-\phi_i^* (M^+ M)_{ij} \phi_j} \quad (18)$$

(rather than using “extended” propagators).

The matrix elements are computed on a  $12^3 \times 24$  lattice at  $\beta = 5.7$  (corresponding to a lattice spacing  $a^{-1} = 1.10(1)$  GeV) for two values of  $\kappa$ . We therefore do not attempt an extrapolation to the chiral limit ( $\kappa_c = 0.14351$ ) but present results separately for each value of  $\kappa$ . We find:

$$L_1 = \begin{cases} -0.30 \pm 0.03 & (\kappa = 0.13843) \\ -0.22 \pm 0.03 & (\kappa = 0.14077) \end{cases} \quad (19)$$

$$L_2 = \begin{cases} 0.23 \pm 0.02 & (\kappa = 0.13843) \\ 0.17 \pm 0.02 & (\kappa = 0.14077) \end{cases}, \quad (20)$$

which implies that (neglecting the systematic error due to the chiral extrapolation)

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = \begin{cases} 0.91 \pm 0.01 & (\kappa = 0.13843) \\ 0.93 \pm 0.01 & (\kappa = 0.14077) \end{cases}. \quad (21)$$

These results imply that spectator effects are not sufficiently large to explain the discrepancy between the theoretical prediction and experimental result in eq. (4).

## 4. CONCLUDING REMARKS

We find that the matrix elements of the 4-quark operators (13–16) satisfy the vacuum saturation hypothesis remarkably well. A similar feature is true for the  $\Delta B = 2$  operators which contribute to  $B - \bar{B}$  mixing. We do not have a good understanding yet of this phenomenon.

The results for the mesonic matrix elements lead to a prediction for the ratio of the lifetimes of the charged and neutral mesons which is in agreement with the experimental result in eq. (3). Our study of the baryonic matrix elements indicates that spectator effects are not sufficiently large to explain the experimental ratio of lifetimes in eq. (4). This discrepancy between the theoretical prediction and the experimental measurement remains an important problem to solve. We do stress, however, that our calculations are exploratory, and a more precise simulation is necessary, in particular to allow for a reliable extrapolation to the chiral limit.

Table 1  
Lattice perturbative coefficients and operators

$p_i$	$P_i$	$q_i$	$Q_i$
$-\frac{4}{3} \log(\lambda^2 a^2) + 8.46$	$\bar{b}\Gamma q\bar{q}\tilde{\Gamma}b$	+1	$\bar{b}\Gamma q\bar{b}\tilde{\Gamma}q$
$-\log(\lambda^2 a^2) + 6.19$	$\bar{b}t^a\Gamma t^a q\bar{q}\tilde{\Gamma}b + \bar{b}\Gamma q\bar{q}t^a\tilde{\Gamma}t^a b$	+1	$\bar{b}t^a\Gamma t^a q\bar{b}\tilde{\Gamma}q + \bar{b}\Gamma q\bar{b}t^a\tilde{\Gamma}t^a q$
-6.89	$\bar{b}t^a\gamma^0\Gamma\gamma^0 t^a q\bar{q}\tilde{\Gamma}b + \bar{b}\Gamma q\bar{q}t^a\gamma^0\tilde{\Gamma}\gamma^0 t^a b$	+1	$\bar{b}t^a\gamma^0\Gamma\gamma^0 t^a q\bar{b}\tilde{\Gamma}q + \bar{b}\Gamma q\bar{b}t^a\gamma^0\tilde{\Gamma}\gamma^0 t^a q$
$2\log(\lambda^2 a^2) - 4.53$	$\bar{b}t^a\Gamma q\bar{q}\tilde{\Gamma}t^a b$	-1	$\bar{b}t^a\Gamma q\bar{b}t^a\tilde{\Gamma}q$
$\log(\lambda^2 a^2) - 6.19$	$\bar{b}\Gamma t^a q\bar{q}\tilde{\Gamma}t^a b + \bar{b}t^a\Gamma q\bar{q}t^a\tilde{\Gamma}b$	-1	$\bar{b}\Gamma t^a q\bar{b}\tilde{\Gamma}t^a q + \bar{b}t^a\Gamma q\bar{b}\tilde{\Gamma}t^a q$
-6.89	$\bar{b}\Gamma\gamma^0 t^a q\bar{q}\tilde{\Gamma}\gamma^0 t^a b + \bar{b}t^a\gamma^0\Gamma q\bar{q}t^a\gamma^0\tilde{\Gamma}b$	+1	$\bar{b}\Gamma\gamma^0 t^a q\bar{b}t^a\gamma^0\tilde{\Gamma}q + \bar{b}t^a\gamma^0\Gamma q\bar{b}\tilde{\Gamma}\gamma^0 t^a q$
$-\log(\lambda^2 a^2) + 5.12$	$\bar{b}\Gamma t^a q\bar{q}\tilde{\Gamma}b$	-1	$\bar{b}\Gamma t^a q\bar{b}\tilde{\Gamma}t^a q$
$-\frac{1}{4} \log(\lambda^2 a^2) + 1.05$	$\bar{b}\Gamma\sigma^{\mu\nu}t^a q\bar{q}t^a\sigma^{\nu\mu}\tilde{\Gamma}b$	-1	$\bar{b}\Gamma\sigma^{\mu\nu}t^a q\bar{b}\tilde{\Gamma}\sigma^{\nu\mu}t^a q$
-2.43	$\bar{b}\Gamma\gamma^\mu t^a q\bar{q}t^a\gamma^\mu\tilde{\Gamma}b$	+1	$\bar{b}\Gamma\gamma^\mu t^a q\bar{b}\tilde{\Gamma}\gamma^\mu t^a q$

The analytic expression for  $p_i$  can be found in [3].

## ACKNOWLEDGEMENTS

It is a pleasure to thank Chris Michael, Hartmut Wittig, Jonathan Flynn, Luigi Del Debbio, Giulia De Divitiis, Vicente Gimenez and Carlotta Pittori for many helpful discussions. This work was supported by PPARC grants GR/L29927, GR/L56329 and GR/I55066.

## APPENDIX: 1-loop lattice perturbative corrections to 4-quark operators

The most difficult component of the evaluation of the 1-loop perturbative matching between four-quark lattice operators and those renormalized in a continuum scheme is the perturbative expansion on lattice. In this appendix we present the corresponding results for generic operators  $P_0$  and  $Q_0$  (whose matrix elements contribute to lifetimes and  $B$ - $\bar{B}$  mixing respectively):

$$P_0 \equiv \bar{b}\Gamma q\bar{q}\tilde{\Gamma}b, \quad (\Delta B = 0) \quad (22)$$

$$Q_0 \equiv \bar{b}\Gamma q\bar{b}\tilde{\Gamma}q, \quad (\Delta B = 2). \quad (23)$$

$\Gamma \otimes \tilde{\Gamma}$  represents an arbitrary spinor and color tensor. These operators mix under renormalization with other 4-quark operators, listed in table 1<sup>3</sup>:

$$P_0^{1\text{ loop}} = P_0 + \frac{\alpha_s(a^{-1})}{4\pi} \sum_i p_i P_i \quad (24)$$

$$Q_0^{1\text{ loop}} = Q_0 + \frac{\alpha_s(a^{-1})}{4\pi} \sum_i p_i q_i Q_i. \quad (25)$$

The Feynman rules correspond to the tree-level SW-improved action for massless light quarks and with a small gluon-mass ( $\lambda$ ) as the infrared regulator. The dependence on  $\lambda$ , of course, cancels when the corresponding continuum calculation is combined with the lattice one.

## REFERENCES

1. I. Bigi, M. Shifman and N. Uraltsev, Ann. Rev. Nucl. Part. Sci. **47** (1997) 591 and references therein.
2. M. Neubert and C. T. Sachrajda, Nucl. Phys. **B483** (1997) 339.
3. M. Di Pierro and C. T. Sachrajda, hep-lat/9805028 (to be published in Nucl. Phys. B).
4. C. Michael and J. Peisa, hep-lat/9802015.
5. M. Di Pierro, C. T. Sachrajda and C. Michael, in preparation.
6. V. Gimenez and J. Reyes, hep-lat/9806023

<sup>3</sup>These results were also obtained independently in ref. [6].

# B Lifetimes from Lattice Simulations\*

Massimo Di Pierro

Dept. of Physics and Astronomy, Univ. of Southampton, Southampton SO17 1BJ, UK

We evaluate the matrix elements of the four-quark operators which contribute to the lifetimes of  $B$ -mesons and the  $\Lambda_b$ -baryon. We find that the spectator effects are indeed larger than naive expectations based purely on power counting even if they do not appear to be sufficiently large to fully account for the discrepancy between the  $O(1/m_b^2)$  theoretical prediction and experimental measurement of the ratio of lifetimes  $\tau(\Lambda_b)/\tau(B)$ .

## 1. INTRODUCTION

Inclusive decays of heavy hadrons can be studied in the framework of the heavy quark expansion, in which lifetimes are computed as series in inverse powers of the mass of the  $b$ -quark [3]. For an arbitrary hadron  $H$

$$\tau^{-1}(H) = \frac{G_F^2 m_b^5}{192\pi^3} \frac{|V_{cb}|^2}{2m_H} \sum_{i \geq 0} c_i m_b^{-i} \quad (1)$$

where

- $c_0$  corresponds to the decay of a free-quark and is universal.
- $c_1$  is zero because the operators of dimension four can be eliminated using the equations of motion.
- $c_2$  can be estimated and is found to be small.
- $c_3$  contains a contribution proportional to

$$\langle H | \bar{b} \Gamma q \bar{q} \tilde{\Gamma} b | H \rangle \quad (2)$$

The term  $c_2$  contains an ambiguity due to the fact that when it is factorized in a perturbative part times a non perturbative one, the latter contains a divergence which is cancelled by the former only at an infinite order in perturbation theory. In order to avoid complications due to renormalon ambiguities we consider ratios of lifetimes. The differences in these ratios due to operators of dimension 5 can be estimated and are found to be small.

\*This letter is based on the two papers [1, 2] written in collaboration with C. T. Sachrajda and C. Michael.

The term of eq. (2) originates from integrating out the internal excitations of the box diagram in which two fermionic lines exchange two  $W$ . This is the first term in the expansion to which the interaction between the heavy quark and the constituent light (anti)quark contribute. Although this is an  $O(m_b^{-3})$  correction, it may be significant since it contains a phase-space enhancement.

The aim of our work is to compute (making use of lattice simulations) these spectator contributions to  $c_3$  for  $B_q$ -mesons and the  $\Lambda_b$ -baryon. The experimental values for the ratios of lifetimes of these particles are

$$\frac{\tau(B^-)}{\tau(B^0)} = 1.06 \pm 0.04 \quad (3)$$

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = 0.78 \pm 0.04. \quad (4)$$

The discrepancy between the experimental value in eq. (4) and the theoretical prediction of  $\tau(\Lambda_b)/\tau(B^0) = 0.98$  (based on the Operator Product Expansion in eq. (1) including terms in the sum up to those of  $O(m_b^{-2})$ ) is a major puzzle. It is therefore particularly important to compute the  $O(m_b^{-3})$  spectator contributions to this ratio.

The ratios in eqs. (3) and (4) can be expressed in terms of 6 matrix elements:

$$\frac{\tau(B^-)}{\tau(B^0)} = a_0 + a_1 \varepsilon_1 + a_2 \varepsilon_2 + a_3 B_1 + a_4 B_2 \quad (5)$$

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = b_0 + b_1 \varepsilon_1 + b_2 \varepsilon_2 + b_3 L_1 + b_4 L_2 \quad (6)$$

where <sup>2</sup>

$$B_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b} \gamma^\mu L q \bar{q} \gamma_\mu L b | B \rangle}{2m_B} \quad (7)$$

$$B_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b} L q \bar{q} R b | B \rangle}{2m_B} \quad (8)$$

$$\varepsilon_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b} \gamma^\mu L t^a q \bar{q} \gamma_\mu L t^a b | B \rangle}{2m_B} \quad (9)$$

$$\varepsilon_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle B | \bar{b} L t^a q \bar{q} R t^a b | B \rangle}{2m_B} \quad (10)$$

$$L_1 \equiv \frac{8}{f_B^2 m_B} \frac{\langle \Lambda | \bar{b} \gamma^\mu L q \bar{q} \gamma_\mu L b | \Lambda \rangle}{2m_\Lambda} \quad (11)$$

$$L_2 \equiv \frac{8}{f_B^2 m_B} \frac{\langle \Lambda | \bar{b} \gamma^\mu L t^a q \bar{q} \gamma_\mu L t^a b | \Lambda \rangle}{2m_\Lambda} \quad (12)$$

and the coefficients  $a_i$  and  $b_i$  are given by

	value	value	
$a_0$	+1.00	$b_0$	+0.98
$a_1$	-0.697	$b_1$	-0.173
$a_2$	+0.195	$b_2$	+0.195
$a_3$	+0.020	$b_3$	+0.030
$a_4$	+0.004	$b_4$	-0.252

The values in the table correspond to operators renormalized in the continuum  $\overline{\text{MS}}$  renormalization scheme at the scale  $\mu = m_B$ .

## 2. GENERAL REMARKS

Matrix elements such as those of eqs. (7–12) encode the contribution of soft QCD effects which cannot be evaluated in perturbation theory. These effects can be computed by performing a numerical evaluation of the path integral that defines the matrix elements. The standard procedure consists in approximating a portion of continuum space-time with a discrete four dimensional lattice endowed with an Euclidean metric, and formulating QCD on this lattice. The theory, without any model dependent assumption,

<sup>2</sup>In terms of the parameters  $\tilde{B}$  and  $r$  introduced in ref. [4]

$$\begin{aligned} r &= -6L_1 \\ \tilde{B} &= -2L_2/L_1 - 1/3 \end{aligned}$$

presents an exact gauge invariance. The discrepancy between the discretized correlation functions and the continuum ones is a function of the lattice spacing ( $a$ ) that can (in principle) be systematically reduced.

The lattice provides a natural momentum cut-off because modes with frequency higher than the inverse lattice spacing cannot propagate. The limit  $a \rightarrow 0$  can be taken and it becomes evident how the lattice formulation of QCD is just a way of regularizing the theory. Moreover for any finite value of  $a$ , lattice QCD can be thought of as an effective theory of QCD: the effects of heavy modes are encoded in its parameters (which can be evaluated by direct comparison with phenomenology) and in the matching coefficients (which can be evaluated in perturbation theory) that multiply the lattice matrix elements (which are computed numerically).

To relate the matrix elements of eqs. (7–12) to the lattice ones, it is necessary to evolve them down to an energy scale that can be simulated,  $a^{-1}$  (using the renormalization group equation), and then match them with the matrix elements regularized on the lattice (and renormalized at the lattice scale,  $a$ ). In our simulations we adopt a Sheikholeslami-Wolpert (SW) lattice action which improves the convergence of the correlation functions to those of continuum QCD from order  $O(a)$  to order  $O(\alpha_s a)$ . We compute the corresponding matching coefficients at one loop. Our ignorance of higher order coefficients introduces a systematic error that we include in our analysis.

Moreover because the lattice describes only a finite portion of space-time, massless modes cannot propagate even. For this reason the light quarks ( $u$  and  $d$ ) are simulated with higher masses,

$$m_{\text{light}} \simeq (2a)^{-1} (\kappa^{-1} - \kappa_{\text{crit}}^{-1}) \quad (13)$$

for several values of the parameter  $\kappa$ . Then the results are extrapolated to the chiral limit ( $\kappa \rightarrow \kappa_{\text{crit}}$ ). This procedure introduces an error that, for the matrix elements of eqs. (7–10), has been evaluated. For the matrix elements of eqs. (11–12) this extrapolation has not been possible because only two values for the light quark masses are available at the moment.

The most important systematic error intro-

duced in lattice simulations is the quenched approximation, i.e. quark loops are neglected. This is due to limitations in present computing power and in principle the effects of loops could be evaluated in future simulations. The size of this systematic error is not known and is not quoted in our results.

We have adopted the following standard technique to extract matrix elements of the form  $\langle H|\mathcal{O}_i|H\rangle$  from the lattice. We construct a lattice operator  $J^\dagger$  which has the same quantum numbers as the hadron  $H$ , where  $H$  is the lightest state with a non vanishing superposition with  $J^\dagger$ . We evaluate on the lattice the zero momentum Fourier transform of the 2-point correlation function:

$$C_2(t) = \int d^3x \langle 0|J(\mathbf{x}, t)J^\dagger(\mathbf{x}, 0)|0\rangle \quad (14)$$

and of the 3-point one:

$$C_{3i}(t) = \int d^3x \langle 0|J(\mathbf{x}, t)\mathcal{O}_i(\mathbf{x}, 0)J^\dagger(\mathbf{x}, -t)|0\rangle \quad (15)$$

In the Euclidean space, for large  $t$ , these correlation functions are dominated by the lightest particle created by  $J^\dagger$ , i.e.  $H$ , therefore

$$\lim_{t \rightarrow \infty} Z_{ij} Z_J^2 \frac{C_{3j}(t)}{[C_2(t)]^2} = \frac{\langle H|\mathcal{O}_i|H\rangle}{2m_H} \quad (16)$$

enables us to extract the required matrix element.

The factor  $Z_J$  measures the superposition between  $H$  and  $J^\dagger$  and it can be extracted by fitting the large  $t$  behaviour of  $C_2(t)$  with a single exponential

$$C_2(t) \simeq Z_J e^{mt} \quad (17)$$

The coefficients  $Z_{ij} = \delta_{ij} + O(\alpha_s)$  are the matching coefficients that we have evaluated perturbatively.

In general there are many possible choices for the smearing of the interpolation operator  $J$ . In the case of the operators of eqs. (7-10), we checked that the matrix elements do not depend on the smearing procedure.

### 3. B DECAY

The matrix elements  $B_1, B_2, \varepsilon_1, \varepsilon_2$  are computed on a  $24^3 \times 48$  lattice at  $\beta = 6.2$  (corresponding to a lattice spacing  $a^{-1} = 2.9(1)$  GeV) using

the tree-level improved SW action for three values of  $\kappa = 0.14144, 0.14226, 0.14262$  and are then extrapolated to the chiral limit ( $\kappa_{crit} = 0.14315$ ) [1]. We find

$$B_1 = +1.06 \pm 0.08 \quad (18)$$

$$B_2 = +1.01 \pm 0.06 \quad (19)$$

$$\varepsilon_1 = -0.01 \pm 0.03 \quad (20)$$

$$\varepsilon_2 = -0.02 \pm 0.02 \quad (21)$$

which implies that

$$\frac{\tau(B^-)}{\tau(B^0)} = 1.03 \pm 0.02 \pm 0.03 \quad (22)$$

in agreement with the experimental value, eq. (3). In eq. (22) the first error is purely statistical while the second one encodes our evaluation for systematical uncertainties (excluding quenching effects).

### 4. $\Lambda$ DECAY

The computation the baryonic matrix elements  $L_1$  and  $L_2$  is a little more difficult because of the presence of two constituent light quarks in  $\Lambda_b$ . We have performed an exploratory study in which the light quark propagators are computed using a stochastic method [5] based on the relation

$$M_{ij}^{-1} = \int [d\phi] (M_{jk}\phi_k)^* \phi_i e^{-\phi_i^*(M^+ M)_{lm} \phi_m} \quad (23)$$

The matrix elements are computed on a  $12^3 \times 24$  lattice at  $\beta = 5.7$  (corresponding to a lattice spacing  $a^{-1} = 1.10(1)$  GeV) for two values of  $\kappa$ . We therefore do not attempt an extrapolation to the chiral limit ( $\kappa_{crit} = 0.14351$ ) but present our results separately for each value of  $\kappa$ . We find:

$$L_1 = \begin{cases} -0.30 \pm 0.03 & (\kappa = 0.13843) \\ -0.22 \pm 0.03 & (\kappa = 0.14077) \end{cases} \quad (24)$$

$$L_2 = \begin{cases} +0.23 \pm 0.02 & (\kappa = 0.13843) \\ +0.17 \pm 0.02 & (\kappa = 0.14077) \end{cases}, \quad (25)$$

which implies that

$$\frac{\tau(\Lambda_b)}{\tau(B^0)} = \begin{cases} 0.91 \pm 0.01 & (\kappa = 0.13843) \\ 0.93 \pm 0.01 & (\kappa = 0.14077) \end{cases}. \quad (26)$$

We stress again that these errors do not include systematical errors due to the chiral extrapolation and to quenching. Our results indicates that

spectator effects give rise to a significant difference in the lifetimes of the  $\Lambda_b$  and  $B$ . We remark that our computation of the  $L_1$  and  $L_2$  matrix elements must be considered exploratory because of the large lattice spacing, the size of the lattice and the lack of a chiral extrapolation. We believe that it is important to repeat the calculation on a larger lattice with better statistics.

## 5. NOTES ON FACTORIZATION

While performing our simulation we have found that the matrix elements of the 4-quark operators (18–21) satisfy the vacuum saturation hypothesis (also known as factorization) remarkably well. In fact within statistical errors

$$\langle B_q | \bar{b} \Gamma q \bar{q} \tilde{\Gamma} b | B_q \rangle \simeq \langle B_q | \bar{b} \Gamma q | 0 \rangle \langle 0 | \bar{q} \tilde{\Gamma} b | B_q \rangle \quad (27)$$

is verified for any couple,  $\Gamma$  and  $\tilde{\Gamma}$ , of color  $\otimes$  spin matrices. From the theoretical point of view eq. (27) is true at tree level, but no argument is known to prove that it holds at higher orders in perturbation theory or non-perturbatively. In particular factorization cannot hold at every scale since the renormalization group behaviour is different on both sides of eq. (27).

An analogous relation has been observed for the  $B - \bar{B}$  system

$$\langle B_q | \bar{b} \Gamma q \bar{b} \tilde{\Gamma} q' | \bar{B}_{q'} \rangle \simeq \langle B_q | \bar{b} \Gamma q | 0 \rangle \langle 0 | \bar{b} \tilde{\Gamma} q' | \bar{B}_{q'} \rangle \quad (28)$$

This matrix element contributes as one of the two possible contractions to the matrix element which dominates the mixing amplitude:

$$\begin{aligned} \langle B_q | \bar{b} \Gamma q \bar{b} \tilde{\Gamma} q | \bar{B}_q \rangle &= \langle B_q | \bar{b} \Gamma q \bar{b} \tilde{\Gamma} q' | \bar{B}_{q'} \rangle \\ &+ \langle B_q | \bar{b} \Gamma q' \bar{b} \tilde{\Gamma} q | \bar{B}_{q'} \rangle \end{aligned} \quad (29)$$

Note that since the second contraction can be reduced to the first by use of the Fierz identities, the assumption of factorization completely determines the total mixing amplitude, eq. (29). Results for  $B - \bar{B}$  mixing consistent with factorization have been obtained independently by many groups among the lattice community, and there is general agreement on them.

We believe that the observed phenomenon of factorization constitutes *per se* an interesting result that has to be explained to fully understand the physics of  $B$  decays and  $B - \bar{B}$  mixing.

## ACKNOWLEDGEMENTS

It is a pleasure to thank Chris Michael, Hartmut Wittig, Jonathan Flynn, Luigi Del Debbio, Giulia De Divitiis, Vicente Gimenez and Carlotta Pittori for many helpful discussions. I also want to thank Chris T. Sachrajda for his help, his support and his teachings during my staying in Southampton.

This work was supported by PPARC grants GR/L29927, GR/L56329 and GR/I55066.

## REFERENCES

- 1 M. Di Pierro and C. T. Sachrajda, Nucl.Phys. **B534** (1998) 373–391
- 2 M. Di Pierro, C. T. Sachrajda and C. Michael, hep-lat/9906031
- 3 I. Bigi, M. Shifman and N. Uraltsev, Ann. Rev. Nucl. Part. Sci. **47** (1997) 591 and references therein.
- 4 M. Neubert and C. T. Sachrajda, Nucl. Phys. **B483** (1997) 339.
- 5 C. Michael and J. Peisa, hep-lat/9802015.

## Lattice determination of the $B^*B\pi$ coupling \*

G. De Divitiis<sup>a</sup>, L. Del Debbio<sup>b</sup>, M. Di Pierro<sup>b</sup>, J.M. Flynn<sup>b</sup> and C. Michael<sup>c</sup> (UKQCD Collaboration)

<sup>a</sup> Dip. di Fisica, Univ. di Roma “Tor Vergata”, via della ricerca scientifica 1, 00133 Roma, Italy

<sup>b</sup> Dept. of Physics and Astronomy, Univ. of Southampton, Southampton SO17 1BJ, UK

<sup>c</sup> DAMTP, Univ. of Liverpool, Liverpool L69 3BX, UK

The coupling  $g_{B^*B\pi}$  is related to the form factor at zero momentum of the axial current between  $B^*$  and  $B$  states. Moreover it is related to the effective coupling between heavy mesons and pions that appears in the heavy meson chiral Lagrangian. This coupling has been evaluated on the lattice using static heavy quarks and light quark propagators determined by a stochastic inversion of the fermionic bilinear. We found the value  $g = 0.42(4)(8)$ . Beside its theoretical interest, this quantity has phenomenological implications in  $B \rightarrow \pi + \bar{l}l$  decays.

### 1. INTRODUCTION

The matrix element

$$\langle B^0(p)\pi^+(q) | B^{*+}(p') \rangle \quad (1)$$

is usually parametrised in terms of the form factor  $g_{BB^*\pi}(q^2)$  in the following way:

$$-g_{B^*B\pi}(q^2)q_\mu\eta^\mu(p')(2\pi)^4\delta^4(p' - p' - q) \quad (2)$$

where  $\eta^\mu(p')$  is the polarization vector of the asymptotic state  $B^{*+}(p')$ .

For on-shell external states,  $p' = p + q$ , one can perform an LSZ reduction of the pion in eq. (1)

$$i(m_\pi^2 - q^2) \int e^{iqx} \langle B(p) | \pi(x) | B^*(p') \rangle d^3x \quad (3)$$

Using the PCAC definition of the pion in terms of the axial current

$$\pi(x) = \frac{1}{m_\pi^2 f_\pi} \partial^\mu A_\mu(x) \quad (4)$$

one obtains

$$q^\mu \frac{m_\pi^2 - q^2}{m_\pi^2 f_\pi} \int e^{iqx} \langle B^0(p) | A_\mu(x) | B^*(p') \rangle d^3x \quad (5)$$

Since in the limit  $q \rightarrow 0$ , eq. (5) and eq. (2) parametrise the same matrix element, eq. (1), one derives a Goldberger-Treiman like relation for the pion system[1]

$$g_{B^*B\pi}(0) = \frac{2m_B}{f_\pi} g \quad (6)$$

where  $g$  is defined as

$$g = \int \frac{\langle B(p) | A_\mu(x) | B^*(p') \rangle}{2m_B} \eta^\mu d^3x \quad (7)$$

This parameter  $g$  coincides with the effective coupling of the Heavy Meson Chiral Lagrangian[2] and can be used to constrain the form factors that appear in the matrix elements of the weak vector current,  $\bar{u}\gamma^\mu b$ . One of the best present techniques to extract  $V_{ub}$  is by comparing the experimental values for these form factors with the theoretical ones, obtained by fitting lattice Monte Carlo results. Therefore a precise determination of  $g$  can be used to increase the precision in the lattice determination of these form factors and reduce the theoretical uncertainty on  $V_{ub}$ .

Even though the matrix element of eq. (1) does not directly correspond to a physical process, because the  $B^* \rightarrow B + \pi$  is kinematically forbidden, the equivalent process for  $D$  systems  $constant1/m_c$  corrections.

### 2. SIMULATION

We have performed our simulations on 20 quenched gauge configurations, generated on a  $12^3 \times 24$  lattice at  $\beta = 5.7$ , corresponding to  $a^{-1} = 1.10$  GeV, with a tadpole improved SW action ( $c_{SW} = 1.57$ ). The heavy quark propagators are evaluated in the static limit, while the light quark propagators are evaluated performing

\*talk presented by Massimo Di Pierro

a stochastic inversion on the fermion matrix  $Q$

$$(Q^{-1})_{ij} = \int [d\phi] (Q_{jk}\phi_k)^* \phi_i e^{-\phi_i^*(Q^\dagger Q)_{lm} \phi_m} \quad (8)$$

with 10 pseudofermionic fields  $\phi_i$  for each gauge configuration, generated by Monte Carlo. Moreover, the maximal variance reduction technique is implemented to reduce the statistical noise[3]. Two values of  $\kappa_1 = 0.13843$  and  $\kappa_2 = 0.14077$ , corresponding to a light quark mass of 140 MeV and 75 MeV respectively, with a critical value  $\kappa_{crit} = 0.14351$ . The interpolating operator  $J^\dagger(J)$  that creates (annihilates) a static  $B$  meson is smeared with a two-step fuzzing procedure for the light quark fields. We indicate the local operators with the superscript L and the fuzzed ones with the superscript F.

We have evaluated the two point correlation function at zero momentum

$$C_2^{FF(FL)}(t) = \overline{\langle 0 | J(\mathbf{y}, 0) J^\dagger(\mathbf{y}, t) | 0 \rangle} \quad (9)$$

both for two-fuzzed (FF) and one-fuzzed one-local (FL)  $J$  operators. The average is on the spatial position  $\mathbf{y}$  and it has the effect of increasing the statistics. This is made possible by using all-to-all stochastic propagators. From fitting the asymptotic (large  $t$ ) behaviour of  $C_2$  with a single exponential we have extracted  $Z^{L(F)} = \langle 0 | J^{L(F)} | B \rangle / \sqrt{2m_B}$ .

Analogously we have computed the three point correlation function at zero momentum

$$C_{3\mu}^{FF}(r, t_1, t_2) = \overline{\langle 0 | J(\mathbf{y}, -t_1) A_\mu(\mathbf{x} + \mathbf{y}, 0) J^\dagger(\mathbf{y}, t_2) | 0 \rangle} \quad (10)$$

for two-fuzzed  $J$  operators. Here the average is on both the spatial coordinates  $\mathbf{y}$  and  $\mathbf{x}$ , but keeping fixed  $r = |\mathbf{x}|$ . For the spatial indices  $\mu = 1, 2, 3$  we also used the rotational invariance of the lattice so that  $C_{31} = C_{32} = C_{33}$ .

Finally we have computed

$$E^0(r, t) = (Z^F)^2 C_{30}^{FF}(r, t, t) / [C_2^{FF}(t)]^2 \quad (11)$$

and

$$\overline{E}(r, t) = (Z^F)^2 C_{31}^{FF}(r, t, t) / [C_2^{FF}(t)]^2 \quad (12)$$

These two functions of  $r$  are plotted in fig. 1 for the lightest value of  $\kappa$  and values of  $t = 3, 4, 5, 6$ .

As expected  $E^0(r, t) \simeq 0$ , because the polarization of a static  $B^*$  meson is orthogonal to the temporal direction, while the averaged spatial component  $\overline{E}(r, t)$  is almost independent on  $t$  and presents an exponential decay in  $r$  (this is confirmed by a number of different fitting tests and by visualizing the points in log scale). In fact in the asymptotic regime in  $t$

$$\overline{E}(r, t) = \langle B | A_\mu(r) | B^* \rangle / \sqrt{2m_B} \quad (13)$$

Going back to the definition, eq. (7), the lattice regularised  $g$  coupling can be extracted from the spatial integral of  $f(r) = S e^{-r/r_0}$ , the function fitting  $\overline{E}(r, t)$ . We find

$$g^{\text{latt}} = \int f(r) (4\pi r^2) dr = \begin{cases} 0.54(5) & \text{for } \kappa_1 \\ 0.53(5) & \text{for } \kappa_2 \end{cases} \quad (14)$$

We observe that the main source of error in  $g^{\text{latt}}$  is the lattice breaking of rotational invariance, in fact the distribution of the points  $\overline{E}(r)$  around the fitting function  $f(r)$  exhibits a regular pattern which is independent on  $t$  and  $\kappa$ .

### 3. MATCHING

The  $B$  meson decay constant can be extracted, in the static approximation, from

$$f_B^{\text{static}} = Z_A^{\text{static}} \sqrt{\frac{2}{m_B}} Z^L a^{-3/2} \quad (15)$$

where for our lattice  $Z_A^{\text{static}} = 0.78$  is computed using the Lepage-Mackenzie procedure[4] of matching lattice vs continuum at the best scale, which in our case is determined to be  $q^* a = 2.29$ . Our result is  $f_B^{\text{static}} = 0.43(1)(8)$  GeV. We observe that a single-exponential fit gives a value of  $f_B^{\text{static}}$  20% bigger than the value extracted from a three-exponential fit[3]. In fact we obtain a value for this quantity that lies above other lattice calculations of the same quantity. We take into account this effect by adding a systematic error to our results of the order 20%.

The quantity we are interested in is the  $g$  coupling renormalized in the  $\overline{\text{MS}}$  scheme at the  $m_B$  scale

$$g = Z_A^{\text{tadpole}} g^{\text{latt}} = Z_A^{\text{1-loop}} \frac{u_0}{u_0^{\text{1-loop}}} g^{\text{latt}} \quad (16)$$

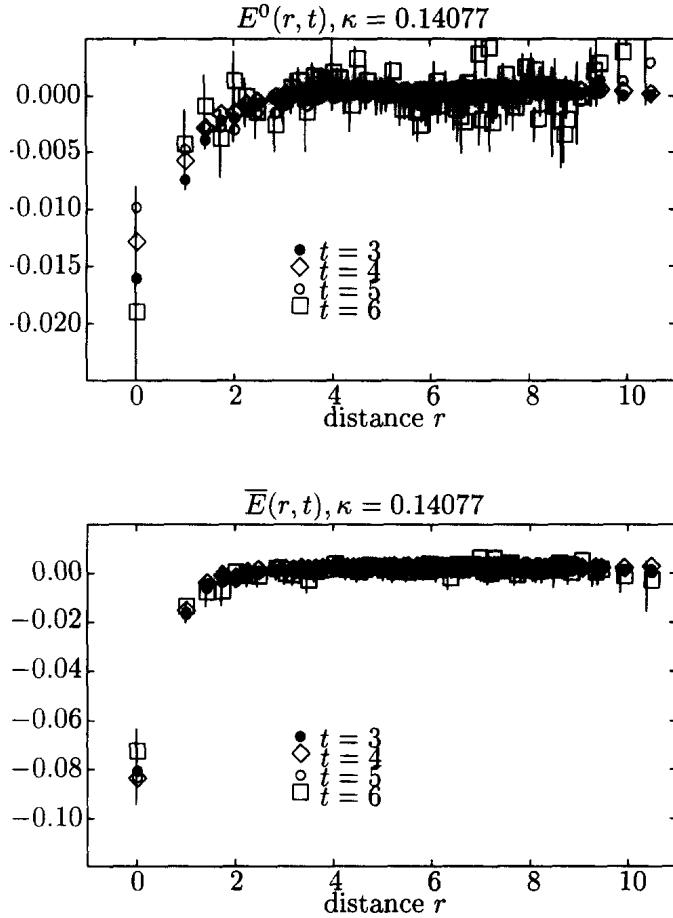


Figure 1. Plots of  $E^0(r, t)$  and  $\bar{E}(r, t)$  as functions of  $r$  for different values of  $t$  ( $= 3, 4, 5, 6$ ) for  $\kappa_2$ .

where  $Z_A^{1\text{-loop}} = 1 - 13.8 \frac{\alpha_{CE}}{4\pi}$ ,  $u_0^{1\text{-loop}} = 1 - \pi^2 \frac{\alpha_{CE}}{4\pi}$  and  $u_0 = 0.86081$  is the average plaquette. These values imply  $Z_A^{\text{tadpole}} = 0.806$ .

#### 4. CONCLUSIONS

Substituting the values of eq. (14) into eq. (16) we obtain

$$g = \begin{cases} 0.44(4) & \text{for } \kappa_1 \\ 0.43(4) & \text{for } \kappa_2 \end{cases} \quad (17)$$

Performing a naive extrapolation to the chiral limit and including our evaluation for the system-

atic error we summarise our result as

$$g = 0.42(4)(8) \quad (18)$$

This number has to be compared to best estimate from a global analysis of available results[2]

$$g \simeq 0.38 \quad (19)$$

and with an estimate obtained from lattice data for the semileptonic B decay form factor (assuming Vector Meson Dominance),

$$g = 0.50(5)(10) \quad (20)$$

One remark is in order. We consider this study an exploratory one because of the small lattice, the large lattice spacing, the poor chiral extrapolation and the use of the novel technique of stochastic propagators. As we observed, the  $g$  parameter has important phenomenological applications and we believe our study shows that this number can be evaluated on the lattice. A better determination is required and is feasible with present computing power.

#### REFERENCES

1. G. De Divitiis, L. Del Debbio, M. Di Pierro J. Flynn, C. Michael and J. Peisa [UKQCD Collaboration], JHEP 9810 (1998) 010, and ref. therein.
2. for a review, see R. Casalbuoni *et al.*, Phys. Rep. 281 (1997) 145
3. C. Michael and J. Peisa, hep-lat/9802015
4. G. P. Lepage and P. B. Mackenzie, Phys. Rev. D48 (1993) 225

## The second moment of the pion's distribution amplitude \*

L. Del Debbio, M. Di Pierro, A. Dougall and C. Sachrajda (UKQCD Collaboration)

Dept. of Physics and Astronomy, Univ. of Southampton, Southampton SO17 1BJ, UK

We present preliminary results for the second moment of the pion's distribution amplitude. The lattice formulation and the phenomenological implications are briefly reviewed, with special emphasis on some subtleties that arise when the Lorentz group is replaced by the hypercubic group. Having analysed more than half of the available configurations, the result obtained is  $\langle \xi^2 \rangle_L = 0.06 \pm 0.02$ .

### 1. INTRODUCTION

The distribution amplitude  $\phi(x_1, x_2)$  gives the probability for finding two collinear partons with fractions  $x_1$  and  $x_2$  of the meson's momentum [1]. It plays an important role e.g. in exclusive hard scattering processes and in non-leptonic decays of heavy mesons, since it allows one to separate a hard scattering amplitude, computed in perturbation theory, from the effect of the soft Fourier modes, whose dynamics is non-perturbative.

For example, the relevant diagram for the pion electromagnetic form factor is shown in Fig. 1. The form factor is defined as:

$$\langle \pi(p') | V_\mu | \pi(p) \rangle = F(q^2)(p + p')_\mu \quad (1)$$

and can be written in terms of the distribution amplitude:

$$F(Q^2) = \int [dx][dy] \phi^\dagger(x, Q^2) T_H(x, y; Q^2) \phi(y, Q^2) \quad (2)$$

where  $[dx] = dx_1 dx_2 \delta(x_1 + x_2 - 1)$ , and  $T_H$  is the perturbative amplitude computed at the quark level.

The amplitude  $\phi$  is given by the non-local Fourier transform of the matrix element of the fermionic bilinear [2]:

$$\phi_{\alpha\beta}^{ab}(x, Q^2) \sim FT [\langle 0 | T \psi_\alpha^a(z_1) \bar{\psi}_\beta^b(z_2) | \pi \rangle] \quad (3)$$

A light-cone expansion relates  $\phi$  to fermion bilinears with covariant derivatives. In particular the

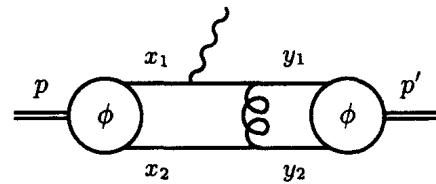


Figure 1. Photon-pion interaction, factorised into a hard (perturbative) scattering amplitude and a non-perturbative part described by the pion distribution amplitude.

$m$ -th moment,

$$\langle \xi^m \rangle \equiv \int d\xi \xi^m \phi(\xi, Q^2), \quad (4)$$

can be extracted from the matrix element of the lowest-twist operator appearing in the OPE of Eq. 3:

$$\langle 0 | O_{\mu_0 \dots \mu_m}(0) | \pi(p) \rangle = f_\pi p_{\mu_0} \dots p_{\mu_m} \langle \xi^m \rangle + \dots \quad (5)$$

where

$$O_{\mu_0 \dots \mu_m}(0) = (-i)^n \bar{\psi} \gamma_{\mu_0} \gamma_5 \overset{\leftrightarrow}{D}_{\mu_1} \dots \overset{\leftrightarrow}{D}_{\mu_m} \psi \quad (6)$$

and the ellipses indicate terms that are proportional to  $p^2 g_{\mu_i \mu_j}$ .

The distribution amplitude has been studied in recent years using sum rules (see results quoted in [1]) and lattice simulations [3,4]. The accuracy of the lattice determinations so far has not been good enough to compare precisely with sum rules predictions. The aim of the current study is to use modern lattice technology to improve the precision of the result.

\*talk presented by Luigi Del Debbio

## 2. LATTICE DETAILS

The following UKQCD quenched configurations have been used to compute the relevant matrix elements.

- $\beta = 6.2$ ,  $a^{-1} = 2.64 \pm 0.10$  GeV,  $24^3 \times 48$  lattice;
- SW action, with  $c_{\text{sw}} = 1.61$ ;
- the three  $\kappa$  values used in our simulation (0.13460, 0.13510, 0.13530) correspond to physical light pseudoscalar masses ranging from 350 to 850 MeV.
- 180 configurations are available, the results presented here are based on the analysis of a subset of 129;
- non-improved local operators have been used, without fuzzing of the light quarks.

## 3. POWER-LIKE DIVERGENCIES

In this work, we concentrate on the second moment of the distribution amplitude, i.e.  $m = 2$  in Eq. 5 and 6 above. The choice of the Lorentz indices for  $O_{\mu_0\mu_1\mu_2}$  is crucial in order to avoid mixing with lower-dimensional operators. We choose  $\mu_0$  to be the time-direction. It is convenient, but not necessary, to avoid time-derivatives and hence we do not symmetrise  $\mu_0$  with  $\mu_1, \mu_2$ . Using the notation of [5] to classify the irreducible representations of the hypercubic group, we obtain:

- $O_{\mu[\nu\sigma]}$ , with  $\mu \neq \nu \neq \sigma \neq \mu$  and symmetrised over  $\nu$  and  $\sigma$ , transforms like a  $(\frac{1}{2}, \frac{1}{2}) \oplus \mathbf{8}$  reducible representation, and therefore does not mix with lower dimensional operators. However the  $\mathbf{8}$  irrep leads to a term proportional to  $p_\mu \left[ \frac{p_\nu^2 + p_\sigma^2}{2} - p_\tau^2 \right]$ , where  $\tau$  is the fourth available index, which needs to be subtracted in order to have an operator proportional to  $p_\mu p_\nu p_\sigma$ , as in the continuum limit. This subtraction is performed in the definition of  $R_1$  below. Note that two spatial components of  $p$  need to be non-zero to get a non-vanishing signal.

- $O_{\mu\nu\nu}$ , with  $\mu \neq \nu$  transforms like a  $(\frac{1}{2}, \frac{1}{2}) \oplus \mathbf{8}$  representation; however the subtracted operator:

$$O'_{411} = \left( O_{411} - \frac{O_{422} + O_{433}}{2} \right)$$

simply transforms like  $\mathbf{8}$ . In this case there is no mixing, and a non-zero signal is obtained with just one non-vanishing component of  $p = (1, 0, 0)$ .

## 4. PRELIMINARY RESULTS

The relevant matrix elements are extracted from suitably defined two-point functions. For a generic operator  $Q$ :

$$C_2^Q(t) = \sum_{\mathbf{x}} e^{i\mathbf{px}} \langle Q(\mathbf{x}, t) \bar{\psi} \gamma_5 \psi(0) \rangle \xrightarrow[t \rightarrow \infty]{Z}{2E} \langle 0 | Q(0) | \pi(p) \rangle e^{-Et} \quad (7)$$

where  $Z = \langle \pi(p) | \bar{\psi} \gamma_5 \psi(0) | 0 \rangle$ .

For large time separations:

$$\begin{aligned} R_1 &= \frac{C_2^O(t)}{C_2^A(t)} \Big|_{\mathbf{p}=(1,1,0)} - 2 \frac{C_2^O(t)}{C_2^A(t)} \Big|_{\mathbf{p}=(1,0,0)} \\ &= p_1 p_2 \langle \xi^2 \rangle \\ R_2 &= C_2^{O'}(t)/C_2^A(t) \\ &= p_1^2 \langle \xi^2 \rangle \end{aligned}$$

Only the results for the heavier  $\kappa$  are presented here. The pion propagator is shown in Fig. 2. The effective mass and single-exponential fits give consistent results for  $6 < t < 15$ . Since the relative error on the signal of interest for the second moment becomes large for  $t > 15$ , the fitting ranges chosen in this work are shorter than the ones usually used for light spectroscopy. Despite this discrepancy, our value for  $f_\pi$  agrees with other UKQCD determinations from the same set of configurations within  $2\sigma$ .

The time dependence of  $R_2$ , obtained from 129 configurations, is given in Fig. 3. We have currently analysed  $R_1$  on 55 configurations only. In order to compare the two determinations, the results for  $R_1$  are also displayed in the same figure. The agreement between the two sets of results is good. At this preliminary stage  $R_2$  alone is used to extract our final number.

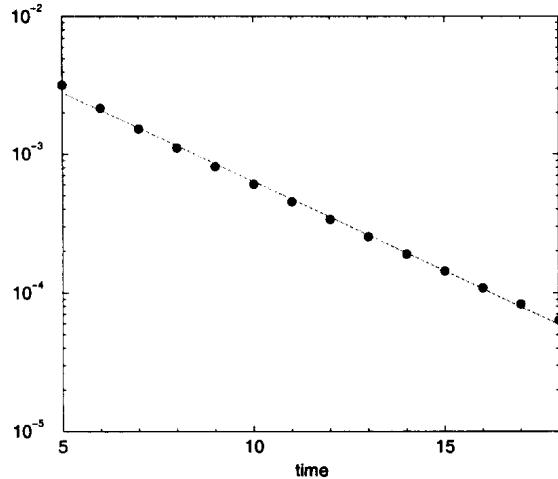


Figure 2. Pion propagator from 129 configurations at  $\kappa = 0.13460$

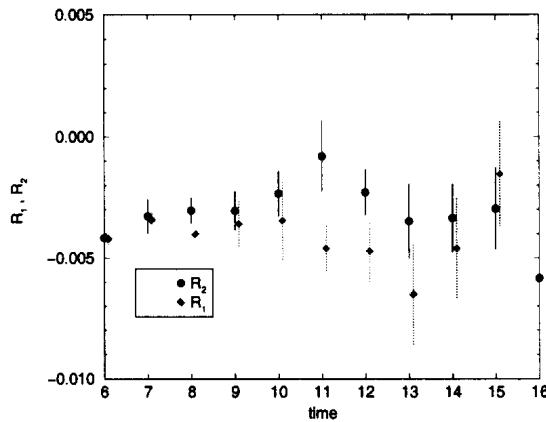


Figure 3. Ratios of two-point functions  $R_1$  and  $R_2$  for  $\kappa = 0.13460$ . The results for  $R_1$  and  $R_2$  are obtained from 55 and 129 configurations respectively.

## 5. CONCLUSIONS

Fitting the points in Fig. 3 to a constant yields

$$\langle \xi^2 \rangle_L = 0.06 \pm 0.02$$

for the lattice value of the second moment of the pion distribution amplitude. The analysis of the complete set of available configurations for both  $R_1$  and  $R_2$ , together with a full discussion of the systematic errors, will be presented in a forthcoming publication [6]. The statistical error in our result will of course decrease as the number of configurations is increased. The multi-

plicative renormalisation of the operator has yet to be calculated, either perturbatively or non-perturbatively.

Our preliminary result for the value of  $\langle \xi^2 \rangle$  suggests that it is lower than the value predicted by sum rules. Such a result could be explained for example by a more peaked distribution for  $\xi$  around the origin.

Computations of the distribution amplitudes of the  $\rho$  meson and the proton are currently under way. Our analysis can also be extended to the case of non-degenerate quarks and so a study of the distribution amplitudes of the  $K$  and  $B$  mesons is feasible.

## REFERENCES

1. for a review and extensive references, see V.L. Chernyak, I.R. Zhitnitsky, Phys. Rep. **112** (1984) 173
2. S. Brodsky et al., Phys. Lett. **91 B** (1980) 239
3. G. Martinelli and C. Sachrajda, Phys. Lett. **B 190** (1987) 151
4. T. Daniel, R. Gupta, D. Richards, Phys. Rev. **D 43** 3715 (1991)
5. J. Mandula, G. Zweig, J. Govaerts, Nucl. Phys. **B 228** (1980) 91
6. L. Del Debbio, M. Di Pierro, A. Dougall, C.T. Sachrajda, in preparation

# Excited heavy-light systems and hadronic transitions

M. Di Pierro and E. Eichten

Fermilab, Batavia, Illinois 60510

(Received 23 April 2001; published 31 October 2001)

A detailed study of orbital and radial excited states in  $D, D_s, B$ , and  $B_s$  systems is performed. The chiral quark model provides the framework for the calculation of pseudoscalar meson ( $\pi, K, \dots$ ) hadronic transitions among heavy-light excited and ground states. To calculate the excited states masses and wave functions, we must resort to a relativistic quark model. Our model includes the leading order corrections in  $1/m_{(c,b)}$  (e.g., mixing). Numerical results for masses and light hadronic transition rates are compared to existing experimental data. The effective coupling of the chiral quark model can be determined by comparing with independent results from lattice simulations ( $g_A^8 = 0.53 \pm 0.11$ ) or fitting to known widths ( $g_A^8 = 0.82 \pm 0.09$ ).

DOI: 10.1103/PhysRevD.64.114004

PACS number(s): 12.39.Ki, 12.38.Gc

## I. INTRODUCTION

Although heavy quark spectroscopy is now a rather mature subject, a number of interesting issues remain. In particular, the detailed properties of the excitation spectrum of heavy-light mesons ( $D, D_s, B, B_s$ ) and their light hadronic transitions are yet to be fully understood. Experimentally, much of this excitation spectrum remains to be observed. Only the ground state  $S$  waves and a few of the  $j_l=3/2$   $P$  waves are presently well established. However, many of these states will be accessible in the present and future  $B$  factories: CLEO, BaBar, Belle, Collider Detector at Fermilab (CDF), D0, BTeV, and CERN Large Hadron Collider (LHC B). In addition to furthering our understanding of QCD dynamics, the detailed study of these excited states may have practical benefits. For example, triggering on excited states may provide an efficient method of same side  $B$  tagging in hadron colliders. Tagging is essential to the study of  $CP$  violation in the  $B$  system.

The theoretical tools available to determine the properties of excited states in heavy-light systems include heavy quark effective theory (HQET) and low energy chiral effective theory [1]. Unfortunately, these tools are not sufficient to determine the detailed properties of these states. Lattice gauge theory is the only existing technique that allows the systematic study of all the aspects of QCD in heavy-light systems. Detailed studies of the  $P$ -wave excited heavy-light states within the quenched approximation already exist [2]. Future lattice studies will provide more insight into the nature of QCD dynamics as well as the masses and static properties of heavy-light hadrons.

It is clear that a model to estimate the hadronic transitions from excited state to ground states would also be very useful. Such a formalism has been developed and applied extensively to transitions in heavy-heavy ( $\bar{Q}Q'$ ) systems [3]. However, heavy-light mesons are more difficult because the light quark is subject to the full nonperturbative QCD dynamics. One possible approach to providing a framework for these hadronic transitions is to use the chiral quark model [4]. This has been suggested by Goity and Roberts [5].

In this paper we closely follow the work done in Refs. [5–7]. To compute the masses and wave functions of the

excited states we use a Dirac equation for the light quark in the potential generated by the heavy quark (including first order corrections in the heavy quark expansion and mixing effects). We then use these masses and wave functions to compute the hadronic decay amplitudes of excited heavy mesons in the context of a chiral quark model [4].

The main differences between the present and preceding works are in the choice for the parameters of the chromoelectric potential and in the inclusion of mixing effects both in the spectrum and in the decay amplitudes. Moreover, we use our results for the radial wave functions of the excited mesons to make a comparison with recent lattice results. From the comparison we extract an estimation for  $g_A^8$ , the effective coupling of the quark to the pseudoscalar mesons. We find  $g_A^8 = 0.53 \pm 0.11$ .

We present numerical results for the low-lying spectrum (excited states up to the  $3S$  states). We also compute the pseudoscalar meson hadronic transitions for these states as a function of the chiral quark model effective coupling constant. Comparing our results with recent experimental width measurements we estimate this effective coupling  $g_A^8 = 0.82 \pm 0.09$ .

In Sec. II we discuss our determination of the spectrum of excited states. Our notation, the choice of the potential, inclusion of mixing and other order  $1/m_h$  corrections are explained. Details of the masses and wave functions are presented for the low-lying excitation spectrum. A comparison is made with the present experimental data. Our treatment of hadronic decays is described in Sec. III. The analytic results are summarized in Eqs. (31)–(33). Explicit expressions for the coupling coefficients appearing in these equations are given in Appendix A. Also in Sec. III, details of the partial rates for the  $1S$  and  $1P$  states in the  $D, D_s, B$  and  $B_s$  systems are presented. Again comparison is made with the present experimental data. A complete list of the remaining results for masses and partial decay widths is reported in Appendix B.

## II. SPECTRUM

### A. Basic model and notation

The general Hamiltonian of the heavy-light system can be expanded in powers of  $(1/m_h)$

$$\mathcal{H} = \mathcal{H}^{(0)} + \frac{1}{m_h} \mathcal{H}^{(1)} + \frac{1}{m_h^2} \mathcal{H}^{(2)} + \dots \quad (1)$$

However, even within the heavy quark limit, the general form of the zeroth order Hamiltonian,  $\mathcal{H}^0$ , still involves the full nonperturbative QCD dynamics for the remaining degrees of freedom (including light quark pair creation and gluonic degrees of freedom). At present it cannot be solved analytically. We are forced to resort to use a relativistic potential model for  $\mathcal{H}^0$ .

We model the most general heavy-light meson (in the  $D, D_s, B, B_s$  family),  $H$ , as a bound state of a light quark ( $q$ ) and a heavy quark ( $h$ ). The heavy quark is treated as a static source of chromoelectric field and the only quantum number associated with it is its spin. The light quark is treated relativistically and its state is described by the wave function  $\psi_{n,l,j,m}(r, \theta, \varphi)$ . In analogy with the hydrogen atom, we introduce the following quantum numbers:

- $n$ , the number associated with the radial excitations;
- $l$ , the orbital angular momentum;
- $j$ , the total angular momentum of the light quark;
- $m$ , the component of  $j$  along the  $\hat{z}$  axis;
- $J$ , the total angular momentum of the system;
- $M$ , the component of  $J$  along the  $\hat{z}$  axis;
- $S$ , the spin of the heavy quark along the  $\hat{z}$  axis.

The parameters of our model are the masses of the light quarks ( $m_q$  for  $q=u,d$  or  $s$ ), the masses of the heavy quarks ( $m_h$  for  $h=c$  or  $b$ ) and the chromoelectric potential of the heavy quark [ $V(r)$ ].

The total wave function of the system can be decomposed as follows:

$$\Psi_{n,l,j,J,M}(r, \theta, \varphi) = \sum_{S \in \{-1/2, +1/2\}} C_{j,m;1/2,S}^{J,M} \psi_{n,l,j,m}(r, \theta, \varphi) \otimes \xi_S \quad (2)$$

where  $C_{j,m;1/2,S}^{J,M}$  are the usual Clebsch-Gordan coefficients and  $\xi_S$  is a two component spinor representing the heavy quark. Equation (2) is a solution of the following eigenvalue problem:

$$\mathcal{H} \Psi_{n,l,j,J,M} = E_{n,l,j,J} \Psi_{n,l,j,J,M} \quad (3)$$

where  $\mathcal{H}$  is the Hamiltonian of the system. The energy levels in Eq. (3) do not depend on  $M$  because of rotational invariance.

We rewrite Eq. (2) introducing the most general parametrization for the four spin components of the light quark wave function

$$\begin{aligned} \Psi_{n,l,j,J,M}(r, \theta, \varphi) &= \sum_{S \in \{-1/2, +1/2\}} C_{j,m;1/2,S}^{J,M} \\ &\times \begin{pmatrix} if_{n,l,j}^0(r) k_{l,j,m}^+ Y_{m-1/2}^l(\theta, \varphi) \\ if_{n,l,j}^0(r) k_{l,j,m}^- Y_{m+1/2}^l(\theta, \varphi) \\ f_{n,l,j}^1(r) k_{2j-l,j,m}^+ Y_{m-1/2}^{2j-l}(\theta, \varphi) \\ f_{n,l,j}^1(r) k_{2j-l,j,m}^- Y_{m+1/2}^{2j-l}(\theta, \varphi) \end{pmatrix} \otimes \xi_S. \end{aligned} \quad (4)$$

Here  $Y_m^l(\theta, \varphi)$  are spherical harmonics that encode the angular dependence while  $f_{n,l,j}^0(r)$ ,  $f_{n,l,j}^1(r)$  are real functions that encode that radial dependence.  $k_{l,j,m}^+$  and  $k_{l,j,m}^-$  are fixed, up to an overall phase, by imposing a normalization condition. Our choice of the phase is such that

$$k_{l,j,m}^\pm = \begin{cases} + \sqrt{\frac{l \pm m + 1}{2l+1}} & \text{for } j = l + \frac{1}{2}, \\ \pm \sqrt{\frac{l \mp m + 1}{2l+1}} & \text{for } j = l - \frac{1}{2}. \end{cases} \quad (5)$$

## B. Choice of the potential

Within our basic framework,  $\mathcal{H}^{(0)}$  is given by the relativistic Dirac Hamiltonian

$$\mathcal{H}^{(0)} = \gamma^0(-i\partial + m_q) + V(r) \quad (6)$$

and the rotational-invariant potential is the sum of a constant factor ( $M_h$ ), a scalar part ( $V_s$ ) and (the zeroth component of) a vector part ( $V_v$ )

$$V(r) = M_h + \gamma^0 V_s(r) + V_v(r). \quad (7)$$

The constant  $M_h$  is an overall energy shift that depends on the heavy quark flavor and, in general, it is not equal to  $m_h$ , as often assumed in the literature. For this reason we consider  $m_h$  and  $M_h$  two independent parameters of the model.

Asymptotic freedom suggests that at short distances the potential is dominated by a vector part that asymptotically approaches a Coulomb potential,  $V \sim V_v \sim 1/r$ . On the other hand, lattice simulations indicate that at large distances the potential is confining, scalar and asymptotically linear,  $V \sim V_s \sim r$ .

The naive assumption about a short distance Coulomb-like divergent behavior of the potential is doomed because it gives rise to ultraviolet divergences (as discussed in Ref. [8] and Ref. [9]). In this context the divergence arises in the  $1/m_h$  correction to the energy and it is due to the inconsistency of a static point-like source (the heavy quark) within a relativistic framework. One solution is assuming that the heavy quark is static but not point-like, therefore the potential that it generates is a convolution of the Coulomb-like

potential and the square of the heavy quark wave function (peaked around the center of mass of the system and smeared within some small length scale  $\lambda^{-1}$ ).

More generally, one is allowed to cure this divergence by regulating the potential close to the origin (on a length scale of the order  $\lambda^{-1}$ ). Different choices for the regulator are allowed and they do not affect the physics we want to describe, providing that  $\lambda^{-1}$  is small enough. The values of the parameters that appear in the Hamiltonian, on the contrary, depend on this choice since they run with  $\lambda$ . In fact, to obtain the same spectrum, different choices for the regulator imply different fitting parameters.

We chose to regulate the vector potential by assuming a Gaussian shape for the wave function of the heavy quark,  $\Phi(x) = \exp(-x^2\lambda^2/2)$ , and with this choice

$$V_v(r) = -\frac{4}{3} \int |\Phi(x)|^2 \frac{\alpha_s}{|\mathbf{r}-\mathbf{x}|} d^3x = -\frac{4}{3} \frac{\alpha_s}{r} \text{erf}(\lambda r). \quad (8)$$

For the scalar potential we assume a simple linear form

$$V_s(r) = br + c. \quad (9)$$

We observe that  $c$  is not a physical parameter since it can be absorbed into the definition of  $m_q$ . For this reason  $c$  will be omitted from now on.

Summarizing, the nine parameters of our model are

$$\alpha_s, \lambda, b, m_u, m_s, m_c, M_c, m_b, M_b \quad (10)$$

where  $m_u \equiv m_d$  and  $m_s$  are mass parameters for the light  $u, d$  and  $s$  quarks, respectively, equivalent to constituent quark masses shifted by the constant amount  $c$  of Eq. (9), which is undetermined in our model.  $m_c$  is the mass of the  $c$  quark with  $M_c$  the corresponding energy shift and analogously for the  $b$  quark.

### C. $1/m_h$ correction

For any given set of input parameters we solve the eigenvalue problem, Eq. (3), using the Hamiltonian of Eq. (6) and the potential specified by Eqs. (7), (8) and (9). In this way we determine the radial wave functions  $f_{n,l,j}^0$  and  $f_{n,l,j}^1$  associated with the energy levels  $E_{n,l,j}^{(0)}$ . We then compute  $1/m_h$  corrections to the energy levels in first order perturbation theory

$$E_{n,l,j,J} = E_{n,l,j}^{(0)} + \frac{1}{m_h} \delta E_{n,l,j,J}^{(1)} \quad (11)$$

where, ignoring for the moment the mixing of the states,

$$\delta E_{n,l,j,J}^{(1)} = \sum_M \int \Psi_{n,l,j,J,M}^\dagger(x) \mathcal{H}^{(1)} \Psi_{n,l,j,J,M}(x) d^3x. \quad (12)$$

The analytical expression for  $\mathcal{H}^{(1)}$  has been derived in Ref. [6] using the Bethe-Salpeter formalism. In terms of the radial wave functions (after the analytical integration of the angular part), we rewrite  $\delta E^{(1)}$  as a sum of three contributions

$$\delta E_{n,l,j,J}^{(1)} = A_{n,l,j,J} + B_{n,l,j,J} + C_{n,l,j,J}. \quad (13)$$

These terms are, respectively, (i) the kinetic energy

$$A_{n,l,j,J} = -\frac{1}{2} \int_0^\infty \left[ f_{n,l,j}^0 \left( \partial_r^2 + \frac{2}{r} \partial_r - \frac{l^2 + l}{r^2} \right) f_{n,l,j}^0 \right] r^2 dr \quad (14)$$

$$\times f_{n,l,j}^1 \left( \partial_r^2 + \frac{2}{r} \partial_r - \frac{\bar{l}^2 + \bar{l}}{r^2} \right) f_{n,l,j}^1 \right] r^2 dr \quad (15)$$

with  $\bar{l} = 2j - l$ . (ii) A shift due to spin-orbit interaction

$$B_{n,l,j,J} = \int_0^\infty V_v \left[ f_{n,l,j}^1 \left( \partial_r - \frac{l}{r} \right) f_{n,l,j}^0 - f_{n,l,j}^0 \times \left( \partial_r + \frac{l+2}{r} \right) f_{n,l,j}^1 \right] r^2 dr \quad (16)$$

for  $j = l + \frac{1}{2}$ , or

$$B_{n,l,j,J} = \int_0^\infty V_v \left[ f_{n,l,j}^1 \left( \partial_r + \frac{l+1}{r} \right) f_{n,l,j}^0 - f_{n,l,j}^0 \times \left( \partial_r - \frac{l-1}{r} \right) f_{n,l,j}^1 \right] r^2 dr \quad (17)$$

for  $j = l - \frac{1}{2}$ . (iii) The hyperfine splitting

$$C_{n,l,j,J} = (-1)^{J-l} \frac{2j+1}{2J+1} \int_0^\infty (\partial_r V_v) f_{n,l,j}^0 f_{n,l,j}^1 r^2 dr. \quad (18)$$

### D. Mixing

In Eqs. (11) and (12) we assumed that the Hamiltonian was diagonal. This is not the case because the  $1/m_h$  interaction term in the Hamiltonian mixes states. In general correction terms can mix any states with the same total angular momentum,  $J$ , and parity,  $P$ . However, there are only two types of sizable mixings. Large mixing can occur for pairs of states  $\Psi_{n,l,j,J,M}$  and  $\Psi_{n',l',j',J,M}$  with (1)  $n = n'$ ,  $l = l'$  and  $j + \frac{1}{2} = j' - \frac{1}{2} = l = J$  (i.e., mixing within a given  $n$  and  $l$  multiplet) or (2)  $n+1 = n'$ ,  $l+2 = l'$  and  $j + \frac{1}{2} = j' - \frac{1}{2} = l = J$  (e.g.,  $S$ - $D$  mixing). The off-diagonal term in the Hamiltonian that mixes such pairs of states has the form

TABLE I. Tabulated spectrum for  $D$  mesons.  $E^0$  denotes the lowest order energies.  $E^{\text{phys.}}$  includes all the order  $1/m_h$  corrections. (All units are in GeV.) The mixings between lowest order states are denoted by  $\phi$ .

$H(n^j L_J)$	$m_{\text{expt.}}$	$E^0$	$E^{\text{phys.}}$	$\phi$ (%)
$D(1^{1/2}S_0)$	1.865	1.895	1.868	
$D(1^{1/2}S_1)$	2.007	1.895	2.005	
$D(1^{1/2}P_0)$		2.282	2.377	
$D(1^{3/2}P_1)$	2.422	2.253	2.417	-10.92
$D(1^{3/2}P_2)$	2.459	2.253	2.460	
$D(1^{1/2}P_1)$		2.282	2.490	10.92
$D(2^{1/2}S_0)$		2.447	2.589	
$D(2^{1/2}S_1)$		2.447	2.692	2.17
$D(1^{5/2}D_2)$		2.504	2.775	-5.41
$D(1^{3/2}D_1)$		2.553	2.795	-2.17
$D(1^{5/2}D_3)$		2.504	2.799	
$D(1^{3/2}D_2)$		2.553	2.833	5.41
$D(2^{1/2}P_0)$		2.683	2.949	
$D(2^{3/2}P_1)$		2.679	2.995	-10.70
$D(2^{3/2}P_2)$		2.679	3.035	1.79
$D(2^{1/2}P_1)$		2.683	3.045	10.70
$D(1^{7/2}F_3)$		2.709	3.074	-3.17
$D(1^{7/2}F_4)$		2.709	3.091	
$D(1^{5/2}F_2)$		2.760	3.101	-1.79
$D(1^{5/2}F_3)$		2.760	3.123	3.17
$D(3^{1/2}S_0)$		2.823	3.141	
$D(3^{1/2}S_1)$		2.823	3.226	

$$\begin{aligned} \epsilon &= \frac{1}{m_h} \sum_M \int \Psi_{n,l,j,J,M}^\dagger(x) \mathcal{H}^{(1)} \Psi_{n',l',j',J,M}(x) d^3x \\ &= (-1)^{J-l} \frac{1}{m_h} \frac{\sqrt{J(J+1)}}{2J+1} \int_0^\infty (\partial_r V_v) [f_{n,l,j}^0 f_{n',l',j'}^1 \\ &\quad + f_{n,l,j}^1 f_{n',l',j'}^0] r^2 dr \end{aligned} \quad (19)$$

and it induces a mixing in the wave functions and in the energy levels

$$\begin{pmatrix} \psi_{n,l,j,m} \\ \psi_{n',l',j',m} \end{pmatrix}^{\text{phys}} = \begin{pmatrix} 1 & +\frac{\epsilon}{2\Delta} \\ -\frac{\epsilon}{2\Delta} & 1 \end{pmatrix} \begin{pmatrix} \psi_{n,l,j,m} \\ \psi_{n',l',j',m} \end{pmatrix} + \mathcal{O}(\epsilon^2) \quad (20)$$

$$\begin{pmatrix} E_{n,l,j,J} \\ E_{n',l',j',J} \end{pmatrix}^{\text{phys}} = \begin{pmatrix} E_{n,l,j,J} + \frac{\epsilon^2}{2\Delta} \\ E_{n',l',j',J} - \frac{\epsilon^2}{2\Delta} \end{pmatrix} + \mathcal{O}(\epsilon^2) \quad (21)$$

with  $\Delta = (E_{n,l,j,J} - E_{n',l',j',J})/2$ .

We find that the effect of mixing is generally negligible except for the  $P$  waves, where the mixing among wave functions can be of the order of 10%. In Tables I–IV we report the value of  $\phi = (100/2)\epsilon/\Delta$  for each excited state. It measures, in percent, the contribution of the mixing to the wave function.

### E. Determination of the parameters and predictions

The nine parameters of our model, Eq. (10), are determined numerically as follows: We define a function  $\mathcal{F}$  of the input parameters that finds eigenvalues and eigenfunctions of Eq. (3) using a fourth order Runge-Kutta formula, corrects the energy levels by including the  $1/m_h$  perturbative corrections (including mixing effects) and returns

$$\chi^2 = \sum_{\text{observed states}} \left( \frac{E_{n,l,j,J}^{\text{phys.}} - m_{n,l,j,J}}{\delta m_{n,l,j,J}} \right)^2 \quad (22)$$

where  $E_{n,l,j,J}^{\text{phys.}}$  are the computed energy levels and  $m_{n,l,j,J} \pm \delta m$  are the measured masses (with their experimental error) of the corresponding particles.

We then minimize  $\mathcal{F}$  in its nine dimensional domain. We repeat this procedure with different sets of starting parameters until we are confident that we have found the absolute minimum. The experimental data used for the “observed states” in the fit are reported in the third column of Tables I–IV.

Our best fit gives the following values for the parameters:

$$\begin{aligned} \alpha_s &= 0.339 \\ \lambda &= 2.823 \text{ GeV} \\ b &= 0.257 \text{ GeV}^2 \\ m_u &= 0.071 \text{ GeV} \\ m_s &= 0.216 \text{ GeV} \\ m_c &= 1.511 \text{ GeV} \\ M_c &= 1.292 \text{ GeV} \\ m_b &= 4.655 \text{ GeV} \\ M_b &= 4.685 \text{ GeV}. \end{aligned} \quad (23)$$

The corresponding predicted spectrum is reported in Fig. 1 and in the fifth column of Tables I–IV. The best fit parameters reported here differ slightly from those reported in Ref. [10] because we use here most recent value for  $m_{n,l,j,J}$ .

TABLE II. Tabulated spectrum for  $D_s$  mesons. (All units are in GeV.) The notation is as in Table I.

$H(n^jL_J)$	$m_{\text{expt.}}$	$E^0$	$E^{\text{phys.}}$	$\phi$ (%)
$D_s(1^{1/2}S_0)$	1.969	1.988	1.965	
$D_s(1^{1/2}S_1)$	2.112	1.988	2.113	
$D_s(1^{1/2}P_0)$		2.374	2.487	
$D_s(1^{3/2}P_1)$	2.535	2.353	2.535	-11.62
$D_s(1^{3/2}P_2)$	2.573	2.353	2.581	
$D_s(1^{1/2}P_1)$		2.374	2.605	11.62
$D_s(2^{1/2}S_0)$		2.540	2.700	
$D_s(2^{1/2}S_1)$		2.540	2.806	1.97
$D_s(1^{5/2}D_2)$		2.606	2.900	-6.11
$D_s(1^{3/2}D_1)$		2.648	2.913	-1.97
$D_s(1^{5/2}D_3)$		2.606	2.925	
$D_s(1^{3/2}D_2)$		2.648	2.953	6.11
$D_s(2^{1/2}P_0)$		2.777	3.067	
$D_s(2^{3/2}P_1)$		2.775	3.114	-10.58
$D_s(2^{3/2}P_2)$		2.775	3.157	1.81
$D_s(2^{1/2}P_1)$		2.777	3.165	10.58
$D_s(1^{7/2}F_3)$		2.812	3.203	-3.60
$D_s(1^{7/2}F_4)$		2.812	3.220	
$D_s(1^{5/2}F_2)$		2.857	3.224	-1.81
$D_s(1^{5/2}F_3)$		2.857	3.247	3.60
$D_s(3^{1/2}S_0)$		2.917	3.259	
$D_s(3^{1/2}S_1)$		2.917	3.345	

TABLE III. Tabulated spectrum for  $B$  mesons. (All units are in GeV.) The notation is as in Table I.

$H(n^jL_J)$	$m_{\text{expt.}}$	$E^0$	$E^{\text{phys.}}$	$\phi$ (%)
$B(1^{1/2}S_0)$	5.279	5.288	5.279	
$B(1^{1/2}S_1)$	5.325	5.288	5.324	
$B(1^{3/2}P_1)$		5.646	5.700	-6.00
$B(1^{1/2}P_0)$		5.675	5.706	
$B(1^{3/2}P_2)$		5.646	5.714	
$B(1^{1/2}P_1)$		5.675	5.742	6.00
$B(2^{1/2}S_0)$		5.840	5.886	
$B(2^{1/2}S_1)$		5.840	5.920	0.69
$B(1^{5/2}D_2)$		5.897	5.985	-1.96
$B(1^{5/2}D_3)$		5.897	5.993	
$B(1^{3/2}D_1)$		5.946	6.025	-0.69
$B(1^{3/2}D_2)$		5.946	6.037	1.96
$B(2^{1/2}P_0)$		6.076	6.163	
$B(2^{3/2}P_1)$		6.072	6.175	-9.11
$B(2^{3/2}P_2)$		6.072	6.188	0.50
$B(2^{1/2}P_1)$		6.076	6.194	9.11
$B(1^{7/2}F_3)$		6.102	6.220	-0.99
$B(1^{7/2}F_4)$		6.102	6.226	
$B(1^{5/2}F_2)$		6.153	6.264	-0.50
$B(1^{5/2}F_3)$		6.153	6.271	0.99
$B(3^{1/2}S_0)$		6.216	6.320	
$B(3^{1/2}S_1)$		6.216	6.347	

TABLE IV. Tabulated spectrum for  $B_s$  mesons. (All units are in GeV.) The notation is as in Table I.

$H(n^jL_J)$	$m_{\text{expt.}}$	$E^0$	$E^{\text{phys.}}$	$\phi$ (%)
$B_s(1^{1/2}S_0)$	5.369	5.381	5.373	
$B_s(1^{1/2}S_1)$	5.417	5.381	5.421	
$B_s(1^{1/2}P_0)$		5.767	5.804	
$B_s(1^{3/2}P_1)$		5.746	5.805	-7.19
$B_s(1^{3/2}P_2)$		5.746	5.820	
$B_s(1^{1/2}P_1)$		5.767	5.842	7.19
$B_s(2^{1/2}S_0)$		5.933	5.985	
$B_s(2^{1/2}S_1)$		5.933	6.019	0.64
$B_s(1^{5/2}D_2)$		5.999	6.095	-2.31
$B_s(1^{5/2}D_3)$		5.999	6.103	
$B_s(1^{3/2}D_1)$		6.041	6.127	-0.64
$B_s(1^{3/2}D_2)$		6.041	6.140	2.31
$B_s(2^{1/2}P_0)$		6.170	6.264	
$B_s(2^{3/2}P_1)$		6.168	6.278	-9.81
$B_s(2^{3/2}P_2)$		6.168	6.292	0.51
$B_s(2^{1/2}P_1)$		6.170	6.296	9.81
$B_s(1^{7/2}F_3)$		6.205	6.332	-1.15
$B_s(1^{7/2}F_4)$		6.205	6.337	
$B_s(1^{5/2}F_2)$		6.250	6.369	-0.51
$B_s(1^{5/2}F_3)$		6.250	6.376	1.15
$B_s(3^{1/2}S_0)$		6.310	6.421	
$B_s(3^{1/2}S_1)$		6.310	6.449	

We remark that  $m_u$  and  $m_s$  are mass parameters that differ from the constituent quark masses for an overall undetermined constant shift.

As a consistency check of our results we observe that the mass splitting  $m_s - m_u \approx 140$  MeV comes out in agreement with naive expectations based on Gell-Mann-Okubo type relations. This difference also agrees, within 1%, with the corresponding splitting determined in Ref. [6] and used in Ref. [5] as input for their calculations.

Remarkably  $m_b \approx M_b$  with much better agreement than expected. Moreover,  $\lambda^{-1} \approx 0.06$  fm is smaller than any other length scale involved in the problem, as was required.

Figure 2 shows some of the computed radial wave functions for non-strange mesons,  $f_{n,l,j}^0(r)$  and  $f_{n,l,j}^1(r)$ . These wave functions do not include  $1/m_h$  corrections, therefore they are the same for  $D$  and  $B$  mesons. The corresponding wave functions for strange mesons are very similar.

Figure 3 shows density plots for each couple of independent spin components of some of the computed light-quark wave functions,  $|f_{n,l,j}^0 Y_0^l|^2$  in black and  $|f_{n,l,j}^1 Y_0^{2j-l}|^2$  in gray. They represent the analogous, in the heavy meson systems, of the orbitals of the hydrogen atom.

## F. Comparison with experiment

The comparison of our results to the present experimental information on the excitation spectrum of the  $(D, D_s, B, B_s)$  mesons is given in Table V. States which were used in the determining our best fit parameters are so indicated.

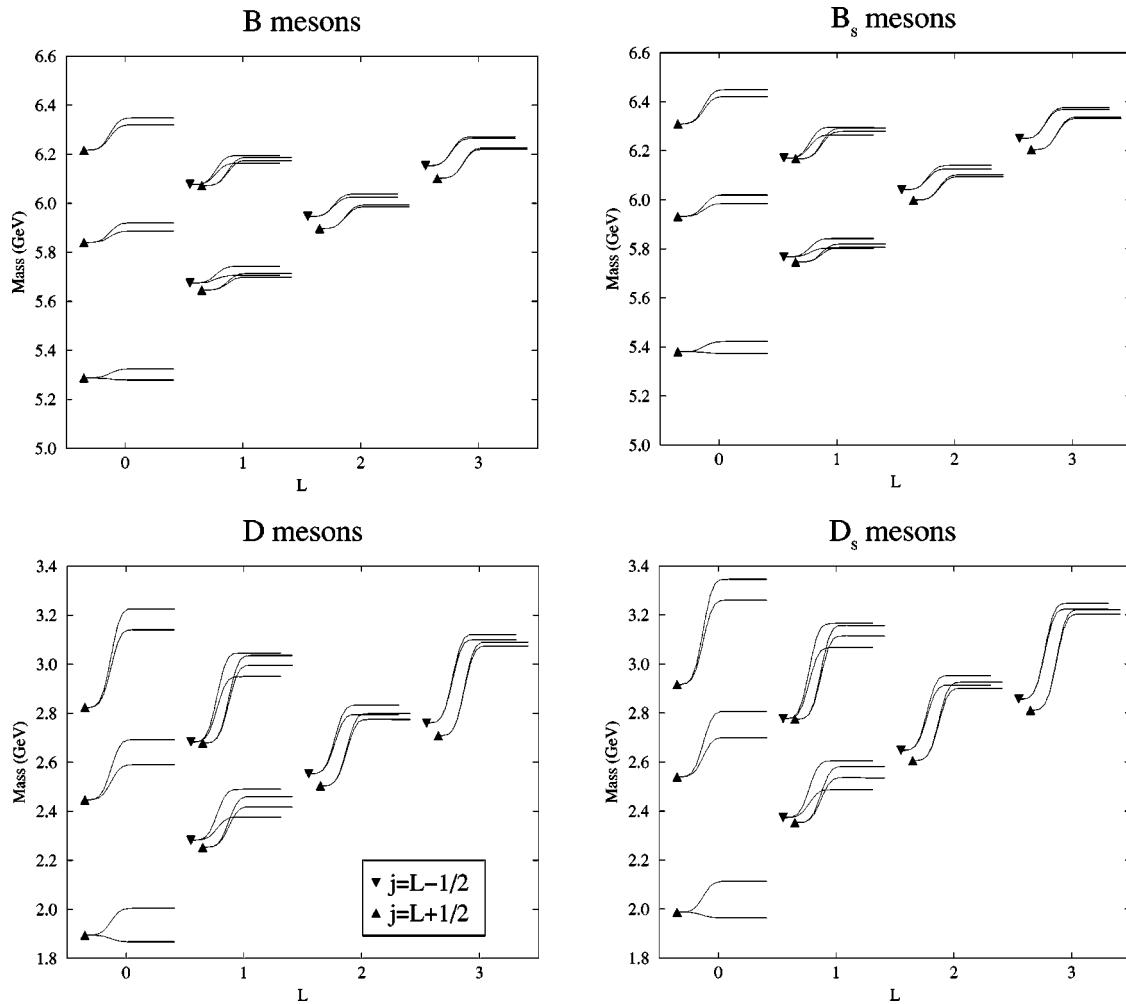


FIG. 1. Computed spectrum of excited states in the  $\{D, D_s, B, B_s\}$  family. The plot shows the spectrum before and after  $1/m_h$  corrections (including mixing). These corrections are responsible for the hyperfine splitting. The horizontal axis is the orbital angular momentum of the meson ( $L$ ). For each value of  $L$  and  $j$  there is a doublet of states ( $J=j-\frac{1}{2}$  and  $J=j+\frac{1}{2}$ , with lower and higher energy, respectively).

Our model is in excellent agreement with the better established  $P$  waves in the  $D$  and  $D_s$  systems. In particular the  $D_1^*$  ( $1^{1/2}P_1$ ) fits the recent measurement of CLEO [14]. For the  $P$ -waves of the  $B$  meson systems the agreement with preliminary measurements is somewhat less impressive.

However, many of the existing experimental fits for individual masses of these states relied on patterns of masses for the  $j_l=1/2$  states not found in our model. For example, our relativistic quark model predict  $m_{1,1,1/2,1} > m_{1,1,3/2,1}$  for a relatively broad range of parameters consistent with light spectroscopy.<sup>1</sup> Also, we obtain a splitting for the  $^{1/2}P_J$  states more than twice as big as the splitting for the  $^{3/2}P_J$  states.

Preliminary results from L3 [12] for the masses of  $P$ -wave  $B$  meson excitations are

$$B_1^*: m_{1,1,1/2,1} = (5.670 \pm 0.010_{\text{stat}} \pm 0.013_{\text{syst}}) \text{ GeV} \quad (L3) \quad (24)$$

$$B_2^*: m_{1,1,3/2,2} = (5.768 \pm 0.005_{\text{stat}} \pm 0.006_{\text{syst}}) \text{ GeV} \quad (L3). \quad (25)$$

The L3 results were derived using the constraint that  $m_{1,1,1/2,1} - m_{1,1,1/2,0} = m_{1,1,3/2,2} - m_{1,1,3/2,1} = 12$  MeV. This assumption is not realized in our model. Similar assumptions are needed for the extractions the masses of  $P$ -wave  $B$  meson excitations from OPAL [13] and CDF [18]. It would be interesting to reanalyze results using the pattern expected in this relativistic quark model. Finally, the observation of the  $D'^*$  by DELPHI [15] is not consistent with searches by CLEO [16] and OPAL [17].

### G. Regge trajectories

We find that  $E_{n,l,j,J} - M_h$  all lie on Regge trajectories parameterized by  $mJ + q_{n,l,j}$  with  $m \approx 0.7$  (as shown in Fig. 4). This is a well understood phenomenon for light spectroscopy

<sup>1</sup>This result is known in the literature as spin-orbit inversion. It was first predicted by Schnitzer [20] and later by the models of Isgur [21] and Ebert [22].

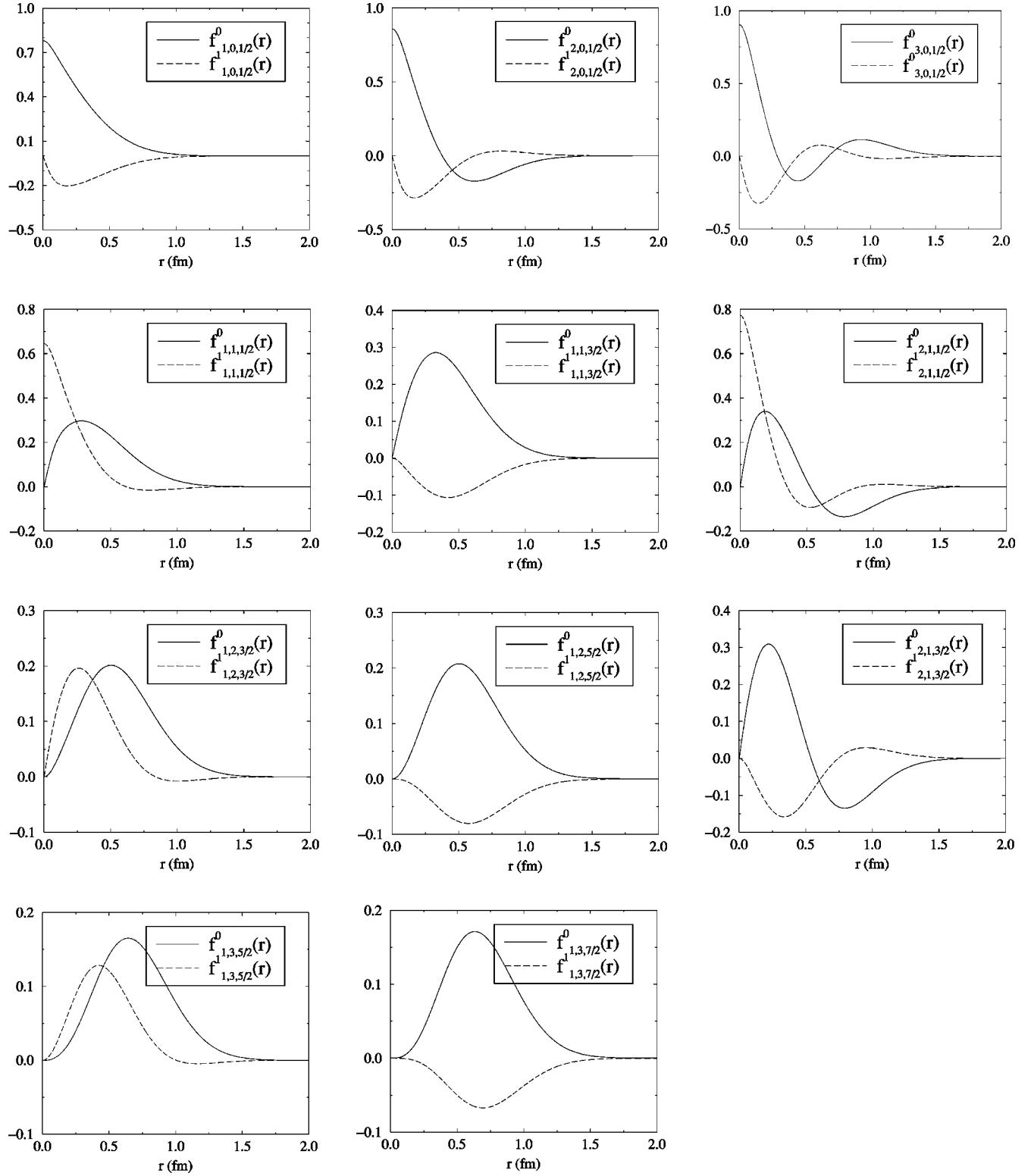
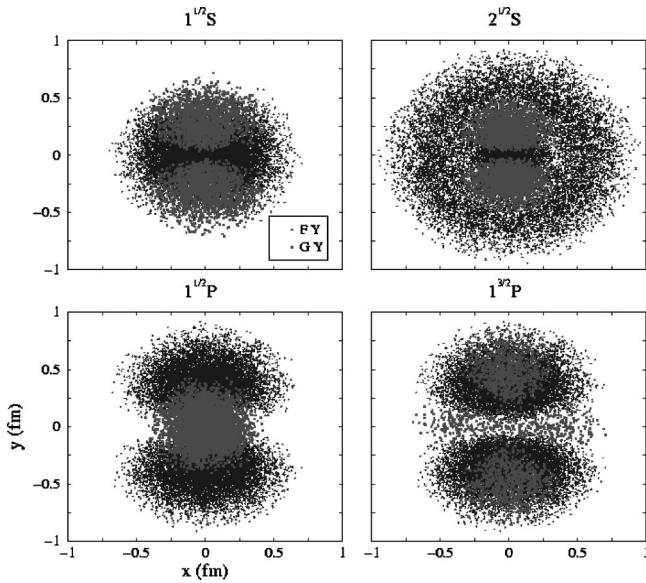


FIG. 2. Radial wave functions for some excited states (for non-strange mesons). The continuum (dashed) line refers to the  $f^0(r)$  ( $f^1(r)$ ) function. These plots do not include the mixing contribution.

but, for light mesons,  $m=2\pi b$  (where  $b$  is the string tension). This is about a factor two bigger than we find. Our result can be explained in the non-relativistic limit or, alternatively, in a simple and naive string picture: Ignoring the

short distance behavior of the potential, we model the meson as a classical light quark attached through a string to the center of mass of the system. The energy of the system,  $E$ , (i.e., the energy of the string) is related to the classical an-

FIG. 3. Orbitals for some excited  $B$  mesons.

gular momentum,  $J$ , by  $E^2 = \pi b J$ . The factor two in the light meson picture can be explained with the fact that the latter rotate around a center of mass that is located at the middle of the string while for heavy-light mesons the center of mass coincides with one of the two ends of the string.

TABLE V. The heavy-light spectrum compared to experiment. We report the difference between the excited state masses and the ground state ( $D$  or  $B$ ) in each case.

Charmed meson masses (MeV)			Bottom meson masses (MeV)				
Model	Expt.	[Ref.]	Model	Expt.	[Ref.]		
$D^* - D$	137 <sup>a</sup>	141,142	[11]	$B^* - B$	45 <sup>a</sup>	46	[11]
$D_0^* - D$	512			$B_0^* - B$	427		
$D_1^* - D$	622	596(53)	[14]	$B_1^* - B$	463	391(16)	[12] <sup>b</sup>
$D_1 - D$	549 <sup>a</sup>	558(2)	[11]	$B_1 - B$	421	459(9)	[13] <sup>b</sup>
$D_2^* - D$	592 <sup>a</sup>	594(2)	[11]	$B_2^* - B$	421	431(20)	[18] <sup>b</sup>
				$B_2^* - B$	435	489(8)	[12] <sup>b</sup>
						460(13)	[19] <sup>b</sup>
				$(B^{**} - B)$		418(9)	[11] <sup>c</sup>
$D' - D$	721			$B' - B$	607		
$D'^* - D$	824	772(6)	[15]	$B'^* - B$	641		
		not seen	[16,17]				
$D_s - D$	97 <sup>a</sup>	99,104	[11]	$B_s - B$	94 <sup>a</sup>	90(2)	[11]
$D_s^* - D_s$	148 <sup>a</sup>	144	[11]	$B_s^* - B_s$	48 <sup>a</sup>	46	[11]
$D_{s0}^* - D_s$	512			$B_{s0}^* - B_s$	431		
$D_{s1}^* - D_s$	640			$B_{s1}^* - B_s$	469		
$D_{s1} - D_s$	570 <sup>a</sup>	566(1)	[11]	$B_{s1} - B_s$	432		
$D_{s2}^* - D_s$	616 <sup>a</sup>	605(2)	[11]	$B_{s2}^* - B_s$	447		
				$(B_s^{**} - B_s)$		484(15)	[11] <sup>c</sup>
$D_s' - D_s$	735			$B_s' - B_s$	612		
$D_s'^* - D_s$	841			$B_s'^* - B_s$	646		

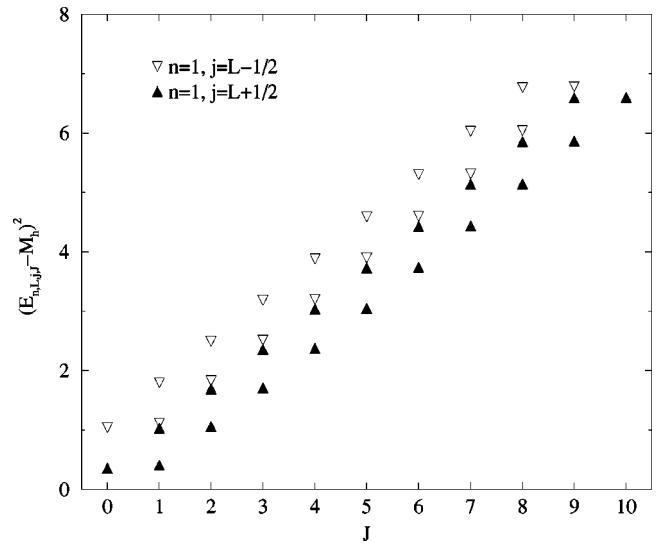
<sup>a</sup>Experimental input to model parameters fit.<sup>b</sup>Theoretical estimates for some of the mass splittings have been used as input.<sup>c</sup>Experimental signal is a sum over resonances with  $J=0,1,2$ .

FIG. 4. Regge trajectories for some of our orbitally excited heavy mesons.

This simple picture shows that the bulk of an heavy meson mass is dominated by the mass of the heavy quark plus the potential energy associated to the large distance interaction ( $\propto br$ ), and again gives support to our assumption that short distance behavior of the potential has a small contribution to the spectrum.

### III. HADRONIC TRANSITIONS

#### A. Transition amplitudes

We start by considering the most general hadronic transition of the form

$$H' \rightarrow H + x \quad (26)$$

where  $H'$  and  $H$  are two heavy-light mesons containing the same heavy quark, with wavefunctions  $\Psi_{n',l',j',J',M'}$  and  $\Psi_{n,l,j,J,M}$ , respectively, and  $x$  can be any light meson with momentum  $\mathbf{p}$ . Although we keep our formalism general, in this paper we only compute numerically decays in which  $x$  is a pseudoscalar meson belonging to the flavor octet ( $\pi$ ,  $K$ ,  $\eta$ ).

In the context of the chiral quark model [4] this transition is mediated by an effective interaction of the form

$$\mathcal{L}_{\text{int}} = \frac{g_A^8}{\sqrt{2}f_x} \bar{q}_i' X \mathcal{M}^{ij} q_j + \mathcal{O}(\partial^2) \quad (27)$$

where  $f_x$  can be identified with  $f_\pi \approx 130$  MeV and  $g_A^8$  is an effective coupling.  $X = \not{p}\gamma^5$  is the spin structure associated to the transition,  $i, j$  are SU(3)<sub>flavor</sub> indices and

$$\mathcal{M} = \sqrt{2} \begin{pmatrix} \frac{1}{\sqrt{2}}\pi^0 + \frac{1}{\sqrt{6}}\eta_8 & \pi^+ & K^+ \\ \pi^- & -\frac{1}{\sqrt{2}}\pi^0 + \frac{1}{\sqrt{6}}\eta_8 & K^0 \\ K^- & \bar{K}^0 & -\frac{2}{\sqrt{6}}\eta_8 \end{pmatrix} \quad (28)$$

is the usual  $SU(3)_{L+R}$  invariant representation of the pseudoscalar mesons.

The transition mediated by this Lagrangian is associated with the following matrix element:

$$I^{H'Hx}(\mathbf{p}) = \frac{i g_A^8 \zeta}{\sqrt{2}f_x} \int \bar{\Psi}_{n,l,j,J,M}(z) X e^{i\mathbf{p} \cdot \mathbf{z}} \Psi_{n',l',j',J',M'}(z) d^3 z \quad (29)$$

where  $\zeta$  is a coefficient that characterize the flavor structure of the decay. A list of all possible cases has been derived from Eq. (28) and is reported in Table VI. The physical  $\eta$  and  $\eta'$  are of course mixtures of the ideal  $\eta_8$  and  $\eta_1$ . In particular,  $\eta = \eta_8 \cos(\theta_p) - \eta_1 \sin(\theta_p)$  where  $\theta_p \approx -10.1^\circ$  with a large uncertainty. In this work we ignore this mixing and we assume  $\theta_p = 0$ . The corrective multiplicative factor for a different choice for  $\theta_p$  can be derived by the reader using Table VI.

The exponential in Eq. (29) can be expanded in products of spherical harmonics and spherical Bessel functions, thus giving

TABLE VI. List of decay channels for  $B$  (or  $D$ ) mesons with the corresponding flavor factor  $\zeta$ .

$H' \rightarrow H + x$	$\zeta$	$H' \rightarrow H + x$	$\zeta$
$B^0 \rightarrow B^0 + \pi^0$	1	$B_s \rightarrow B_s + \eta_8$	$-2/\sqrt{3}$
$B^\pm \rightarrow B^\pm + \pi^0$	1	$B_s \rightarrow B^0 + K$	$\sqrt{2}$
$B^\pm \rightarrow B^0 + \pi^\pm$	$\sqrt{2}$	$B_s \rightarrow B^\pm + K^\mp$	$\sqrt{2}$
$B^0 \rightarrow B^\pm + \pi^\mp$	$\sqrt{2}$	$B^0 \rightarrow B_s + \bar{K}$	$\sqrt{2}$
$B^0 \rightarrow B^0 + \eta_8$	$1/\sqrt{3}$	$B^\pm \rightarrow B_s + K^\pm$	$\sqrt{2}$
$B^\pm \rightarrow B^\pm + \eta_8$	$1/\sqrt{3}$	$B_q \rightarrow B_q + \eta_1$	$\sqrt{\frac{2}{3}} + \mathcal{O}\left(\frac{1}{N_c}\right)$

$$I^{H'Hx}(\mathbf{p}) = \sum_{l_x, m_x} Y_{m_x}^{l_x^*}(\hat{p}) C_{J,M;l_x,m_x}^{J',M'} \mathcal{A}_{l_x}^{H'Hx}(X,p). \quad (30)$$

Equation (30) implicitly defines the transition amplitude,  $\mathcal{A}_{l_x}^{H' \rightarrow Hx}(X,p)$ , for a  $x$  in a given eigenstate  $l_x$  of its angular momentum. By projecting the matrix  $X$  on the basis presented in the Appendix A, the transition amplitude can be rewritten as a linear combination of terms, each factorized into a radial part and a spin dependent part

$$\begin{aligned} \mathcal{A}_{l_x}^{H' Hx}(X,p) = & \frac{i g_A^8 \zeta}{\sqrt{2}f_x} \sum_{ab=\{0,1\}} \sum_k c_{l_x}^{ab,k}(X) \\ & \times \int_0^\infty f_{n',l',j'}^a(r) j_k(rp) f_{n,l,j}^b(r) r^2 dr. \end{aligned} \quad (31)$$

The coefficients  $c_{l_x}^{ab,k}(X)$  depend on the quantum numbers of the mother and the daughter heavy mesons. Their explicit expression is given in the Appendix A. The integrals are computed numerically.

One can extend our analysis for octet pseudoscalar transitions to the approximately flavor singlet  $\eta'$ . In the large  $N_c$  limit the  $\eta_1$  combines with the octet to form a nonet. In that case the effective interaction in the Lagrangian takes the form

$$\mathcal{L}_{\text{int}} = \frac{g_A^1 \zeta}{\sqrt{2}f_{\eta_1}} \bar{q}_i' X \eta_1 q_i + \mathcal{O}(\partial^2) \quad (32)$$

with  $X = \not{p}$  and  $g_A^1 \approx g_A^8$ . This symmetry is badly broken in QCD. However, it is reasonable to assume that in these transitions that the form [Eq. (32)] still holds. If one further assumes that the spatial wave functions of  $\pi$  and  $\eta'$  are approximately equal, one obtains  $f_{\eta'} \approx f_\pi$ . Hence the coefficient  $\zeta$  can be set to  $\sqrt{2/3}$  both for heavy strange and non-strange decaying heavy mesons.

The situation for decays in which  $x$  is a light vector mesons ( $\rho$ ,  $\omega$ ,  $K^*$ ) is different. When compared to the pseudoscalar mesons, they have a different spin coupling to the quarks ( $X = \not{k}$  where  $\epsilon_\mu$  is the polarization vector of the me-

TABLE VII. List of partial decay rates for  $1S$   $D$  mesons. The measured  $D^*$  and  $D$  masses [11] are used in these results.

Channel	$l_\pi$	$p_\pi$ (MeV)	$\Gamma_\pi/(g_A^8)^2$ (keV)
$D^{*0}(1^{1/2}S_1) \rightarrow D^0(1^{1/2}S_0) + \pi^0$	1	$42.8 \pm .2$	$62 \pm 1$
$D^{*+}(1^{1/2}S_1) \rightarrow D^0(1^{1/2}S_0) + \pi^+$	1	$39.4 \pm .5$	$97 \pm 3$
$D^{*+}(1^{1/2}S_1) \rightarrow D^+(1^{1/2}S_0) + \pi^0$	1	$38.1 \pm .3$	$44 \pm 1$

son), a different effective coupling ( $g_V \neq g_A^8$ ), and a different wave function ( $f_\rho \neq f_\pi$ ). With these replacements Eq. (31) remains valid for decays with emission of light vector mesons.<sup>2</sup> The detailed study of these vector meson transitions is deferred to a future paper.

### B. Partial widths

The partial width for the transition in Eq. (26) (for a light meson  $x$  emitted with total momentum  $p$  and angular momentum  $l_x$ ) is given by

$$\Gamma_x(H' \rightarrow H + x; l_x) = \frac{p}{8\pi^2} \frac{2J+1}{2J'+1} \frac{m_H}{m_{H'}} |\mathcal{A}_{l_x}^{H'Hx}(X, p)|^2 \quad (33)$$

where  $m_{H'}$  and  $m_H$  are the masses of the mother and daughter heavy mesons respectively.

The total hadronic decay width (via a pseudoscalar meson transition) is defined simply as the sum of the partial widths:

$$\Gamma_M^{H'} = \sum_H \sum_{x=\{\pi, \eta_8, K\}} \sum_{l_x} \Gamma_x(H' \rightarrow H + x; l_x). \quad (34)$$

Transitions involving excited states very near their kinematic threshold for an allowed decay (e.g. where the light pseudoscalar momentum is less than 100 MeV) are extremely sensitive to our calculated mass values. This is particularly true for the allowed transitions within the  $1S$  multiplets. In these cases, even the small mass differences between the charged and neutral states are important. Using the physical masses for the various  $D$  mesons [11], the individual pion transitions are shown in Table VII.

After removing these phase space uncertainties, reasonable variations in our model parameters gave variations of about 10% in the overall hadronic widths. The listed branching ratios with emission of a  $\pi$  are flavor blind and sum over the final state pion charge (i.e., they have been computed with  $\zeta = \sqrt{3}$ ). Each exclusive decay can be deduced by correcting for this factor using Table VI to determine the relative strength of the charged and neutral decays. In addition, a small phase space correction should be included appropriate to the slight difference in the masses of the various charge

<sup>2</sup>It is possible to relate these the pseudoscalar and vector couplings and coefficients within the context of an approximate  $SU(6)_W$  symmetry for the low-lying states.

TABLE VIII. The heavy-light  $1P$  state hadronic transition rates for  $D$  and  $D_s$  mesons.  $H' \rightarrow H + x$ . Decays denoted with an (\*) are allowed only because of the order  $1/m_h$  mixing of states.  $p_x$  and  $\Gamma_x/(g_A^8)^2$  are in MeV.

$H'(n'j'l_{J'})$	$H(njl_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$
$D(1^{1/2}P_0)$	$D(1^{1/2}S_0)$	$\pi$	0	437	189
$D(1^{3/2}P_1)$	$D(1^{1/2}S_1)$	$\pi$	0	355	(*) 1.7
	$D(1^{1/2}S_1)$	$\pi$	2	355	14.5
$D(1^{3/2}P_2)$	$D(1^{1/2}S_0)$	$\pi$	2	506	24.6
	$D(1^{1/2}S_1)$	$\pi$	2	394	13.7
$D(1^{1/2}P_1)$	$D(1^{1/2}S_1)$	$\pi$	0	420	181
$D_s(1^{1/2}P_0)$	$D(1^{1/2}S_0)$	$K$	0	325	236
$D_s(1^{3/2}P_1)$	$D(1^{1/2}S_1)$	$K$	0	175	(*) 1.89
	$D(1^{1/2}S_1)$	$K$	2	175	0.3
$D_s(1^{3/2}P_2)$	$D(1^{1/2}S_0)$	$K$	2	442	8.9
	$D(1^{1/2}S_1)$	$K$	2	264	1.4
$D_s(1^{1/2}S_0)$	$\eta$	2	248	0.4	
$D_s(1^{1/2}P_1)$	$D(1^{1/2}S_1)$	$K$	0	302	224

states. These rates are shown in Table VIII for the  $D$  and  $D_s$  mesons and in Table IX for the  $B$  and  $B_s$  mesons. For the  $1^{3/2}P_{(1,2)}$   $B_s$  states, our model predicts that they are below threshold for  $K$  transitions to the  $1^{1/2}S_{(0,1)}$   $B$  states. However, this is very sensitive to the details of the model. So, for completeness, we note the partial rates divided by the appropriate phase space factor at  $p_K=0$  in Table X. A list of the allowed transitions for other low-lying excited states is reported in Appendix B.

### C. Comparison to lattice results

As one more consistency check of our model we compare our prediction for the transition<sup>3</sup>  $B \rightarrow B^* + \pi$  with model independent results coming from lattice simulations. We define

$$\mathcal{A}^{BB^*\pi}(r) = \frac{1}{3} \sum_{\mu=1,2,3} \int \langle B^* | A_\mu(\mathbf{r}) | B \rangle d\Omega_r \quad (35)$$

with  $A_\mu(\mathbf{r}) = \bar{q}(t, \mathbf{r}) \gamma_\mu \gamma^5 q(t, \mathbf{r})$  at  $t=0$ . The same matrix element can be expressed in terms of our radial wave function and, in the limit  $p_\pi \rightarrow 0$ ,

$$\mathcal{A}^{BB^*\pi}(r) = -g_A^8 \left[ (f_{1,0,1/2}^0(r))^2 - \frac{1}{3} (f_{1,0,1/2}^1(r))^2 \right] + \mathcal{O}(p_\pi). \quad (36)$$

<sup>3</sup>Even if this transition is kinematically forbidden it is physically relevant to the  $B \rightarrow \pi + l + \bar{\nu}$  exclusive decay under the assumption of vector meson dominance.

TABLE IX. The  $1P$  state hadronic transition rates for  $B$  and  $B_s$  systems.  $H' \rightarrow H + x$ . Decays denoted with an (\*) are allowed only because of the order  $1/m_h$  mixing of states. Values for  $p_x$  and  $\Gamma_x/(g_A^8)^2$  are in MeV.

$H'(n'j'l_J)$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$
$B(1^{1/2}P_0)$	$B(1^{1/2}S_0)$	$\pi$	0	388	186
$B(1^{3/2}P_1)$	$B(1^{1/2}S_1)$	$\pi$	0	338	(*) 0.5
	$B(1^{1/2}S_1)$	$\pi$	2	338	13.1
$B(1^{3/2}P_2)$	$B(1^{1/2}S_0)$	$\pi$	2	396	10.6
	$B(1^{1/2}S_1)$	$\pi$	2	352	9.5
$B(1^{1/2}P_1)$	$B(1^{1/2}S_1)$	$\pi$	0	381	180
$B_s(1^{1/2}P_0)$	$B(1^{1/2}S_0)$	$K$	0	170	159
$B_s(1^{1/2}P_1)$	$B(1^{1/2}S_1)$	$K$	0	153	143

$g_A^8$  is the effective coupling of the transition as defined in Eq. (27).<sup>4</sup> In the chiral quark model  $g_A^8$  is an effective parameter and it has to be given as input. On the other side, in the context of lattice computations, the matrix element in Eq. (35) follows directly from first principles (the QCD Lagrangian) and can be computed explicitly. By fitting the lattice results of Ref. [24] with our prediction for  $\mathcal{A}^{BB^*\pi}(r)$  as function of  $g_A^8$  we are able to determine<sup>5</sup>

$$g_A^8 = Z_A g_A^{8 \text{ (lattice)}} = 0.53 \pm 0.11 \quad (37)$$

where  $g_A^{8 \text{ (lattice)}}$  is the naive lattice result extracted from the fit and  $Z_A = 0.78$  is the lattice matching factor discussed in Ref. [24]. The error includes the statistical error due to the simulation and the fits ( $\approx 10\%$ ), and an estimate of the systematic error in the matching coefficient and in the chiral extrapolation. This is a preliminary result, as those of Ref. [24] are, because of the small lattice size and the poor chiral extrapolation. In any case our result is in agreement with the lattice determination of Ref. [25],  $g_A^8 = 0.61 \pm 0.13$ , and with experimental results from nucleon and hyperon  $\beta$  decays,  $g_A^8 = 0.58 \pm 0.02$  [26]. A more precise lattice determination is possible and may be carried out in the near future.

Apart for the overall normalization given by  $g_A^8$ , the radial dependence of the function  $\mathcal{A}_{B^*B\pi}(r)$  is predicted independently by our model and by the lattice computation. In Fig. 5 we present a comparison between our analytical result, with adjusted normalization, and the lattice data. We believe

<sup>4</sup>Note from Eq. (36) that in the non-relativistic limit the coupling  $g_A^8$  coincides with the coupling constant  $g$  that appears in the heavy meson chiral Lagrangian [23]. The lattice result for this quantity is  $g = 0.42 \pm 0.09$ .

<sup>5</sup>We set to zero terms of the order  $\mathcal{O}(p_\pi)$  for consistency with the lattice computation.

TABLE X. Decay rates for  $1^{3/2}P_{(1,2)} B_s$  mesons with phase space dependence divided out. These states are very near the kinematical threshold in our model. Values for  $p_x$  and  $\Gamma_x/(g_A^8)^2$  are in MeV.

$H'(n'j'l_J)$	$H(n^j l_J)$	$x$	$l_x$	$\Gamma_x/(g_A^8)^2 \times (100/p_x)^{(2l_x+1)}$
$B_s(1^{3/2}P_1)$	$B(1^{1/2}S_1)$	$K$	0	(*) $4.92 \times 10^{-1}$
$B_s(1^{3/2}P_1)$	$B(1^{1/2}S_1)$	$K$	2	$2.38 \times 10^{-2}$
$B_s(1^{3/2}P_2)$	$B(1^{1/2}S_0)$	$K$	2	$9.48 \times 10^{-3}$
$B_s(1^{3/2}P_2)$	$B(1^{1/2}S_1)$	$K$	2	$1.43 \times 10^{-2}$

this comparison provides a satisfactory consistency check of the two methods.<sup>6</sup>

#### D. Comparison to experiment

The total width of the  $D^{*+}$  meson has recently been measured by the CLEO Collaboration [27]. They obtain  $\Gamma = 96 \pm 4_{\text{stat}} \pm 22_{\text{syst}} (\text{keV})$ . Combining this measurement with the well-known branching ratios for the various pionic transitions gives a measurement of chiral coupling constant. We obtain

$$g_A^8 = 0.82 \pm 0.09 \quad (38)$$

within our model.

Within the chiral quark model, the coupling determined from any transition should agree (i.e. the coupling is independent of particular initial or final heavy-light states). Table XI lists the various determinations of  $g_A^8$  from existing data. Within the existing large uncertainties the various determinations are consistent.

#### IV. CONCLUSIONS

We have computed the spectrum and hadronic decay width of the excited  $D$ ,  $D_s$ ,  $B$  and  $B_s$  mesons using a relativistic quark model for the masses and wave function of the heavy-light mesons. This work is based on that of Refs. [8,6,5] but departs from these previous works because we choose a simpler form for the potential and determined its parameters exclusively from fitting the experimental heavy-light spectrum. Moreover, we computed all corrections within the model to order  $1/m_h$ , including mixing between nearby states with the same  $J^P$ .

Our spectrum results agree very well with the existing data in the  $D$  and  $D_s$  systems. The agreement in the  $B$  and  $B_s$  systems is also fairly good but the experimental situation for the  $P$  states is not yet completely resolved.

For example, our model predicts a spin-orbit inversion for the excited  $P$ -wave states in these systems. This agrees with the recent CLEO [14] results for the  $D$  mesons; but is in

<sup>6</sup>It also suggests a possible use of chiral quark model wave functions to isolate excited states contributions in lattice correlation functions.

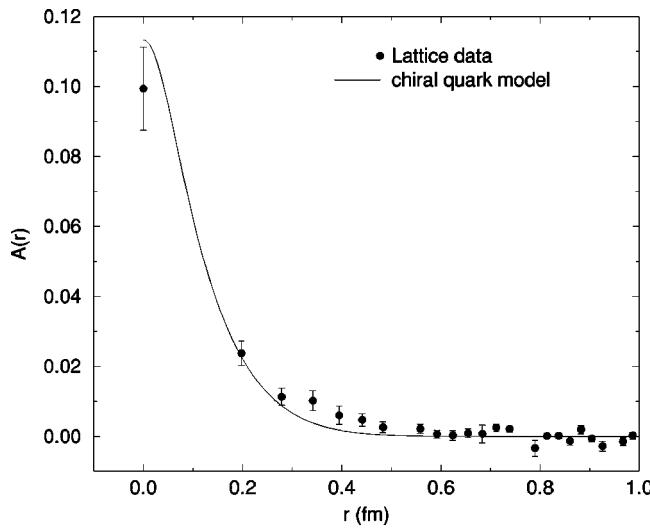


FIG. 5. Comparison between a prediction of our model and the lattice QCD result [24].

disagreement with preliminary *L3* results [12] for the *B* mesons. However, in our model we find that the splitting between the  $j_l=1/2$  *P*-wave states ( $J=0,1$ ) is more than twice as large as the splitting of the  $j_l=3/2$  *P*-wave states ( $J=1,2$ ), which is inconsistent with the assumption of equal splitting used by *L3* [12] in their analysis.

The information on the spectrum and the wave function was used to compute a complete list of allowed hadronic decays for these excited mesons into lower energy states with emission of a light pseudoscalar meson ( $\eta$ ,  $\pi$  or  $K$ ) and their relative branching ratios.

We also compared our model prediction for  $\langle B^* | A_\mu^*(r) | B \rangle$  with lattice results. We find good agreement between the shapes of the decay amplitudes. We used this comparison to extract a preliminary lattice determination of

$g_A^8$ , the effective coupling of the chiral quark model. We find the value  $g_A^8 = 0.53 \pm 0.11$ . Using this value for  $g_A^8$  to translate the total hadronic decay widths in physical units, we would conclude that the widths of the *D* meson *P* waves are consistently below experimental findings. Of course, the lattice results have not yet been extrapolated to the continuum.

Comparison of our results with a recent experiment measurement of the  $D^{*+}$  width [27] yields  $g_A^8 = 0.82 \pm 0.09$ . Using this value, we find much better overall agreement with experimental findings.

Finally, there is also a whole set of hadronic decays with emission of light vector mesons that we did not include in our study. For the higher excited states these transitions give an important contribution to the total physical widths. We intend to study these transitions in a future paper.

## ACKNOWLEDGMENTS

We wish to acknowledge the authors of Ref. [24] from whose work we extracted the lattice data of Fig. 5. One of us, D.P., thanks G. Chiodini for stimulating discussions on experimental issues. This work was performed at Fermilab, a U.S. Department of Energy Laboratory (operated by the University Research Association, Inc.) under contract DE-AC02-76CHO3000.

## APPENDIX A

In order to be able to factorize the expression for the transition amplitude, Eq. (30), into radial and angular parts, it is convenient to adopt the following basis for the  $\Gamma$  matrices:

$$\Gamma_\mu^{00} = \begin{pmatrix} \sigma_\mu & 0 \\ 0 & 0 \end{pmatrix}, \quad \Gamma_\mu^{01} = \begin{pmatrix} 0 & \sigma_\mu \\ 0 & 0 \end{pmatrix}, \quad (\text{A1})$$

TABLE XI. The  $1S$  and  $1P$  heavy-light hadronic transition rates compared to experiment. All widths are in MeV unless otherwise indicated. Experimental values are from the Particle Data Group (PDG) [11] unless otherwise indicated.

State	Width (expt.)	[Ref.]	$\Gamma/(g_A^8)^2$ (model)	$g_A^8$
$D^+(1^{1/2}S_1)$	$96 \pm 4 \pm 22$ (keV)	[27]	$143$ (keV) <sup>a</sup>	$0.82 \pm 0.09$
$D^0(1^{3/2}P_1)$	$18.9_{-3.5}^{+4.6}$		$16$	$1.09_{-0.11}^{+0.12}$
$D^+(1^{3/2}P_1)$	$28 \pm 8$		$16$	$1.32_{-0.27}^{+0.18}$
$D^+(1^{1/2}P_1)$	$290_{-79}^{+101} \pm 26 \pm 36$	[14]	$181$	$1.27 \pm 0.22$
$D^0(1^{3/2}P_2)$	$23 \pm 5$		$38$	$0.77 \pm 0.08$
$D^+(1^{3/2}P_2)$	$25 \pm 8$		$38$	$0.81_{-0.14}^{+0.12}$
$D_s(1^{3/2}P_1)$	$\leq 2.3$		$2.0$	$\leq 1.07$
$D_s(1^{3/2}P_2)$	$15 \pm 5$		$10.9$	$1.17_{-0.11}^{+0.18}$
$B(1^{1/2}P_1)$	$73 \pm 44$	[12] <sup>b</sup>	$180$	$0.64_{-0.21}^{+0.17}$
$B(1^{3/2}P_1)$	$18_{-13}^{+15} \pm 29$	[13] <sup>b</sup>	$14.0$	$1.13_{-1.13}^{+0.77}$
$B(1^{3/2}P_2)$	$41 \pm 43$	[12] <sup>b</sup>	$20.0$	$1.43_{-1.43}^{+0.61}$

<sup>a</sup>Theoretical value corrected for phase space observed mass (see Table VII) and the 1.6% branching ratio to  $D^+ + \gamma$  [11].

<sup>b</sup>Experimental results depend strongly on model dependent assumptions.

$$\Gamma_\mu^{10} = \begin{pmatrix} 0 & 0 \\ \sigma_\mu & 0 \end{pmatrix}, \quad \Gamma_\mu^{11} = \begin{pmatrix} 0 & 0 \\ 0 & \sigma_\mu \end{pmatrix} \quad (\text{A2})$$

where  $\sigma_0$  is the  $2 \times 2$  identity and  $\sigma_i$  are the usual Pauli matrices. On this basis we obtain simple expressions for the  $c_{l_x}^{ab,k}(X)$  coefficients<sup>7</sup>

$$c_{l_x}^{00,k}(X) = c_0 \text{tr}(\gamma^0 X \Gamma_\mu^{00}) \langle j', m', l' | \sigma^\mu Y_{k,m_k} | j, m, l \rangle \quad (\text{A4})$$

$$c_{l_x}^{01,k}(X) = c_0 \text{tr}(\gamma^0 X \Gamma_\mu^{01}) \langle j', m', l' | \sigma^\mu Y_{k,m_k} | j, m, 2j-l \rangle \quad (\text{A5})$$

$$c_{l_x}^{10,k}(X) = c_0 \text{tr}(\gamma^0 X \Gamma_\mu^{10}) \langle j', m', 2j-l' | \sigma^\mu Y_{k,m_k} | j, m, l \rangle \quad (\text{A6})$$

$$c_{l_x}^{11,k}(X) = c_0 \text{tr}(\gamma^0 X \Gamma_\mu^{11}) \langle j', m', 2j-l' | \sigma^\mu Y_{k,m_k} | j, m, 2j-l \rangle \quad (\text{A7})$$

where

$$c_0 = 4\pi(-i)^{l_x} \frac{\sum_s C_{j', M' - s; 1/2, s}^{J', M'} C_{j, M - s; 1/2, s}^{J, M}}{C_{l_x, m_x; j, M}^{J', M'}} \frac{1}{2}. \quad (\text{A8})$$

For the particular case  $X = \not{p} \gamma^5$  the angular dependence from the  $\mathbf{p}$  vector disappears and we obtain the following explicit expression:

$$c_{l_x}^{00,k}(\not{p} \gamma^5) = +c_2 |p| \langle j', l' | |T_{l_x}^{(k)}| | j, l \rangle \quad (\text{A9})$$

$$c_{l_x}^{01,k}(\not{p} \gamma^5) = +c_1 p_0 \langle j', l' | |Y_k| | j, 2j-l \rangle \quad (\text{A10})$$

$$c_{l_x}^{10,k}(\not{p} \gamma^5) = -c_1 p_0 \langle j', 2j-l' | |Y_k| | j, l \rangle \quad (\text{A11})$$

with

<sup>7</sup>In this appendix we follow the notation of Elbaz [28] where

$$[a_1 \dots a_n] = \sqrt{(2a_1+1) \dots (2a_n+1)}. \quad (\text{A3})$$

$$c_1 = 4\pi i (-i)^{l_x} (-)^{j'-(1/2)+J'+l'-l+l_x} [J] \times \left\{ \begin{array}{ccc} l_x & j & j' \\ \frac{1}{2} & J' & J \end{array} \right\} \delta_{k,l_x} \quad (\text{A13})$$

$$c_2 = 4\pi (-i)^k (-)^{j'-(1/2)+J'-k} [Jk] \times \left\{ \begin{array}{ccc} l_x & j & j' \\ \frac{1}{2} & J' & J \end{array} \right\} \left( \begin{array}{ccc} 1 & k & l_\pi \\ 0 & 0 & 0 \end{array} \right). \quad (\text{A14})$$

Explicit expressions for the Wigner-Eckart reduced tensors are

$$\langle l' j' \| Y_k \| l j \rangle = \frac{[jj' ll' k]}{\sqrt{4\pi}} (-)^{j+l'-l+1/2} \left\{ \begin{array}{ccc} l' & k & l \\ 0 & 0 & 0 \end{array} \right\} \times \left\{ \begin{array}{ccc} k & l' & l \\ \frac{1}{2} & j & j' \end{array} \right\} \quad (\text{A15})$$

and

$$\langle l' j' \| T_{l_x}^{(k)} \| l j \rangle = \sqrt{\frac{3}{2\pi}} [jj' ll' k l_x] (-)^{l'} \left\{ \begin{array}{ccc} l' & k & l \\ 0 & 0 & 0 \end{array} \right\} \times \left\{ \begin{array}{ccc} \frac{1}{2} & l' & j' \\ \frac{1}{2} & l & j \\ 1 & k & l_x \end{array} \right\}. \quad (\text{A16})$$

Our expression for the spin structure of the decay amplitudes, reported in this appendix, disagree with Ref. [5] in the overall phase factor. This factor does not affect their results but is relevant in case of mixing.

## APPENDIX B

In this appendix we list the hadronic transitions  $H' \rightarrow H + x$  for the low-lying excited states not discussed in Sec. III B. Table XII lists the transitions in descending order according to (i) the flavor of the decaying heavy meson  $H'$ ; (ii) its mass; (iii) the decay channel.

Transitions with a branching ratio less than 1% are not reported. Some decays are marked with an asterisk. These are decays that apparently do not conserve the heavy quark spin ( $|j' - j| \leq l_x \leq |j' + j|$ ) but, we remind the reader, our initial and final states are physical states and therefore such decays become allowed because of mixing effects.

TABLE XII. Hadronic transitions  $H' \rightarrow H + x$  for the low-lying states not discussed in Sec. III B.

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$															
$H' = D(2^{1/2}S_0) \quad m = 2.589 \text{ GeV}$																								
$D(1^{1/2}S_1)$	$\pi$	1	504	14.5	$D(1^{1/2}S_1)$	$\eta$	1	528	4.4															
$D(1^{1/2}P_0)$	$\pi$	0	154	7.0	$D(1^{1/2}S_1)$	$\pi$	1	697	22.9															
$H(n^j l_J)$																								
$H' = D(2^{1/2}S_1) \quad m = 2.692 \text{ GeV}$																								
$D(1^{1/2}S_0)$	$\eta$	1	518	1.4	$D(1^{3/2}P_1)$	$\pi$	2	363	3.6															
$D(1^{1/2}S_0)$	$\pi$	1	688	39.9	$D(1^{3/2}P_2)$	$\pi$	0	323	87.0															
$D(1^{1/2}S_1)$	$\pi$	1	587	30.4	$D(1^{3/2}P_2)$	$\pi$	2	323	3.2															
$D(1^{3/2}P_1)$	$\pi$	2	225	1.9	$D_s(1^{1/2}S_1)$	$K$	1	456	13.0															
$D(1^{1/2}P_1)$	$\pi$	0	141	6.2	$H(n^j l_J)$																			
$D_s(1^{1/2}S_0)$	$K$	1	460	5.6	$H' = D(2^{1/2}P_0) \quad m = 2.949 \text{ GeV}$																			
$H(n^j l_J)$																								
$H' = D(1^{5/2}D_2) \quad m = 2.775 \text{ GeV}$																								
$D(1^{1/2}S_1)$	$\pi$	3	652	20.1	$D(1^{1/2}S_0)$	$\eta$	0	756	11.7															
$D(1^{1/2}P_0)$	$\pi$	2	347	7.3	$D(1^{1/2}S_0)$	$\pi$	0	875	88.0															
$D(1^{3/2}P_1)$	$\pi$	2	308	4.3	$D(1^{3/2}P_1)$	$\pi$	1	467	15.2															
$D(1^{3/2}P_2)$	$\pi$	2	266	1.4	$D(1^{1/2}P_1)$	$\pi$	1	403	60.6															
$D(1^{1/2}P_1)$	$\pi$	2	236	1.3	$D(2^{1/2}S_0)$	$\pi$	0	311	51.9															
$D_s(1^{1/2}S_1)$	$K$	3	387	0.5	$D_s(1^{1/2}S_0)$	$K$	0	706	66.6															
$H(n^j l_J)$																								
$H' = D(1^{3/2}D_1) \quad m = 2.795 \text{ GeV}$																								
$D(1^{1/2}S_0)$	$\eta$	1	620	4.0	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$															
$D(1^{1/2}S_0)$	$\pi$	1	764	18.6	$H' = D(2^{3/2}P_1) \quad m = 2.995 \text{ GeV}$																			
$D(1^{1/2}S_1)$	$\pi$	1	668	6.8	$D(1^{1/2}S_1)$	$\eta$	2	685	2.7															
$D(1^{3/2}P_1)$	$\pi$	0	328	87.2	$D(1^{1/2}S_1)$	$\pi$	2	818	62.5															
$D(1^{3/2}P_1)$	$\pi$	2	328	2.4	$D(1^{1/2}P_0)$	$\pi$	1	540	1.5															
$D(1^{3/2}P_2)$	$\pi$	2	286	1.4	$D(1^{3/2}P_1)$	$\pi$	1	506	5.9															
$D_s(1^{1/2}S_0)$	$K$	1	566	15.0	$D(1^{3/2}P_2)$	$\pi$	3	470	4.0															
$D_s(1^{1/2}S_1)$	$K$	1	412	3.3	$D(1^{1/2}P_1)$	$\pi$	1	444	3.9															
$H(n^j l_J)$																								
$H' = D(1^{5/2}D_3) \quad m = 2.799 \text{ GeV}$																								
$D(1^{1/2}S_0)$	$\eta$	3	624	0.7	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$															
$D(1^{1/2}S_0)$	$\pi$	3	767	18.0	$H' = D(2^{3/2}P_2) \quad m = 3.035 \text{ GeV}$																			
$D(1^{1/2}S_1)$	$\pi$	3	671	13.2	$D(1^{1/2}S_0)$	$\eta$	2	828	3.3															
$D(1^{3/2}P_1)$	$\pi$	2	331	2.5	$D(1^{1/2}S_0)$	$\pi$	2	936	53.4															
$D(1^{3/2}P_2)$	$\pi$	2	290	5.2	$D(1^{1/2}S_1)$	$\eta$	2	721	2.3															
$D(1^{1/2}P_1)$	$\pi$	2	261	3.5	$D(1^{1/2}S_1)$	$\pi$	2	848	47.1															
$D_s(1^{1/2}S_0)$	$K$	3	570	2.1	$D(1^{3/2}P_1)$	$\pi$	3	541	4.3															

TABLE XII. (*Continued*).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = D(2^{1/2}P_1) \quad m = 3.045 \text{ GeV}$														
$D(1^{1/2}S_1)$	$\eta$	0	730	10.9	$D(1^{1/2}S_1)$	$\pi$	2	910	4.9					
$D(1^{1/2}S_1)$	$\pi$	0	855	85.4	$D(1^{1/2}P_0)$	$\pi$	3	645	4.3					
$D(1^{1/2}P_0)$	$\pi$	1	582	61.0	$D(1^{3/2}P_1)$	$\pi$	3	613	3.2					
$D(1^{3/2}P_1)$	$\pi$	1	549	6.6	$D(1^{3/2}P_2)$	$\eta$	1	332	3.4					
$D(1^{3/2}P_2)$	$\pi$	1	513	19.9	$D(1^{3/2}P_2)$	$\pi$	1	579	23.7					
$D(1^{1/2}P_1)$	$\pi$	1	488	61.5	$D(1^{3/2}P_2)$	$\pi$	3	579	2.6					
$D(2^{1/2}S_1)$	$\pi$	0	306	50.6	$D(1^{1/2}P_1)$	$\pi$	3	554	1.2					
$D_s(1^{1/2}S_1)$	$K$	0	667	59.3	$D(1^{5/2}D_2)$	$\pi$	2	300	1.2					
$H' = D(1^{7/2}F_3) \quad m = 3.074 \text{ GeV}$														
$D(1^{1/2}S_1)$	$\eta$	4	754	0.5	$D(1^{5/2}D_3)$	$\pi$	2	277	3.6					
$D(1^{1/2}S_1)$	$\pi$	4	875	13.1	$D_s(1^{1/2}S_1)$	$K$	2	735	3.3					
$D(1^{1/2}P_0)$	$\pi$	3	606	11.4	$D_s(1^{3/2}P_2)$	$K$	1	203	3.7					
$D(1^{3/2}P_1)$	$\pi$	3	573	7.3	$H(n^j l_J)$									
$D(1^{3/2}P_2)$	$\pi$	3	538	3.8	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$D(1^{1/2}P_1)$	$\pi$	3	513	5.0	$H' = D(3^{1/2}S_0) \quad m = 3.141 \text{ GeV}$									
$D(1^{5/2}D_2)$	$\pi$	2	251	2.9	$D(1^{1/2}S_1)$	$\eta$	1	811	1.7					
$D(1^{3/2}D_1)$	$\pi$	2	230	3.3	$D(1^{1/2}S_1)$	$\pi$	1	923	44.5					
$D_s(1^{1/2}S_1)$	$K$	4	693	1.5	$D(1^{1/2}P_0)$	$\pi$	0	660	2.9					
$H' = D(1^{7/2}F_4) \quad m = 3.091 \text{ GeV}$														
$D(1^{1/2}S_0)$	$\eta$	4	872	0.6	$D(1^{3/2}P_2)$	$\pi$	2	594	15.1					
$D(1^{1/2}S_0)$	$\pi$	4	974	9.8	$D(2^{1/2}S_1)$	$\pi$	1	396	16.3					
$D(1^{1/2}S_1)$	$\pi$	4	887	7.8	$D(2^{1/2}P_0)$	$\pi$	0	128	6.5					
$D(1^{3/2}P_1)$	$\pi$	3	587	4.4	$D_s(1^{1/2}S_1)$	$K$	1	751	8.7					
$D(1^{3/2}P_2)$	$\pi$	3	552	9.9	$D_s(1^{1/2}P_0)$	$K$	0	384	1.4					
$D(1^{1/2}P_1)$	$\pi$	3	527	9.6	$H(n^j l_J)$									
$D(1^{5/2}D_2)$	$\pi$	2	268	0.6	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$D(1^{5/2}D_3)$	$\pi$	2	244	2.9	$H' = D(3^{1/2}S_1) \quad m = 3.226 \text{ GeV}$									
$D(1^{3/2}D_2)$	$\pi$	2	208	2.4	$D(1^{1/2}S_0)$	$\eta$	1	976	3.2					
$D_s(1^{1/2}S_0)$	$K$	4	824	2.0	$D(1^{1/2}S_0)$	$\pi$	1	1066	50.2					
$D_s(1^{1/2}S_1)$	$K$	4	707	0.9	$D(1^{1/2}S_1)$	$\pi$	1	984	53.5					
$H(n^j l_J)$														
$H' = D(1^{5/2}F_2) \quad m = 3.101 \text{ GeV}$														
$D(1^{1/2}S_0)$	$\pi$	2	981	3.7	$D(1^{3/2}P_1)$	$\pi$	2	697	23.1					
$D(1^{1/2}S_1)$	$\pi$	2	895	1.8	$D(1^{3/2}P_2)$	$\pi$	2	663	17.0					
$D(1^{3/2}P_1)$	$\eta$	1	362	3.7	$D(2^{1/2}S_0)$	$\pi$	1	560	41.3					
$D(1^{3/2}P_1)$	$\pi$	1	595	22.2	$D(2^{1/2}S_1)$	$\pi$	1	473	33.5					
$D(1^{3/2}P_1)$	$\pi$	3	595	1.4	$D(2^{1/2}P_1)$	$\pi$	0	113	5.6					
$D(1^{3/2}P_2)$	$\pi$	1	560	2.2	$D_s(1^{1/2}S_0)$	$K$	1	930	17.6					
$D(1^{3/2}P_2)$	$\pi$	3	560	2.1	$D_s(1^{1/2}S_1)$	$K$	1	822	13.4					
$D(1^{1/2}P_1)$	$\pi$	3	536	2.6	$H(n^j l_J)$									
$D(1^{5/2}D_2)$	$\pi$	0	279	56.8	$H' = D_s(2^{1/2}S_0) \quad m = 2.700 \text{ GeV}$									
$D(1^{5/2}D_2)$	$\pi$	2	279	3.4	$D(1^{1/2}S_1)$	$K$	1	424	3.12					
$D_s(1^{1/2}S_0)$	$K$	2	832	3.1	$D_s(1^{1/2}S_1)$	$\eta$	1	187	0.04					
$D_s(1^{1/2}S_1)$	$K$	2	716	1.2										
$D_s(1^{3/2}P_1)$	$K$	1	250	6.0										

TABLE XII. (Continued).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = D_s(2^{1/2}S_1) \quad m = 2.806 \text{ GeV}$														
$D(1^{1/2}S_0)$	$K$	1	661	21.1	$D(1^{1/2}S_0)$	$K$	2	943	34.9					
$D(1^{1/2}S_1)$	$K$	1	539	12.2	$D(1^{1/2}S_1)$	$K$	2	848	28.9					
$D_s(1^{1/2}S_0)$	$\eta$	1	540	6.2	$D(1^{3/2}P_1)$	$K$	3	485	1.8					
$D_s(1^{1/2}S_1)$	$\eta$	1	371	1.5	$D(1^{1/2}P_1)$	$K$	1	400	11.0					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$D(2^{1/2}S_0)$	$K$	2	255	1.3					
$H' = D_s(1^{5/2}D_2) \quad m = 2.900 \text{ GeV}$														
$D(1^{1/2}S_1)$	$K$	3	629	10.6	$D_s(1^{1/2}S_0)$	$\eta$	2	854	14.8					
$D_s(1^{1/2}S_1)$	$\eta$	3	486	1.4	$D_s(1^{1/2}S_1)$	$\eta$	2	738	9.5					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = D_s(1^{3/2}D_1) \quad m = 2.913 \text{ GeV}$														
$D(1^{1/2}S_0)$	$K$	1	752	26.1	$H' = D_s(2^{1/2}P_1) \quad m = 3.165 \text{ GeV}$	$D(1^{1/2}S_1)$	$K$	0	854					
$D(1^{1/2}S_1)$	$K$	1	641	10.7		$D(1^{1/2}P_0)$	$K$	1	537					
$D(1^{3/2}P_1)$	$K$	0	47	12.6		$D(1^{3/2}P_1)$	$K$	1	494					
$D_s(1^{1/2}S_0)$	$\eta$	1	644	39.7		$D(1^{3/2}P_2)$	$K$	1	446					
$D_s(1^{1/2}S_1)$	$\eta$	1	501	15.2		$D(1^{1/2}P_1)$	$K$	1	410					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$		$D_s(1^{1/2}S_1)$	$\eta$	0	745					
$H' = D_s(1^{5/2}D_3) \quad m = 2.925 \text{ GeV}$														
$D(1^{1/2}S_0)$	$K$	3	762	11.4	$D_s(1^{1/2}P_0)$	$\eta$	1	357	8.1					
$D(1^{1/2}S_1)$	$K$	3	652	7.3		$D_s(1^{3/2}P_1)$	$\eta$	1	280					
$D_s(1^{1/2}S_0)$	$\eta$	3	656	3.1		$D_s(1^{3/2}P_2)$	$\eta$	1	184					
$D_s(1^{1/2}S_1)$	$\eta$	3	514	1.1	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H' = D_s(1^{7/2}F_3) \quad m = 3.203 \text{ GeV}$									
$H' = D_s(1^{3/2}D_2) \quad m = 2.953 \text{ GeV}$										$D(1^{1/2}S_1)$	$K$	4	884	9.2
$D(1^{1/2}S_1)$	$K$	1	677	35.0		$D(1^{1/2}P_0)$	$K$	3	575	5.5				
$D_s(1^{1/2}S_1)$	$\eta$	1	543	16.4		$D(1^{3/2}P_1)$	$K$	3	534	3.2				
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$		$D(1^{3/2}P_2)$	$K$	3	489	1.5				
$H' = D_s(2^{1/2}P_0) \quad m = 3.067 \text{ GeV}$										$D(1^{1/2}P_1)$	$K$	3	456	1.6
$D(1^{1/2}S_0)$	$K$	0	875	74.1		$D_s(1^{1/2}S_1)$	$\eta$	4	778	2.2				
$D(1^{3/2}P_1)$	$K$	1	377	32.3		$D_s(1^{1/2}P_0)$	$\eta$	3	409	0.5				
$D(1^{1/2}P_1)$	$K$	1	270	17.8	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$D_s(1^{1/2}S_0)$	$\eta$	0	780	49.1	$H' = D_s(1^{7/2}F_4) \quad m = 3.220 \text{ GeV}$									
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$		$D(1^{1/2}S_0)$	$K$	4	990	7.2				
$H' = D_s(2^{3/2}P_1) \quad m = 3.114 \text{ GeV}$										$D(1^{1/2}S_1)$	$K$	4	897	5.6
$D(1^{1/2}S_1)$	$K$	0	813	(*) 0.7		$D(1^{3/2}P_1)$	$K$	3	552	2.0				
$D(1^{1/2}S_1)$	$K$	2	813	36.6		$D(1^{3/2}P_2)$	$K$	3	508	4.0				
$D(1^{1/2}P_0)$	$K$	1	482	5.5		$D(1^{1/2}P_1)$	$K$	3	475	3.3				
$D(1^{3/2}P_1)$	$K$	1	436	1.6		$D_s(1^{1/2}S_0)$	$\eta$	4	904	2.4				
$D(1^{3/2}P_2)$	$K$	3	383	0.9		$D_s(1^{1/2}S_1)$	$\eta$	4	793	1.4				
$D(1^{1/2}P_1)$	$K$	1	343	4.5										
$D_s(1^{1/2}S_1)$	$\eta$	2	700	11.1										
$D_s(1^{1/2}P_0)$	$\eta$	1	275	1.6										

TABLE XII. (*Continued*).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = D_s(1^{5/2}F_2) \quad m = 3.224 \text{ GeV}$														
$D(1^{1/2}S_0)$	$K$	2	993	5.4	$B(1^{1/2}S_1)$	$\pi$	1	519	21.9					
$D(1^{1/2}S_1)$	$K$	2	900	3.1	$B(1^{1/2}P_0)$	$\pi$	0	114	4.9					
$D(1^{3/2}P_1)$	$K$	1	556	39.1	$H' = B(2^{1/2}S_0) \quad m = 5.886 \text{ GeV}$									
$D(1^{3/2}P_1)$	$K$	3	556	0.9	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$D(1^{3/2}P_2)$	$K$	1	512	4.0	$H' = B(2^{1/2}S_1) \quad m = 5.920 \text{ GeV}$									
$D(1^{3/2}P_2)$	$K$	3	512	1.1	$B(1^{1/2}S_0)$	$\pi$	1	591	19.2					
$D(1^{1/2}P_1)$	$K$	3	480	0.9	$B(1^{1/2}S_1)$	$\pi$	1	550	22.9					
$D_s(1^{1/2}S_0)$	$\eta$	2	907	3.1	$B(1^{3/2}P_1)$	$\pi$	2	167	0.5					
$D_s(1^{1/2}S_1)$	$\eta$	2	796	1.5	$B(1^{1/2}P_1)$	$\pi$	0	109	4.6					
$D_s(1^{3/2}P_1)$	$\eta$	1	372	14.0	$H' = B(1^{5/2}D_2) \quad m = 5.985 \text{ GeV}$									
$D_s(1^{3/2}P_2)$	$\eta$	1	302	0.9	$B(1^{1/2}S_1)$	$\pi$	3	611	17.4					
$H' = D_s(1^{5/2}F_3) \quad m = 3.247 \text{ GeV}$														
$D(1^{1/2}S_1)$	$K$	2	918	8.2	$B(1^{3/2}P_1)$	$\pi$	2	244	1.8					
$D(1^{1/2}P_0)$	$K$	3	619	2.2	$B(1^{1/2}P_0)$	$\pi$	2	237	1.6					
$D(1^{3/2}P_1)$	$K$	3	580	2.0	$B(1^{3/2}P_2)$	$\pi$	2	228	0.7					
$D(1^{3/2}P_2)$	$K$	1	536	42.2	$B(1^{1/2}P_1)$	$\pi$	2	195	0.5					
$D(1^{3/2}P_2)$	$K$	3	536	1.5	$H' = B(1^{5/2}D_3) \quad m = 5.993 \text{ GeV}$									
$D_s(1^{1/2}S_1)$	$\eta$	2	815	3.9	$B(1^{1/2}S_0)$	$\pi$	3	658	11.1					
$D_s(1^{3/2}P_2)$	$\eta$	1	339	12.6	$B(1^{1/2}S_1)$	$\pi$	3	618	10.6					
$H' = D_s(3^{1/2}S_0) \quad m = 3.259 \text{ GeV}$														
$D(1^{1/2}S_1)$	$K$	1	927	19.0	$B(1^{3/2}P_1)$	$\pi$	2	252	0.6					
$D(1^{1/2}P_0)$	$K$	0	630	2.1	$B(1^{3/2}P_2)$	$\pi$	2	237	2.2					
$D(1^{3/2}P_2)$	$K$	2	549	5.1	$B(1^{1/2}P_1)$	$\pi$	2	204	1.3					
$D(2^{1/2}S_1)$	$K$	1	254	1.0	$H' = B(1^{3/2}D_1) \quad m = 6.025 \text{ GeV}$									
$D_s(1^{1/2}S_1)$	$\eta$	1	825	6.8	$B(1^{1/2}S_0)$	$\eta$	1	475	2.8					
$D_s(1^{1/2}P_0)$	$\eta$	0	478	0.9	$B(1^{1/2}S_0)$	$\pi$	1	687	18.2					
$H' = D_s(3^{1/2}S_1) \quad m = 3.345 \text{ GeV}$														
$D(1^{1/2}S_0)$	$K$	1	1080	28.0	$B(1^{1/2}S_1)$	$\pi$	1	647	7.5					
$D(1^{1/2}S_1)$	$K$	1	993	26.6	$B(1^{3/2}P_1)$	$\pi$	0	286	81.4					
$D(1^{3/2}P_1)$	$K$	2	675	12.2	$B(1^{3/2}P_1)$	$\pi$	2	286	1.4					
$D(1^{3/2}P_2)$	$K$	2	635	8.1	$B(1^{1/2}S_0)$	$K$	1	403	7.2					
$D(1^{1/2}P_1)$	$K$	0	607	1.8	$B_s(1^{1/2}S_0)$	$K$	1	330	2.0					
$D(2^{1/2}S_0)$	$K$	1	507	18.8	$H' = B(1^{3/2}D_2) \quad m = 6.037 \text{ GeV}$									
$D(2^{1/2}S_1)$	$K$	1	385	8.4	$B(1^{1/2}S_1)$	$\eta$	1	431	3.3					
$D_s(1^{1/2}S_0)$	$\eta$	1	1001	13.6	$B(1^{1/2}S_1)$	$\pi$	1	658	23.9					
$D_s(1^{1/2}S_1)$	$\eta$	1	896	10.6	$B(1^{3/2}P_1)$	$\pi$	2	299	1.3					
$D_s(1^{3/2}P_1)$	$\eta$	2	523	1.7	$B(1^{3/2}P_2)$	$\pi$	0	284	81.1					
$D_s(2^{1/2}S_0)$	$\eta$	1	309	1.7	$B(1^{3/2}P_2)$	$\pi$	2	284	2.0					
					$B_s(1^{1/2}S_1)$	$K$	1	350	7.1					

TABLE XII. (Continued).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = B(2^{1/2}P_0) \quad m = 6.163 \text{ GeV}$														
$B(1^{1/2}S_0)$	$\eta$	0	643	9.9	$B(1^{1/2}S_1)$	$\eta$	4	659	0.3					
$B(1^{1/2}S_0)$	$\pi$	0	810	97.4	$B(1^{1/2}S_1)$	$\pi$	4	822	11.7					
$B(1^{3/2}P_1)$	$\pi$	1	425	21.1	$B(1^{3/2}P_1)$	$\pi$	3	481	3.5					
$B(1^{1/2}P_1)$	$\pi$	1	383	52.9	$B(1^{1/2}P_0)$	$\pi$	3	475	3.8					
$B(2^{1/2}S_0)$	$\pi$	0	233	39.6	$B(1^{3/2}P_2)$	$\pi$	3	468	1.9					
$B_s(1^{1/2}S_0)$	$K$	0	576	51.0	$B(1^{1/2}P_1)$	$\pi$	3	440	2.1					
$H' = B(2^{3/2}P_1) \quad m = 6.175 \text{ GeV}$														
$B(1^{1/2}S_1)$	$\eta$	2	606	1.5	$B(1^{1/2}S_1)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$B(1^{1/2}S_1)$	$\pi$	2	782	59.6	$H' = B(1^{7/2}F_3) \quad m = 6.220 \text{ GeV}$									
$B(1^{3/2}P_1)$	$\pi$	1	437	1.6	$B(1^{1/2}S_0)$	$\pi$	4	865	7.0					
$B(1^{1/2}P_0)$	$\pi$	1	431	2.4	$B(1^{1/2}S_1)$	$\pi$	4	827	6.7					
$B(1^{3/2}P_2)$	$\pi$	3	423	2.6	$B(1^{3/2}P_1)$	$\pi$	3	486	1.5					
$B(1^{1/2}P_1)$	$\pi$	1	395	3.4	$B(1^{3/2}P_2)$	$\pi$	3	473	4.7					
$B(2^{1/2}S_1)$	$\pi$	2	209	2.3	$B(1^{1/2}P_1)$	$\pi$	3	445	4.4					
$B(1^{3/2}D_1)$	$\pi$	0	55	1.4	$B(1^{5/2}D_3)$	$\pi$	2	183	0.8					
$B_s(1^{1/2}S_1)$	$K$	2	534	4.6	$B_s(1^{1/2}S_0)$	$K$	4	647	0.5					
$H' = B(2^{3/2}P_2) \quad m = 6.188 \text{ GeV}$														
$B(1^{1/2}S_0)$	$\eta$	2	671	1.2	$B_s(1^{1/2}S_1)$	$K$	4	594	0.3					
$B(1^{1/2}S_0)$	$\pi$	2	832	36.0	$H' = B(1^{7/2}F_4) \quad m = 6.226 \text{ GeV}$									
$B(1^{1/2}S_1)$	$\eta$	2	621	1.1	$B(1^{1/2}S_0)$	$\pi$	4	865	7.0					
$B(1^{1/2}S_1)$	$\pi$	2	793	39.7	$B(1^{1/2}S_1)$	$\pi$	4	827	6.7					
$B(1^{3/2}P_1)$	$\pi$	3	449	2.2	$B(1^{3/2}P_1)$	$\pi$	3	486	1.5					
$B(1^{3/2}P_2)$	$\pi$	3	436	1.2	$B(1^{3/2}P_2)$	$\pi$	3	473	4.7					
$B(1^{1/2}P_1)$	$\pi$	1	408	5.7	$B(1^{1/2}P_1)$	$\pi$	3	445	4.4					
$B(2^{1/2}S_0)$	$\pi$	2	261	2.4	$B(1^{5/2}D_3)$	$\pi$	2	183	0.8					
$B(2^{1/2}S_1)$	$\pi$	2	224	1.9	$B_s(1^{1/2}S_0)$	$K$	4	647	0.5					
$B(1^{3/2}D_2)$	$\pi$	0	57	1.5	$B_s(1^{1/2}S_1)$	$K$	4	594	0.3					
$B_s(1^{1/2}S_0)$	$K$	2	605	4.2	$H' = B(1^{5/2}F_2) \quad m = 6.264 \text{ GeV}$									
$B_s(1^{1/2}S_1)$	$K$	2	550	3.4	$B(1^{1/2}S_0)$	$\pi$	2	898	3.7					
$H' = B(2^{3/2}P_1) \quad m = 6.194 \text{ GeV}$														
$B(1^{1/2}S_1)$	$\eta$	0	629	9.2	$B(1^{1/2}S_1)$	$\pi$	2	860	2.0					
$B(1^{1/2}S_1)$	$\pi$	0	799	93.2	$B(1^{3/2}P_1)$	$\pi$	1	522	18.9					
$B(1^{3/2}P_1)$	$\pi$	1	455	6.1	$B(1^{3/2}P_2)$	$\pi$	1	509	2.0					
$B(1^{1/2}P_0)$	$\pi$	1	449	30.6	$B(1^{1/2}P_2)$	$\pi$	3	509	1.3					
$B(1^{3/2}P_2)$	$\pi$	1	442	22.6	$B(1^{1/2}P_1)$	$\pi$	3	482	1.3					
$B(1^{1/2}P_1)$	$\pi$	1	414	39.5	$B(1^{5/2}D_2)$	$\pi$	0	236	51.2					
$B(2^{1/2}S_1)$	$\pi$	0	231	38.7	$B(1^{5/2}D_2)$	$\pi$	2	236	1.6					
$B_s(1^{1/2}S_1)$	$K$	0	557	46.3	$B_s(1^{1/2}S_0)$	$K$	2	688	1.9					
$H' = B(1^{5/2}F_3) \quad m = 6.271 \text{ GeV}$														
$B(1^{1/2}S_1)$	$\eta$	2	714	1.0	$B_s(1^{1/2}S_1)$	$K$	2	637	0.9					
$B(1^{1/2}S_1)$	$\pi$	2	866	5.2	$H' = B(1^{5/2}F_4) \quad m = 6.271 \text{ GeV}$									
$B(1^{3/2}P_1)$	$\pi$	3	529	1.2	$B(1^{1/2}S_0)$	$\pi$	2	898	3.7					
$B(1^{1/2}P_0)$	$\pi$	3	523	1.0	$B(1^{1/2}S_1)$	$\pi$	2	860	2.0					
$B(1^{3/2}P_2)$	$\pi$	1	516	20.4	$B(1^{3/2}P_1)$	$\pi$	1	522	18.9					
$B(1^{3/2}P_2)$	$\pi$	3	516	1.5	$B(1^{3/2}P_2)$	$\pi$	1	509	2.0					
$B(1^{5/2}D_3)$	$\pi$	0	235	51.0	$B(1^{1/2}P_2)$	$\pi$	3	509	1.3					
$B(1^{5/2}D_3)$	$\pi$	2	235	1.7	$B(1^{5/2}D_2)$	$\pi$	0	236	51.2					
$B_s(1^{1/2}S_1)$	$K$	2	644	2.3	$B_s(1^{1/2}S_0)$	$K$	2	688	1.9					

TABLE XII. (*Continued*).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$					
$H' = B(3^{1/2}S_0) \quad m = 6.320 \text{ GeV}$														
$B(1^{1/2}S_1)$	$\eta$	1	766	1.1	$B(1^{1/2}S_0)$	$K$	1	641	26.9					
$B(1^{1/2}S_1)$	$\pi$	1	908	49.5	$B(1^{1/2}S_1)$	$K$	1	592	11.5					
$B(1^{1/2}P_0)$	$\pi$	0	569	1.8	$B_s(1^{1/2}S_0)$	$\eta$	1	486	10.1					
$B(1^{3/2}P_2)$	$\pi$	2	562	11.0	$B_s(1^{1/2}S_1)$	$\eta$	1	420	3.4					
$B(2^{1/2}S_1)$	$\pi$	1	363	9.5	$H(n^j l_J) \quad x \quad l_x \quad p_x \quad \Gamma_x/(g_A^8)^2$									
$B(2^{1/2}P_0)$	$\pi$	0	72	3.5	$H' = B_s(1^{3/2}D_1) \quad m = 6.127 \text{ GeV}$									
$B_s(1^{1/2}S_1)$	$K$	1	697	5.1	$B(1^{1/2}S_0)$	$K$	1	607	36.2					
$H' = B(3^{1/2}S_1) \quad m = 6.347 \text{ GeV}$														
$B(1^{1/2}S_0)$	$\pi$	1	970	31.3	$B_s(1^{1/2}S_1)$	$\eta$	1	438	11.5					
$B(1^{1/2}S_1)$	$\pi$	1	932	43.0	$H' = B_s(2^{1/2}P_0) \quad m = 6.264 \text{ GeV}$									
$B(1^{3/2}P_1)$	$\pi$	2	600	9.4	$B(1^{1/2}S_0)$	$K$	0	785	77.7					
$B(1^{3/2}P_2)$	$\pi$	2	587	7.9	$B(1^{3/2}P_1)$	$K$	1	262	20.6					
$B(1^{1/2}P_1)$	$\pi$	0	560	1.7	$B(1^{1/2}P_1)$	$K$	1	164	3.8					
$B(2^{1/2}S_0)$	$\pi$	1	423	9.3	$B_s(1^{1/2}S_0)$	$\eta$	0	652	40.5					
$B(2^{1/2}S_1)$	$\pi$	1	390	10.8	$H' = B_s(2^{3/2}P_1) \quad m = 6.278 \text{ GeV}$									
$B(2^{1/2}P_1)$	$\pi$	0	64	3.0	$B(1^{1/2}S_1)$	$K$	0	755	(*) 0.7					
$B_s(1^{1/2}S_0)$	$K$	1	775	5.0	$B(1^{1/2}S_1)$	$K$	2	755	28.7					
$B_s(1^{1/2}S_1)$	$K$	1	726	5.2	$B(1^{1/2}P_0)$	$K$	1	278	3.3					
$H' = B_s(2^{1/2}S_0) \quad m = 5.985 \text{ GeV}$														
$B(1^{1/2}S_1)$	$K$	1	416	3.15	$B(1^{1/2}P_1)$	$K$	1	201	1.4					
$H' = B_s(2^{1/2}S_1) \quad m = 6.019 \text{ GeV}$														
$B(1^{1/2}S_0)$	$K$	1	517	5.6	$B_s(1^{1/2}S_1)$	$\eta$	2	614	5.8					
$B(1^{1/2}S_1)$	$K$	1	462	4.9	$H' = B_s(2^{3/2}P_2) \quad m = 6.292 \text{ GeV}$									
$B_s(1^{1/2}S_0)$	$\eta$	1	325	0.4	$B(1^{1/2}S_0)$	$K$	2	813	19.2					
$H' = B_s(1^{5/2}D_2) \quad m = 6.095 \text{ GeV}$														
$B(1^{1/2}S_1)$	$K$	3	555	6.62	$B(1^{1/2}S_1)$	$K$	2	769	19.8					
$B_s(1^{1/2}S_1)$	$\eta$	3	370	0.30	$B(1^{1/2}P_1)$	$K$	1	232	4.2					
$H' = B_s(1^{5/2}D_3) \quad m = 6.103 \text{ GeV}$														
$B(1^{1/2}S_0)$	$K$	3	614	5.03	$B_s(1^{1/2}S_0)$	$\eta$	2	683	4.8					
$B(1^{1/2}S_1)$	$K$	3	564	4.17	$B_s(1^{1/2}S_1)$	$\eta$	2	630	4.2					
$B_s(1^{1/2}S_0)$	$\eta$	3	453	0.46	$H' = B_s(2^{1/2}P_1) \quad m = 6.296 \text{ GeV}$									
$B_s(1^{1/2}S_1)$	$\eta$	3	383	0.21	$B(1^{1/2}S_1)$	$K$	0	773	73.8					

TABLE XII. (*Continued*).

$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$B_s(1^{1/2}S_0)$	$\eta$	2	766	2.4
					$B_s(1^{1/2}S_1)$	$\eta$	2	715	1.3
					$B_s(1^{3/2}P_1)$	$\eta$	1	129	0.9
$B(1^{1/2}S_1)$	$K$	4	808	7.07					
$B(1^{3/2}P_1)$	$K$	3	375	0.51					
$B(1^{1/2}P_0)$	$K$	3	366	0.50					
$B(1^{3/2}P_2)$	$K$	3	354	0.22					
$B(1^{1/2}P_1)$	$K$	3	307	0.14					
$B_s(1^{1/2}S_1)$	$\eta$	4	674	1.04					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$
					$H'(=B_s(1^{7/2}F_3))$	$m=6.332$ GeV			
$B(1^{1/2}S_0)$	$K$	4	857	4.5	$B(1^{1/2}S_1)$	$K$	2	853	9.0
$B(1^{1/2}S_1)$	$K$	4	814	4.1	$B(1^{3/2}P_2)$	$K$	1	420	31.6
$B(1^{3/2}P_1)$	$K$	3	383	0.2	$B_s(1^{1/2}S_1)$	$\eta$	2	723	3.3
$B(1^{3/2}P_2)$	$K$	3	363	0.6					
$B(1^{1/2}P_1)$	$K$	3	317	0.3					
$B_s(1^{1/2}S_0)$	$\eta$	4	732	0.8					
$B_s(1^{1/2}S_1)$	$\eta$	4	681	0.6					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$
					$H'(=B_s(1^{7/2}F_4))$	$m=6.337$ GeV			
$B(1^{1/2}S_0)$	$K$	4	857	4.5	$H'(=B_s(3^{1/2}S_0))$	$m=6.421$ GeV			
$B(1^{1/2}S_1)$	$K$	4	814	4.1	$B(1^{1/2}S_1)$	$K$	1	896	16.5
$B(1^{3/2}P_1)$	$K$	3	383	0.2	$B(1^{1/2}P_0)$	$K$	0	489	0.3
$B(1^{3/2}P_2)$	$K$	3	363	0.6	$B(1^{3/2}P_2)$	$K$	2	479	1.7
$B(1^{1/2}P_1)$	$K$	3	317	0.3	$B_s(1^{1/2}S_1)$	$\eta$	1	770	3.9
$B_s(1^{1/2}S_0)$	$\eta$	4	732	0.8					
$B_s(1^{1/2}S_1)$	$\eta$	4	681	0.6					
$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$	$H(n^j l_J)$	$x$	$l_x$	$p_x$	$\Gamma_x/(g_A^8)^2$
					$H'(=B_s(1^{5/2}F_2))$	$m=6.369$ GeV			
$B(1^{1/2}S_0)$	$K$	2	888	6.0	$H'(=B_s(3^{1/2}S_1))$	$m=6.449$ GeV			
$B(1^{1/2}S_1)$	$K$	2	845	3.5	$B(1^{1/2}S_0)$	$K$	1	963	13.2
$B(1^{3/2}P_1)$	$K$	1	428	29.6	$B(1^{1/2}S_1)$	$K$	1	922	16.0
$B(1^{3/2}P_2)$	$K$	1	409	3.0	$B(1^{3/2}P_1)$	$K$	2	531	2.2
					$B(1^{3/2}P_2)$	$K$	2	514	1.7
					$B_s(1^{1/2}S_0)$	$\eta$	1	848	3.9
					$B_s(1^{1/2}S_1)$	$\eta$	1	800	4.1

- [1] H. Cheng, C. Cheung, G. Lin, Y. C. Lin, T. Yan, and H. Yu, Phys. Rev. D **46**, 5060 (1992); G. Burdman and J. F. Donoghue, Phys. Lett. B **280**, 287 (1992).
- [2] UKQCD Collaboration, C. Michael and J. Peisa, Phys. Rev. D **58**, 034506 (1998); A. Ali Khan *et al.*, *ibid.* **62**, 054505 (2000); Joachim Hein *et al.*, *ibid.* **62**, 074503 (2000); R. Lewis and R. M. Woloshyn, Nucl. Phys. B (Proc. Suppl.) **93**, 192 (2001).
- [3] K. Gottfried, Phys. Rev. Lett. **40**, 598 (1978); M. B. Voloshin, Nucl. Phys. **B154**, 365 (1979); T. Yan, Phys. Rev. D **22**, 1652 (1980).
- [4] H. Georgi and A. Manohar, Nucl. Phys. **B234**, 189 (1984).
- [5] J. L. Goity and W. Roberts, Phys. Rev. D **60**, 034001 (1999).
- [6] J. Zeng, J. W. Van Orden, and W. Roberts, Phys. Rev. D **52**, 5229 (1995).
- [7] E. J. Eichten, C. T. Hill, and C. Quigg, Phys. Rev. Lett. **71**, 4116 (1993).
- [8] S. Godfrey and N. Isgur, Phys. Rev. D **32**, 189 (1985).
- [9] G. P. Lepage, “How to renormalize the Schrödinger equation,” nucl-th/9706029.
- [10] M. Di Pierro and E. Eichten, Nucl. Phys. B (Proc. Suppl.) **93**, 130 (2001).
- [11] Particle Data Group, D. E. Groom *et al.*, Eur. Phys. J. C **15**, 1 (2000).
- [12] L3 Collaboration, M. Acciarri *et al.*, Phys. Lett. B **465**, 323 (1999).
- [13] OPAL Collaboration, G. Abbiendi *et al.*, hep-ex/0010031.
- [14] CLEO Collaboration, S. Anderson *et al.*, Nucl. Phys. **A663**, 647 (2000).
- [15] DELPHI Collaboration, P. Abreu *et al.*, Phys. Lett. B **426**, 231 (1998).
- [16] CLEO Collaboration, J. L. Rodriguez, hep-ex/9901008.
- [17] OPAL Collaboration, G. Abbiendi *et al.*, Eur. Phys. J. C **20**, 445 (2001).
- [18] CDF Collaboration, G. Bauer, hep-ex/9909014.
- [19] ALEPH Collaboration, R. Barate *et al.*, Phys. Lett. B **425**, 215 (1998).
- [20] H. J. Schnitzer, Phys. Lett. **76B**, 461 (1978).
- [21] N. Isgur, Phys. Rev. D **57**, 4041 (1998).
- [22] D. Ebert, V. O. Galkin, and R. N. Faustov, Phys. Rev. D **57**, 5663 (1998).
- [23] R. Casalbuoni, A. Deandrea, N. Di Bartolomeo, R. Gatto, F. Feruglio, and G. Nardulli, Phys. Rep. **281**, 145 (1997), and references therein.
- [24] G. M. De Divitiis *et al.*, J. High Energy Phys. **10**, 010 (1998).
- [25] S. J. Dong, J.-F. Lagae, and K. F. Liu, Phys. Rev. Lett. **75**, 2096 (1995).
- [26] F. E. Close and S. G. Roberts, Phys. Lett. B **316**, 165 (1993).
- [27] CLEO Collaboration, T. E. Coan *et al.*, hep-ex/0102007.
- [28] Edgard Elbaz, *Quantum: The Quantum Theory of Particles, Fields, and Cosmology*, Texts and Monographs in Physics (Springer-Verlag, Berlin, 1998).



ELSEVIER

Nuclear Physics B (Proc. Suppl.) 93 (2001) 130–133

Nuclear Physics B  
PROCEEDINGS  
SUPPLEMENTS  
[www.elsevier.nl/locate/npc](http://www.elsevier.nl/locate/npc)

## Hadronic Decays of Excited Heavy Mesons

M. Di Pierro\* and E. Eichten  
Fermilab, Batavia, IL 60510, USA

We studied the hadronic decays of excited states of heavy mesons ( $D$ ,  $D_s$ ,  $B$  and  $B_s$ ) to lighter states by emission of  $\pi$ ,  $\eta$  or  $K$ . Wavefunctions and energy levels of these excited states are determined using a Dirac equation for the light quark in the potential generated by the heavy quark (including first order corrections in the heavy quark expansion). Transition amplitudes are computed in the context of the Heavy Chiral Quark Model.

### 1. INTRODUCTION

In the context of the most general Quark Model, a heavy-light meson ( $H$ ) is modeled with a light quark ( $q$ ) bound to a static source of chromo-electro-magnetic field (the heavy quark  $h$ ). Its hadronic transitions can be computed assuming that only the light quark enters in the reaction through an effective coupling  $g_A$  (Heavy Chiral Quark Model [1]).

Our work generally follows that of ref. [2–5]. We differ in the choice of the potential. Moreover, we included for the first time the mixing effects in the spectrum and decay amplitudes.

### 2. SPECTRUM

#### 2.1. Notation

The Dirac wavefunction,  $\Psi$ , of the light quark can be determined by solving the eigenvalue problem

$$H\Psi_{n,\ell,j,J,M} = E_{n,\ell,j,J,M}\Psi_{n,\ell,j,J,M} \quad (1)$$

where  $H$  is the Hamiltonian of the system and  $\Psi$  is 4-spinor that represents the wavefunction of the system. In our notation

$$\Psi_{n\ell j JM} = C_{j,m;\frac{1}{2},S}^{J,M} \xi_S \begin{pmatrix} if_{n\ell j}^0 & k_{\ell,j,m}^+ & Y_{m-\frac{1}{2}}^\ell \\ if_{n\ell j}^0 & k_{\ell,j,m}^- & Y_{m+\frac{1}{2}}^\ell \\ f_{n\ell j}^1 & k_{2j-\ell,j,m}^+ & Y_{m-\frac{1}{2}}^{2j-\ell} \\ f_{n\ell j}^1 & k_{2j-\ell,j,m}^- & Y_{m+\frac{1}{2}}^{2j-\ell} \end{pmatrix} \quad (2)$$

where  $n$  is the radial quantum number,  $\ell$  is the orbital quantum number,  $j, m$  are the total spin

\*Talk presented by

of the light quark and its  $z$  component,  $J, M$  are the total spin of the meson and its  $z$  component,  $f^0, f^1$  are radial wavefunctions and  $Y_m^\ell$  are the usual spherical harmonics.  $\xi_S$  is the 2-spinor associated to the heavy quark  $h$  and  $S$  is its spin.

Our convention for the phase and the normalization is such that

$$k_{\ell,j,m}^\pm = \begin{cases} \sqrt{\frac{\ell \pm m + \frac{1}{2}}{2\ell + 1}} & \text{if } j = \ell + \frac{1}{2} \\ \pm \sqrt{\frac{\ell \mp m + \frac{1}{2}}{2\ell + 1}} & \text{if } j = \ell - \frac{1}{2} \end{cases} \quad (3)$$

#### 2.2. Choice of the potential

Ignoring  $1/m_h$  corrections, the most general form of the Hamiltonian that appear in eq. (1) is

$$H^{(0)} = -i\gamma^0\gamma^i\partial_i + \gamma^0m_q + \gamma^0V_s + M_h + V_v \quad (4)$$

where  $V_s$  is a spin independent potential,  $V_v$  is a spin dependent potential and  $M_h$  is a total energy shift (not to be confused with the mass of the heavy quark  $m_h$  that appear in corrections to  $H = H^{(0)} + O(1/m_h)$ ).

Asymptotic freedom suggests that at short distances  $V_v \simeq 1/r$  dominates, while lattice simulations suggest that at large distances  $V_s \simeq r$  dominates. As it was observed in ref. [2], the choice of Coulomb-like potential at short distance is inconsistent with  $1/m_h$  spin-dependent correction to  $H^0$ , because of ultraviolet divergences. The solution of the problem is that, in the context of the Dirac equation with a finite  $m_h$ , it is not correct to localize the heavy quark with a delta function since one must take into account the spatial degrees of freedom of the heavy quark. Our pragmatic approach to the problem is that of delocalizing the heavy quark within a length scale  $1/\lambda$

assuming a Gaussian wave-function,  $\Phi(\mathbf{x})$ , for the former. The effective potential felt by the light quark is, therefore, a convolution of the Coulomb-like potential with the square of the wave-function of the quark:

$$V_v(r) = \int |\Phi(\mathbf{x})|^2 \frac{\alpha_s}{|\mathbf{r} - \mathbf{x}|} d^3\mathbf{x} = \frac{\alpha_s}{r} \text{erf}(\lambda r) \quad (5)$$

Our choice for the spin-independent part of the potential is

$$V_s(r) = br + c \quad (6)$$

(notice from eq. (4) that  $c$  is not a physical parameter since it can be re-absorbed into the definition of  $m_q$ ).

### 2.3. $1/m_h$ correction

We solve the eigenvalue problem associated to the eq. (1) using the Hamiltonian in eq. (4) and the potentials (5,6). In this way we determine the radial wave-functions  $f_{n\ell}^0$ ,  $f_{n\ell}^1$ , and the associated eigenvalues  $E_{n\ell,j}$ .

$1/m_h$  corrections to the the Hamiltonian have been derived in [4] in the Bethe-Salpeter formalism. They are responsible for the spin-orbit interaction, the hyperfine splitting and the mixing of states with the same  $j$ . We include these effects as a perturbation to the energy levels ( $\delta E_{n\ell,j}$ ) and also determine the mixing coefficients for each doublet of states.

In terms of the  $f^i$  the  $1/m_h$  correction to the energy levels reads as

$$\delta E = \int (A + B + C)r^2 dr \quad (7)$$

where one can identify the contributions proportional to  $p^2$ :

$$A = - \sum_i f^i \left( \partial_r^2 + \frac{2}{r} \partial_r - \frac{\ell^2 + \ell}{r^2} \right) f^i \quad (8)$$

the spin orbit interaction, for  $j = \ell + \frac{1}{2}$ :

$$B = V_v \left[ f^1 \left( \partial_r - \frac{\ell}{r} \right) f^0 - f^0 \left( \partial_r + \frac{\ell+2}{r} \right) f^1 \right] \quad (9)$$

for  $j = \ell - \frac{1}{2}$ :

$$B = V_v \left[ f^1 \left( \partial_r + \frac{\ell+1}{r} \right) f^0 - f^0 \left( \partial_r - \frac{\ell-1}{r} \right) f^1 \right] \quad (10)$$

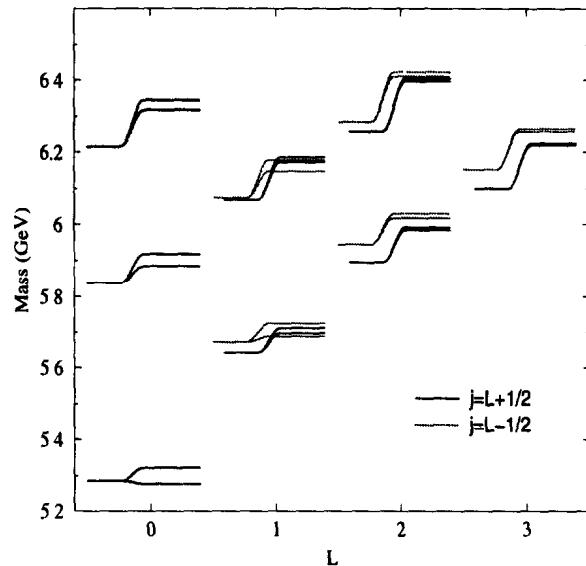


Figure 1. Spectrum of excited  $B$  mesons.

Particle	$n^j \ell_J$	Exp.	Model	$\Gamma_{\text{tot}}/g_A^2$
$D$	$1\frac{1}{2}S_0$	1864	1871	—
$D^*$	$1\frac{1}{2}S_1$	2007	2006	—
$D_1$	$1\frac{3}{2}P_1$	2427	2420	25.0
$D_2^*$	$1\frac{3}{2}P_2$	2459	2462	38.2
$D_s$	$1\frac{1}{2}S_0$	1969	1965	—
$D_s^*$	$1\frac{1}{2}S_1$	2114	2112	—
$D_{s1}$	$1\frac{3}{2}P_1$	2535	2535	94.3
$D_{s2}^*$	$1\frac{3}{2}P_2$	2573	2579	4.9
$B$	$1\frac{1}{2}S_0$	5279	5278	—
$B^*$	$1\frac{1}{2}S_1$	5325	5322	—
$B_0$	$1\frac{1}{2}P_0$		5689	174.6
$B_1$	$1\frac{3}{2}P_1$		5698	14.3
$B_1^*$	$1\frac{1}{2}P_1$		5725	168.0
$B_2^*$	$1\frac{3}{2}P_2$		5712	20.0
$B'$	$2\frac{1}{2}S_0$		5885	28.1
$B'^*$	$2\frac{1}{2}S_1$		5918	49.3
$B_s$	$1\frac{1}{2}S_0$	5369	5369	—
$B_s^*$	$1\frac{1}{2}S_1$	5416	5417	—
$B_{s1}$	$1\frac{3}{2}P_1$		5801	—
$B_{s2}^*$	$1\frac{3}{2}P_2$		5815	0.2

Table 1. Tabulated spectrum for the observed states used in the fit together with predictions for some excited  $B$  states. All units are in MeV.

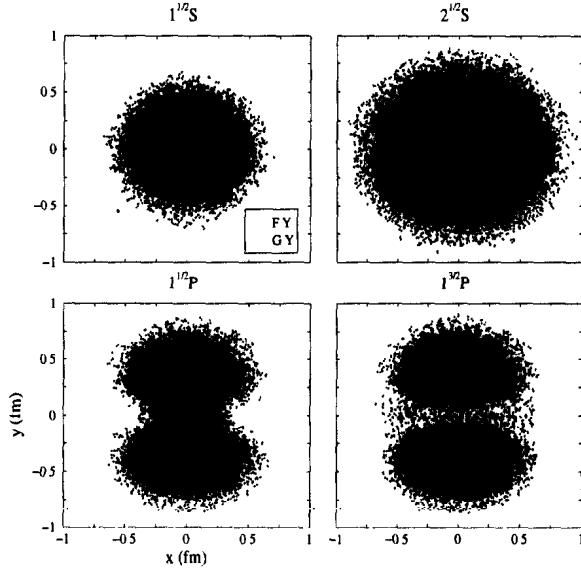


Figure 2. Orbitals for some excited B mesons.

and the hyperfine splitting contribution:

$$C = (-1)^{J-j} \frac{2j+1}{2J+1} V'_v f^0 f^1 \quad (11)$$

#### 2.4. Choice of parameters and predictions

The nine parameters of the model ( $\alpha_s$ ,  $\lambda$ ,  $b$ ,  $m_q|_{q=u}$ ,  $m_q|_{q=s}$ ,  $m_h|_{h=c}$ ,  $M_h|_{h=c}$ ,  $m_h|_{h=b}$ ,  $M_h|_{h=b}$ ) are determined by best fit (minimum  $\chi^2$ ) to the known spectrum of excited states of the  $D$ ,  $D_s$ ,  $B$  and  $B_s$  mesons. Our results are:

$$\begin{aligned} \alpha_s &= 0.339 \\ \lambda &= 2.820 \text{ GeV} \\ b &= 0.257 \text{ GeV} \\ m_u &= 0.073 \text{ GeV} \quad m_s = 0.214 \text{ GeV} \\ m_c &= 1.52 \text{ GeV} \quad M_c = 1.52 \text{ GeV} \\ m_b &= 4.67 \text{ GeV} \quad M_b = 4.68 \text{ GeV} \end{aligned}$$

The spectrum for some of the states plotted in fig. 1 and tabulated in table 1. A density plot of  $|f_{nlj}^0 Y_0^\ell|^2$  and  $|f_{nlj}^1 Y_0^{2J-\ell}|^2$  for some excited states of a B meson is reported in fig. 2.

#### 2.5. Tests of the model

Despite the good fit of the mass spectrum we decided to test our model by comparing some of

the transition amplitudes with some recent lattice results. In particular we computed

$$\mathcal{A}_{B^* \rightarrow B\pi}(r) = \int \langle B^* | A_\mu(r) | B \rangle d\Omega_r \quad (12)$$

using our chiral quark model (where the only unknown parameter is the overall normalization,  $g_A$ , which is the effective coupling of the quark to the axial current,  $A_\mu$ ) and comparing it with the lattice result of ref. [6]. The comparison is shown in fig. 3. In the plot the point at  $r = 0$  is used to fix the relative normalization. A more sophisticated analysis of the lattice results gives<sup>2</sup>  $g_A = 0.42 \pm 0.09$ .

### 3. HADRONIC DECAYS

#### 3.1. Decay amplitudes

We consider here the most general hadronic transition

$$H' \rightarrow H + x \quad (13)$$

where  $H'$  is an heavy meson with associated wavefunction  $\Psi'$ ,  $H$  is an heavy meson with associated wavefunction  $\Psi$  and  $x$  is a light meson with momentum  $p$ . Such a transition is mediated by a matrix element of the form

$$I(X, p) = g_A \int \bar{\Psi}' X e^{-ipr} \Psi' d^3r \quad (14)$$

where  $X$  is the  $4 \times 4$  spin matrix that characterize the transition. In the particular case in which the light meson  $x$  is a pseudoscalar ( $\pi$ ,  $\eta$  or  $K$ )  $X = \not{p}\gamma^5$ , while if the light meson  $x$  is a vector ( $\rho$ ,  $\omega$  or  $K^*$ )  $X = \not{\epsilon}$  (and  $\epsilon$  is the polarization vector of the outcoming vector meson).

The exponential can be expanded in products of spherical harmonics and spherical Bessel functions ( $j_k$ ), thus giving

$$I(X, p) = \sum_{\ell_x, m_x} Y_{m_x}^{\ell_x *}(\hat{p}) C_{J, M; \ell_x, m_x}^{J', M'} \mathcal{A}_{\ell_x}(X, p) \quad (15)$$

We computed  $\mathcal{A}_{\ell_x}(X, p)$  for a complete set of spin matrices  $X$  and proved that it can alway be reduced to integrals of the form

$$\mathcal{A}_{\ell_x}(X, p) = g_A \sum_{i,j=0,1} c_{\ell_x}^{ijk}(X) \int_0^\infty (f^i j_k f^j) r^2 dr \quad (16)$$

<sup>2</sup>this is a preliminary determination valid in the limit  $p_\pi, m_\pi \rightarrow 0$

where  $c_{\ell_x}^{ijk}(X)$  depend also on the quantum numbers of the mother and the daughter mesons but not on the radial wavefunctions.

### 3.2. Decay widths

The decay width for transition of eq. (13), when light meson  $x$  is emitted in an eigenstate of the total momentum  $p$  and of the angular momentum  $\ell_x$ , is given by

$$\Gamma_{\ell_x}^{H'Hx} = \frac{\zeta^2}{16\pi^2 f_\pi^2} \frac{2J+1}{2J'+1} \frac{M}{M'} |\mathcal{A}_{\ell_x}(p\gamma^5, p)|^2 \quad (17)$$

where  $\zeta = \sqrt{3}, 1/\sqrt{3}, 2/\sqrt{3}$  or 1 for  $\pi, \eta$  (for a nonstrange heavy meson),  $\eta$  (for a strange heavy meson) or  $K$  respectively;  $M', J'$  and the mass and total angular momentum of the mother meson  $H'$ ;  $M, J$  are the mass and angular momentum of the daughter meson  $H$ , and  $p$  can be determined by energy-momentum conservation ( $p_0 = M' - M, p = |\mathbf{p}| = \sqrt{p_0^2 - m_x^2}$ ). The total decay width and the branching ratios are defined as

$$\Gamma_{\text{tot}}^{H'} = \sum_{x, \ell_x} \Gamma_{\ell_x}^{H'Hx}; \quad Br(H' \rightarrow Hx; \ell_x) = \frac{\Gamma_{\ell_x}^{H'Hx}}{\Gamma_{\text{tot}}^{H'}} \quad (18)$$

where the sum on  $x$  spans all the hadronic decay modes with emission of a light pseudoscalar meson. The total width for some of the mesons is reported in table 1.

## 4. CONCLUSIONS

We computed the spectrum and the width of hadronic decays of excited  $D, D_s, B$  and  $B_s$  mesons in the context of the chiral quark model. As an example, we report here some hadronic decay channels for the first radial excited  $B$  meson (not including the  $\rho$  decays)

$$\begin{aligned} B(2\frac{1}{2}S_0) &\rightarrow B(1\frac{1}{2}S_1) + \pi \quad (\ell_\pi = 1 \quad Br = 77\%) \\ &\rightarrow B(1\frac{1}{2}P_0) + \pi \quad (\ell_\pi = 0 \quad Br = 22\%) \\ &\rightarrow B(1\frac{1}{2}S_0) + \pi \quad (\ell_\pi = 0 \quad Br = 100\%) \\ &\rightarrow B(1\frac{3}{2}P_2) + \pi \quad (\ell_\pi = 2 \quad Br = 0.33\%) \\ &\rightarrow B(1\frac{1}{2}S_1) + \eta \quad (\ell_\eta = 1 \quad Br = 0.03\%) \end{aligned}$$

More complete tables will be published on a separate paper.

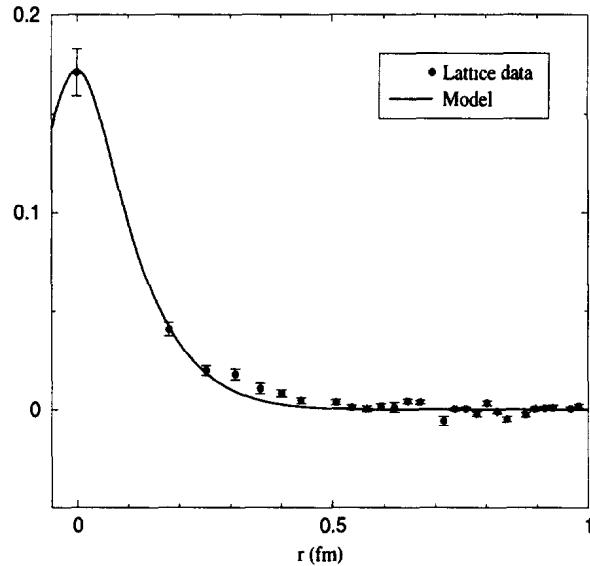


Figure 3. Comparison between a prediction of our model and the lattice QCD result [6].

We finally remark how our model is able to fit the masses of observed excited states within less than 10 MeV discrepancy (within 4 MeV in average) better than was done in preceding works.

This work was performed at Fermilab, a U.S. Department of Energy Lab (operated by the University Research Association, Inc.), under contract DE-AC02-76CHO3000.

## REFERENCES

1. H. Georgi and A. Manohar, Nucl. Phys. **B234** (1984) 189
2. S. Godfrey and N. Isgur, Phys. Rev. **D32** (1985) 32
3. E.J. Eichten, C.T. Hill and C. Quigg, Phys. Rev. Lett. **71** (1993) 4116
4. J. W. Van Orden, J. Zeng and W. Roberts, Phys. Rev. **D52** (1995) 5229
5. J. L. Goity and W. Roberts, Phys. Rev. **D60** (1999) 034001
6. G. M. De Divitiis *et al.*, JHEP 9810:010 (1998)



# Matrix distributed processing: a set of C++ tools for implementing generic lattice computations on parallel systems <sup>☆,☆☆</sup>

Massimo Di Pierro

Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA

Received 26 April 2001; received in revised form 26 May 2001

---

## Abstract

We present a set of programming tools (classes and functions written in C++ and based on Message Passing Interface) for fast development of generic parallel (and non-parallel) lattice simulations. They are collectively called MDP 1.2.

These programming tools include classes and algorithms for matrices, random number generators, distributed lattices (with arbitrary topology), fields and parallel iterations. No previous knowledge of MPI is required in order to use them.

Some applications in electromagnetism, electronics, condensed matter and lattice QCD are presented. © 2001 Elsevier Science B.V. All rights reserved.

**Keywords:** Lattice; Parallel computing; Numerical software

---

## PROGRAM SUMMARY

*Title of the library:* MDP, Matrix Distributed Processing (version 1.2)

*Catalogue identifier:* ADPA

*Program obtainable from:* CPC Program Library, Queen's University of Belfast, N. Ireland

*Program Summary URL:* <http://cpc.cs.qub.ac.uk/summaries/ADPA>

*Computer for which the library is designed:* Any computer or parallel computer, including clusters of workstations

*Software required in order to run MDP:* MPI (Message Passing Interface) is required to run MDP in parallel. MPI is not required to run MDP in single process mode

*Computer(s) on which the library has been tested:* SUN SparcSTATION with Solaris, Linux PCs, cluster of PCs (connected by Ethernet and Myrinet). The code has also been tested partially on a Cray T3E

*Programming language used:* ANSI C++, to be compiled with g++ -ansi -pedantic [input program] -o [output program]

<sup>☆</sup> The realization of this paper and of the software FermiQCD was performed at Fermilab, a U.S. Department of Energy Lab (operated by the University Research Association, Inc.), under contract DE-AC02-76CHO3000.

<sup>☆☆</sup> This program can be downloaded from the CPC Program Library under catalogue identifier: <http://cpc.cs.qub.ac.uk/summaries/ADPA>

*E-mail address:* mdp@fnal.gov (M. Di Pierro).

No. of bytes in distributed software, including test data, etc.:  
2 500 222

Distribution format: tar gzip file

*Nature of the physical problem*

Any problem that can be described in terms of interacting fields discretized on a lattice of some arbitrary shape and topology

*Parallel applications provided together with the library as examples*

A program that solves electrostatic problems (*application1.C*).

A program that computes total impedance in a net of resistors (*application2.C*). An Ising model simulation (*application3.C*). A parallel implementations of the Vegas multidimensional integration algorithm (*MDP\_PVegas.h*, not described here). A complete Lattice QCD package (*FermiQCD*, not described here)

*Web site:* <http://www.phoenixcollective.org/mdp/mdp.html>

## LONG WRITE-UP

### 1. Introduction

One of the biggest discoveries of modern physics is that long distance forces are mediated by fields and these fields are subject to local interactions. Local differential equations, in some field variables, seem to describe any aspect of fundamental physics (from particle physics to gravitation, from thermodynamics to condensed matter).

Often the equations describing complex interacting systems cannot be solved exactly and require a numerical approach, i.e. discretize on a lattice the space on which the equations are defined. The derivative terms of a local equation, once discretized, give origin to quasi-local interaction terms on the lattice.

The good thing of having to deal with quasi-local equations is that one can partition the problem by partitioning the lattice (the space) on which the problem is defined, and only a minimal set of communications is required to pass information about the field variables at the boundary of these partitions (because first and second derivatives of the differential equations in the continuum correspond to nearest neighbor interaction terms on the lattice).

Parallel computers and clusters are the natural hardware for implementing the numerical algorithms that can solve this kind of problems.

MDP is a set of C++ tools for fast development of any kind of parallel numerical computations that involve the lattice discretization of a finite portion of space (and/or time) over a number of parallel processors (supporting Message Passing Interface). It considerably reduce that development costs since one can concentrate on the algorithm instead of focusing on communications.

A number of tricks and optimizations are implemented that make of MDP 1.2 an efficient package for developing, among others, fast and robust Lattice QCD applications.<sup>1</sup>

#### 1.1. Content

The main characteristics of MDP 1.2 are:

- It includes **Matrix**, a class to manipulate matrices of complex (**Complex**) numbers. The assignment operator and the copy constructor have been overloaded to optimize the speed and the memory usage. In fact a **Matrix** object contains a dynamically allocated pointer to the location of memory where the matrix elements are stored. The matrix elements are automatically copied only when it is inevitable, otherwise their memory address is copied. In particular, the temporary **Matrix** object returned by a function stores its elements in the same location of memory used by the function that will take that object as argument. This class is used to implement an interface to the fields.

---

<sup>1</sup> For example, MDP 1.2 has been used here at Fermilab to develop, in a relatively short time, a parallel software for Lattice QCD simulations (see Section 5.5).

- All the calls to MPI are done through a wrapper object, `mpi`, which takes care of generating communication tags and computing the length of the objects to be sent/received by the parallel processes.
- The most important classes in MDP 1.2 are

```
generic_lattice
site
generic_field
```

They enable the user to define lattices of any size, shape and topology and associate any kind of structure to the sites. Other properties of these classes are: automatic communications between the parallel processes (when necessary), parallel I/O for the field variables and functions to move on the lattice.

- For each user defined lattice, one random generator per site is automatically created and is a member variable of `generic_lattice`. This guarantees that any simulation will give the same results whether it runs on a single PC or in parallel, independently of the way the lattice is partitioned. The random generator itself can be seen as a field and it can be saved and restored at any point.
- Each lattice (and each field defined on that lattice) can be partitioned arbitrarily among the different processes (or automatically if the user does not bother to do it) and the constructor of `generic_lattice`, for each process, takes care of allocating the memory for storing the neighbor sites that have to be passed by a neighbor process. The code has been optimized in order to minimize the need for communications and the low level optimization tricks are completely hidden to the high level programmer.
- The same code can run on single process on one PC or multiprocess on a cluster of PCs (or on a supercomputer supporting MPI), without any modification. The results are independent from the number of processes.<sup>2</sup>

As an example of what one can do with MDP 1.2 we present a parallel program that creates a field of  $5 \times 5$  matrices on a  $4 \times 4 \times 4$  lattice (distributed on 4 processors) and sets each matrix (for each site) equal to the inverse of the exponential of a random SU(5) matrix (just to make things a little complicated!). Then it averages each matrix with its 6 nearest neighbors.

```
00: // Program: example01.C
01: #define PARALLEL
02: #define Nproc 4
03: #include "MDP_Lib2.h"
04: #include "MDP_MPI.h"
05:
06: int main(int argc, char **argv) {
07:   mpi.open_wormholes(argc, argv);
08:   int ndim=3;
09:   int mybox[]={4,4,4};
10:   generic_lattice mylattice(ndim, mybox);
11:   Matrix_field F(mylattice,5,5);
12:   site x(mylattice);
13:   forallsites(x)
14:     F(x)=inv(exp(mylattice.random(x).SU(5)));
15:   F.update();
16:   forallsites(x) {
17:     F(x)=1.0/7.0*(F(x)+F(x+0)+F(x-0)+F(x+1)+F(x-1)+F(x+2)+F(x-2));
18:     printf("F(x) for x=(%i,%i)\n", x(0), x(1));
19:     print(F(x));
```

---

<sup>2</sup> Apart for rounding errors that usually are compiler and hardware dependent.

```

20:      } ;
21:      mpi.close_wormholes( ) ;
22:      return 0 ;
23:  } ;

```

Note that each line has a meaning and none of them is superfluous:

- Line 1 tells that the job will run in parallel.
- Line 2 tells that it will run on 4 processes (this statement is optional since the number of processes can be determined at runtime).
- Line 3 and 4 includes the libraries of MDP 1.2.
- Line 7 establishes the communications among the processes.
- Line 8 sets a variable containing the dimensions of the lattice.
- Line 9 fills an array containing the size of the lattice.
- Line 10 defines the lattice, `mylattice`.
- Line 11 defines a field of  $5 \times 5$  matrices `F` on `mylattice` (`Matrix_field` is built-in field based on `generic_field`).
- Line 12 defines a site variable `x` on `mylattice`.
- Lines 13, 14 initialize each field variable `F(x)` with the inverse of the exponential of a random SU(5) (generated using the local random generator of site `x`).
- Line 15 performs all the communication to update boundary sites, i.e. sites that are shared by different processes. They are determined automatically and optimally when the lattice is declared.
- Lines 16, 17 average `F(x)` with its 6 next neighbor sites.
- Lines 18, 19 print out the results.
- Line 21 closes all the communications.

The source code of MDP 1.2 is contained in the files

- `MDP_Lib2.h`,
- `MDP_MPI.h`,
- `MDP_ANSI.h`,
- `MDP_Measure.h`,
- `MDP_Fit.h`,
- `MDP_PVegas.h`,
- `MDP_Prompt.h`,
- `MDP_utils.h`,

plus some examples and applications. Only the classes defined in the first two files will be discussed here.

A reader interested only in parallelization issues can jump directly to Section 4, and go back to Sections 2 and 3 later.

## 2. Nonparallel tools

Four basic classes are declared in the file `MDP_Lib2.h`:<sup>3</sup>

- `myreal`. It is equivalent to `float` unless the user defines the global constant `USE_DOUBLE_PRECISION`. In this case `myreal` stands for `double`.
- `Complex`. Declared as `complex<myreal>`. The imaginary unit is implemented as a global constant `I=Complex(0,1)`.

---

<sup>3</sup> The classes defined in the file `MDP_Lib2.C` have extensively been used in some analysis programs written by the Southampton Theory Group for Lattice QCD applications.

- `DynamicArray<object ,n>`. It is a container for n-dimensional arrays of objects (for examples arrays of `Matrix`). `DynamicArrays` can be resized any time, moreover they can be passed to (and returned by) functions.
  - `Matrix`. An object belonging to this class is effectively a matrix of `Complex` numbers and it may have arbitrary dimensions.
  - `Random_generator`. This class contains the random number generator and member functions to generate random `float` numbers with uniform distribution, Gaussian distribution or any user defined distribution. It can also generate random  $SU(n)$  matrices. Different `Random_generator` objects can be declared and initialized with different seeds.
  - `JackBoot`. It is a class to store sets of data to be used for computing the average of any user defined function acting on the data. It includes member functions for computing Jackknife and Bootstrap errors on this average.
- Moreover the following constants are declared

```
#define and          &&
#define or           ||
#define Pi            3.14159265359
#define I             Complex(0,1)
#define PRECISION    1e-16
#define CHECK_ALL
#define TRUE   1
#define FALSE  0
```

If the definition of `CHECK_ALL` is removed the code will be faster but some checks will be skipped resulting a lesser safe code.

### 2.1. On `DynamicArray` and `Matrix`

The main idea on which these two classes have been built is the solution of the problem of returning objects containing pointers to dynamically allocated memory [1].

Consider the following code involving matrices:

```
Matrix A,B(10,10);
A=B;
A=B*A;
```

In the first assignment one wants each element of `B` to be copied in the corresponding element of `A`, while the second assignment is faster if `B` and `A` are passed to the local variables of the `operator*` by reference (i.e. without copying all the elements). Moreover one wants the local variable created by the `operator*` to occupy the same memory location as the variable that is returned (this avoids copying and wasting space). To implement this idea each `Matrix` object contains a `FLAG` and a pointer to dynamically allocated memory (where the numbers are stored). The copy constructor and the `operator=` have been overloaded in such a way to take care of the status of the `FLAG` and eventually to copy the pointer to the memory, instead of copying the memory containing the real matrix.

A physical location of memory may be pointed by different `Matrix` objects, but this never generates confusion if a few safety rules, stated later, are followed. An automatic system of garbage collecting deallocates the unused memory when there are no objects alive pointing to it.

In this way, in the first assignment of the example,  $11 + 800$  bytes<sup>4</sup> are copied, while in the second assignment only 11 bytes are copied three times (when passing `A` and `B` to the `operator*()` and when returning the result)

---

<sup>4</sup> The number 11 is the size in bytes of a `Matrix` object. It is independent from the size of the memory occupied by the “real” matrix.

to be compared with the 800 bytes of the matrix. The pointer of A is swapped only when the multiplication is terminated without generating confusion between the input and the output of the function. The FLAG takes care of everything and the procedure works also in any recursive expression. For big matrices this is, in principle, faster than it would be possible in FORTRAN or C. (In FORTRAN, for example, it would be necessary to create a temporary array where to store the result of the multiplication and copy it into A.) In practice this is not true because the class Matrix uses dynamic allocation and modern compilers are not able to optimize its algorithms as well as they can optimize algorithms using normal arrays.

We do not suggest to use the classes DynamicArray and Matrix to implement those parts of the user's algorithms that are critical for speed. Despite this, they can be used to implement an efficient and practical interface to field variables (as shown in Section 4.3.4).

## 2.2. Safety rules

The safety rules for DynamicArray and Matrix objects are:

- Always, before returning an *object*, call `prepare(object)`;
- Never explicitly call the copy constructor.
- Do not call the assignment operator of the argument of a function within the function itself.

## 2.3. Using DynamicArray

DynamicArrays are declared using templates. For example the command

```
DynamicArray<float, 4> myarray(2, 3, 6, 5);
```

declares a  $2 \times 3 \times 6 \times 5$  array of float called myarray. The first argument of the template specifies the type of object (i.e. float) and the second argument is the number of dimensions of the array (i.e. 4). An array of this kind can be resized at any time. For example,

```
myarray.dimension(4, 3, 2, 2);
```

transforms myarray in a new  $4 \times 3 \times 2 \times 2$  array of float. A DynamicArray can have up to 10 dimensions. The array elements can be accessed by reference using `operator()` in the natural way

```
myarray(i, j, k, l)
```

where i, j, k, l are integers. The total number of dimensions (in this example 4) is contained in the member variable

```
myarray.ndim
```

Two useful member functions are

```
myarray.size()
```

that returns the total number of elements of the array (as long) and

```
myarray.size(n)
```

that returns the size of the dimension n of the array (as long). The argument n must be an integer in the range [0, ndim - 1].

DynamicArrays can be passed to function and can be returned by functions but, before a DynamicArray is returned the function, `prepare` must be called. Here is an example of a program that returns a DynamicArray.

```
// Program: example02.C
#include "MDP_Lib2.h"

DynamicArray<float,2> f(float x) {
    DynamicArray<float,2> a(2,2);
    a(0,0)=x;
    prepare(a); // IMPORTANT !!
    return a;
};

int main() {
    float x=f(Pi)(0,0);
    printf("%f\n", x);
    return 0;
};
```

It prints 3.141593. Note that `f()` returns a  $2 \times 2$  Matrix. The following is a program that contains a function that takes a DynamicArray as argument

```
// Program: example03.C
#include "MDP_Lib2.h"

DynamicArray<float,2> f(float x) {
    DynamicArray<float,2> a(2,2);
    a(0,0)=x;
    prepare(a);
    return a;
};

void g(DynamicArray<float,2> a) {
    printf("%f\n", a(0,0));
};

int main() {
    DynamicArray<float,2> a;
    a=f(Pi);
    g(a);
    return 0;
};
```

The output of this program is again 3.1415. Note that when `f()` is assigned to `a`, the latter is automatically resized.

Bare in mind that even if the function `g()` does not take its arguments by reference the object `DynamicArray` contains a pointer to the memory where the real data are stored, therefore it is *as if* it was passed by reference. This does not generate confusion providing one follows the three safety rules.

#### 2.4. Using Matrix

A Matrix object, say `M`, is very much like `DynamicArray<Complex,2>`. A Matrix can be declared either by specifying its size or not

```
Matrix M(r,c);      // r rows times c columns
Matrix M;           // a general matrix
```

Even if the size of a matrix has been declared it can be changed anywhere with the command

```
M.dimension(r,c); // r rows times c columns
```

Any `Matrix` is automatically resized, if necessary, when a value is assigned to it. The following lines

```
Matrix M(5,7), A(8,8);
M=A;
printf("%i,%i\n", M.rowmax(),M.colmax());
```

prints 8,8. The member functions `rowmax()` and `colmax()` return, respectively, the number of rows and columns of a `Matrix`.

The element  $(i, j)$  of a matrix `M` can be accessed with the natural syntax

```
M(i,j)
```

where `i, j` are integers. Moreover the class contains functions to perform standard operations among matrices:

```
+, -, *, /, +=, -=, *=, /=, inv, det, exp, sin, cos, log,
transpose, hermitian, minor, identity,....
```

As an example a program to compute

$$\left[ \exp \begin{pmatrix} 2 & 3 \\ 4 & 5i \end{pmatrix} \right]^{-1} \quad (1)$$

looks like this:

```
// Program: example04.C
#include "MDP_Lib2.h"
int main() {
    Matrix a(2,2);
    a(0,0)=2; a(0,1)=3;
    a(1,0)=4; a(1,1)=5*I;
    print(inv(exp(a)));
    return 0;
}
```

It is straightforward to add new functions copying the following prototype

```
Matrix f(const Matrix a, const Matrix b, ...) {
    Matrix M;
    // body,
    prepare(M);
    return M;
}
```

We repeat again the three safety rules one should always remember: one should call `prepare()` before returning a `Matrix`; one should not return a `Matrix` by reference; one should not initialize a `Matrix` using the copy constructor nor call the assignment operator of a `Matrix` argument.

Here is one more example of how to use the class `Matrix`:

```
// Program: example05.C
#include "MDP_Lib2.h"
Matrix cube(Matrix X) {
    Matrix Y;
    Y=X*X*X;
    prepare(Y);
    return Y;
}
int main() {
    Matrix A,B;
    A=Random.SU(3);
    B=cube(A)*exp(A)+inv(A);
    print(A);
    print(B);
    return 0;
}
```

This code prints on the screen a random SU(3) matrix  $A$  and  $B = A^3 e^A + A$ . Some example statements are listed in Table 1. Note that the command

`A=mul_left(B,C);`

is equivalent but faster than

`A=C*transpose(B);`

because it does not involve allocating memory for the transposed of  $B$ .

As one more example of a powerful application of `DynamicArray` here is a program that defines a multidimensional `DynamicArray` of `Matrix` objects and pass it to a function.

Table 1  
Examples of typical instructions acting on `Matrix` objects.  $A, B, C, D$  are assumed to be declared as `Matrix`;  $r, c$  as `int`;  $a, b$  may be any kind of number

Example	C++ with MDP_Lib2.h
$A \in M_{r \times c}(\mathbb{C})$	<code>A.dimension(r,c)</code>
$A_{ij}$	<code>A(i,j)</code>
$A = B + C - D$	<code>A=B+C-D</code>
$A^{(ij)} = B^{(ik)} C^{(kj)}$	<code>A=B*C</code>
$A^{(ij)} = B^{(jk)} C^{(ik)}$	<code>A=mul_left(B,C)</code>
$A = aB + C$	<code>A=a*B+C</code>
$A = a\mathbf{1} + B - b\mathbf{1}$	<code>A=a+B-b</code>
$A = B^T C^{-1}$	<code>A=transpose(B)*inv(C)</code>
$A = B^\dagger \exp(iC)$	<code>A=hermitian(B)*exp(I*C)</code>
$A = \cos(B) + i \sin(B) * C$	<code>A=cos(B)+I*sin(B)*C</code>
$a = \text{real}(\text{tr}(B^{-1}C))$	<code>a=real(trace(inv(B)*C))</code>
$a = \det(B) \det(B^{-1})$	<code>a=det(B)*det(inv(B))</code>

```

// Program: example06.C
#include "MDP_Lib2.h"

DynamicArray<Matrix,3> initialize() {
    DynamicArray<Matrix,3> d(20,20,20);
    int i,j,k;
    for(i=0; i<20; i++)
        for(j=0; j<20; j++)
            for(k=0; k<20; k++) {
                d(i,j,k).dimension(2,2);
                d(i,j,k)(0,0)=k;
                d(i,j,k)(0,1)=i;
                d(i,j,k)(1,0)=j;
                d(i,j,k)(1,1)=k;
            };
    prepare(d);
    return(d);
}

DynamicArray<Matrix,3> f(DynamicArray<Matrix,3> c) {
    DynamicArray<Matrix,3> d(c.size(0),c.size(1),c.size(2));
    int i,j,k;
    for(i=0; i<c.size(0); i++)
        for(j=0; j<c.size(1); j++)
            for(k=0; k<c.size(2); k++)
                d(i,j,k)=sin(c(i,j,k));
    prepare(d);
    return(d);
}

int main() {
    DynamicArray<Matrix,3> a, b;
    a=initialize();
    b=f(a);

    int i=1, j=2, k=3;
    print(a(i,j,k));
    print(b(i,j,k));
    return 0;
}

```

a, b, c, d are 3D  $20 \times 20 \times 20$  arrays of  $2 \times 2$  matrices. The program prints

```

[[ 3.000+0.000i  1.000+0.000i ]
 [ 2.000+0.000i  3.000+0.000i ]]
[[ 0.022+0.000i  -0.691+0.000i ]
 [-1.383+0.000i  0.022+0.000i ]]

```

Note that the instruction

```
d(i,j,k)=sin(c(i,j,k));
```

computes the `sin()` of the Matrix `c(i,j,k)`.

## 2.5. classRandom\_generator

The random generator implemented in `Random_generator` is the Marsaglia random number generator described in Refs. [4,5] (the same generator is also used by the UKQCD collaboration in many large scale numerical simulations). It presents the nice feature that it can be initialized with a single long random number and its correlation time is relatively large compared with the standard random generator `rand()` of C++. To define a random generator, say `myrand`, and use the integer `seed` as seed one should simply pass `seed` to the constructor:

```
Random_generator myrand(seed);
```

For simple non-parallel applications one only needs one random generator. For this reason one `Random_generator` object called `Random` is automatically created and initialized with seed 0.

`Random_generator` contains some member variables for the seeds and four member functions:

- `plain()`; It returns a random `float` number in the interval  $[0, 1]$  with uniform distribution.
- `gaussian()`; It returns a random `float` number  $x$  generated with a Gaussian probability  $P(x) = \exp(-x^2/2)$ .
- `distribution(float (*P)(float, void*, void* a), void* a)`; It returns a random number  $x$  in the interval  $[0, 1]$  generated with a Gaussian probability  $P(x, a)$ , where  $a$  is any set of parameters pointed by `void *a` (the second argument passed to `distribution`). The distribution function  $P$  should be normalized so that  $0 \leq \min P(x) \leq 1$  and  $\max P(x) = 1$ .
- `Random_generator::SU(int n)`; it returns a random Matrix in the group  $SU(n)$  or  $U(1)$  if the argument is  $n = 1$ .

Here is a simple program that creates one random number generator, generates a set of 1000 random Gaussian numbers and counts how many of them are in the range  $[n/2, (n + 1)/2]$  for  $n = 0, \dots, 9$

```
// Program: example07.C
#include "MDP_Lib2.h"
int main() {
    Random_generator random;
    int i,n,bin[10];
    float x;
    for(n=0; n<10; n++) bin[n]=0;
    for(i=0; i<1000; i++) {
        x=random.gaussian();
        for(n=0; n<10; n++)
            if((x>=0.5*n) && (x<0.5*(n+1))) bin[n]++;
    }
    for(n=0; n<10; n++)
        printf("bin[%i] = %i\n", n, bin[n]);
    return 0;
}
```

Here is a program that computes the average of the sum of two sets of 1000 numbers,  $a$  and  $b$ , where  $a$  is generated with probability  $P(a) = \exp(-(a - \bar{a})^2/(2\sigma^2))$  and  $b$  with probability  $Q(b) = \sin(\pi b)$ .

```
// Program: example08.C
#include "MDP_Lib2.h"

float Q(float x, void *a) {
    return sin(Pi*x);
}

int main() {
    Random_generator random;
    int i,n,bin[10],N=100;
    float a,b,average=0, sigma=0.3, a_bar=1;
    for(i=0; i<N; i++) {
        a=(sigma*random.gaussian())+a_bar;
        b=random.distribution(Q);
        average+=a+b;
        printf("average=%f\n", average/(i+1));
    }
    return 0;
}
```

For large N the output asymptotically approaches  $a_{\text{bar}} + 0.5 = 1.5$ .

The algorithm for SU(2) is based on map between  $O(3)$  and SU(2), realized by

$$\{\hat{a}, \alpha\} \rightarrow \exp(i\alpha\hat{a} \cdot \sigma) = \cos(\alpha) + i\hat{a} \cdot \sigma \sin(\alpha), \quad (2)$$

where  $(\sigma^1, \sigma^2, \sigma^3)$  is a vector of Pauli matrices,  $\hat{a} \in S^2$  is a uniform vector on the sphere and  $\alpha \in [0, \pi]$  is a uniform rotation angle around that direction.

A random SU( $n$ ) is generated using the well-known Cabibbo–Marinari iteration [2] of the algorithm for SU(2).

## 2.6. classJackBoot

Suppose one has  $n$  sets of  $m$  different measured float quantities  $x[i][j]$  (where  $i = 0, \dots, n$  and  $j = 0, \dots, m$ ) and one wants to compute the average over  $i$ , of a function  $F(x[i])$ . For example,

```
float x[n][m]={...}
float F(float *x) {
    return x[1]/x[0];
}
float result=0;
for(i=0; i<n; i++)
    result+=F(x[i])/n;
}
```

Than one may ask: what is the error on the result? In general there are two algorithms to estimate this error: Jackknife and Bootstrap [3]. They are both implemented as member functions of the class JackBoot. They work for any arbitrary function  $F$ .

A JackBoot object, let's call it  $jb$ , is a sort of container for the data. After it has been filled it can be asked to return the mean of  $F()$  and its errors. Here is an example of how it works:

```
JackBoot jb(n,m);
```

```

jb.f=F;
for(i=0; i<n; i++)
    for(j=0; j<m; j++)
        jb(i,j)=x[i][j];
printf("Result          = %f\n", jb.mean());
printf("Jackknife error = %f\n", jb.j_err());
printf("Bootstrap error = %f\n", jb.b_err(100));

```

Note that

- The constructor of the class `JackBoot` takes two arguments: the first is the number of configuration; the second is the number of the quantities measured on each configuration.
- `jb.f` is the pointer to the function used in the analysis.
- `jb.mean()` returns the mean.
- `jb.j_err()` returns the Jackknife error.
- `jb.b_err()` returns the Bootstrap error. It takes as argument the number of Bootstrap samples. The default value is 200.

It is possible to declare arrays of `JackBoot` objects, but it is rarely necessary. It is simpler to declare different functions and repeat the analysis using the same `JackBoot` object assigning the pointer `JackBoot::f` to each of the functions at the time.

As another example consider the following program. It generates an array of 100 SU(6) matrices. For each matrix it computes trace and determinant, and returns the average of the ratio between the real part of the trace and the real part of the determinant (with its Jackknife and Bootstrap errors), and the average of the product of the real part of the trace and the real part of the determinant (with its Jackknife and Bootstrap errors)

```

// Program: example09.C
#include "MDP_Lib2.h"
#define n 100
float f1(float *x, float *a) { return x[0]/x[1]; }
float f2(float *x, float *a) { return x[0]*x[1]; }
int main() {
    Matrix A;
    JackBoot jb(n,2);
    int i;
    for(i=0; i<n; i++) {
        A=Random.SU(6);
        jb(i,0)=real(det(inv(A)));
        jb(i,1)=real(det(A));
    }
    jb.f=f1;
    printf("Result x[0]/x[1] = %f\n", jb.mean());
    printf("Jackknife error   = %f\n", jb.j_err());
    printf("Bootstrap error   = %f\n", jb.b_err(100));
    jb.f=f2;
    printf("Result x[0]*x[1] = %f\n", jb.mean());
    printf("Jackknife error   = %f\n", jb.j_err());
    printf("Bootstrap error   = %f\n", jb.b_err(100));
    return 0;
}

```

Note that any user defined function used by `JackBoot` (`f1` and `f2` in the example) has to take two arguments: an array of `float` (containing a set of measurements done on a single configuration) and a pointer to `void`. This is useful to pass some extra data to the function. The extra data must be passed to `JackBoot` by assigning the member variable

```
void* JackBoot::handle;
```

to it.

`JackBoot` has one more member function `plain(int i)`, where the call

```
/* define JackBoot jb(...) */
/* assign a value to the integer i */
jb.plain(i)
```

is completely equivalent to

```
/* define JackBoot jb(...)           */
/* assign a value to the integer i */
float f(float *x, void *a) {
    return x[i];
}
jb.f=f
```

Using `plain` saves one from defining trivial functions for `JackBoot`.

### 3. Parallel tools

With a parallel program we mean a job constituted by many programs running in parallel (eventually on different processors) to achieve a global goal. Each of the running programs, part of the same job, is called a process. Different processes can communicate to each other by exchanging messages (in MPI) or by accessing each-other memory (in computers with shared memory). The latter case is hardware dependent and will not be taken in consideration. With the term “partitioning” we will refer to the way different variables (in particular the field variables defined on a lattice) are distributed among the different processes.

The most common and portable parallel protocol is Message Passing Interface (MPI). It is implemented on a number of different platforms, for example Unix, Linux, Solaris, Windows NT and Cray T3D/E.

When writing a program using MPI, one essentially writes one program for each processor (Multiple Instructions Multiple Data) and MPI provides the functions for sending/receiving information among them.

We present here an example of how MPI works, even if a detailed knowledge of MPI is not required to understand the rest of this paper.

Suppose one wants to compute  $(5 * 7) + (4 * 8)$  in parallel using two processes. A typical MPI program to do it is the following:

```
// Program: example10.C
#include "stdio.h"
#include "mpi.h"
int main(int argc, char **argv) {
    int ME, Nproc;
    MPI_Status status;
    MPI_Init(&argc, &argv);
```

```

MPI_Comm_rank(MPI_COMM_WORLD, &ME);
MPI_Comm_size(MPI_COMM_WORLD, &Nproc);
if(ME==1) {
    int b;
    b=4*8;
    MPI_Send(&b, 1, MPI_INT, 0, 45, MPI_COMM_WORLD);
}
if(ME==0) {
    int a,b;
    a=5*7;
    MPI_Recv(&b, 1, MPI_INT, 1, 45, MPI_COMM_WORLD, &status);
    printf("%i\n", a+b);
}
MPI_Finalize();
};

```

ME is a variable that contains the unique identification number of each running process and Nproc is the total number of processes running the same code. Since different processes have different values of ME they execute different parts of the program (`if (ME==n) ...`).

In the example  $(5 * 7)$  is computed by process 0 while, at the same time  $(4 * 8)$  is computed by process 1. Then process 1 sends (`MPI_Send`) its partial result to process 0, which receives it (`MPI_Recv`) and prints out the sum.<sup>5</sup>

MPI instructions can be classified in

- initialization functions (like `MPI_Init`, `MPI_Comm_rank`, `MPI_Comm_size`),
- one to one communications (like `MPI_Send`, `MPI_Recv`),
- collective communication (where for example one variable can be broadcasted by one process to all the others).

It is not a purpose of this paper to explain how MPI works therefore we refer to [6].

### *3.1. mpi on top of MPI*

MPI is not Object Oriented and it is so general that MPI function calls usually require many arguments. This makes it very easy to do mistakes when programming with MPI.

For this reason the class `mpi_wormhole_class`, a wrapper to MPI, has been created. It has only one global representative object called `mpi`. We believe that `mpi` is easier to use than MPI in the context of lattice simulation.

`mpi` is not intended as a substitute or an extension to MPI, in fact only a subset of the features of MPI are implemented in `mpi` and occasionally one may need explicit MPI calls. Its main purpose is to build an intermediate programming level between MPI and generic\_lattice/generic\_field, which constitute the very essence of MDP 1.2. There are some advantages of having this intermediate level, for example:

- It allows one to compile the code without linking with MPI. In this case `mpi` substitutes some dummy functions to the MPI calls. This allows one to compile and run any parallel program on a single processor computer even if MPI is not installed, and it is also important for testing and debugging the code.
- In case one wants to run MDP 1.2 on a parallel machine that does not support MPI but does support a different protocol, one does not have to rewrite everything but only minor modifications to the class `mpi_wormhole_class` will be necessary.

---

<sup>5</sup> Needless to say that in such a simple program the parallelization is completely useless because the message passing takes more time than the computation of the whole expression!

### 3.2. mpi and wormholes

From now one we will use here the word “wormhole”, in its intuitive sense, to represent the virtual connections between the different processes. One can think of each process to have one wormhole connecting it with each of the other processes. Each wormhole has a number, the number of the process connected to the other end. Process X can send something to process Y just putting something into its wormhole n.Y. Process Y has to get that something from its wormhole n.X. `mpi` is the global object that allows the user to access these wormholes (put/get stuff to/from them). A schematic representation of a 3 processes job is shown in Fig. 1.

Using `mpi`, instead of MPI, program `example10.C` becomes:

```
// Program: example11.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    if(ME==1) {
        int b;
        b=4*8;
        mpi.put(b,0);
```

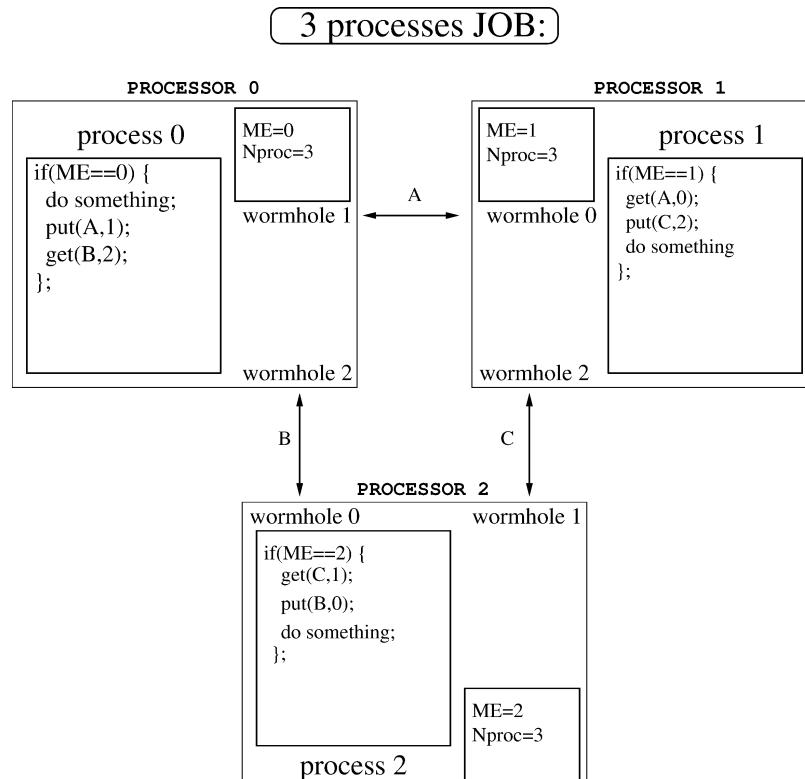


Fig. 1. Example of a 3 processes jobs. ME and Nproc are the global variables that characterize the different processes running (eventually) on different processors. The term wormhole refers to the virtual connection between each couple of processes.

```

} ;
if(ME==0) {
    int a,b;
    a=5*7;
    mpi.get(b,1);
    printf("%i\n" , a+b);
}
mpi.close_wormholes();
}

```

The most important things to notice are:

- (1) A constant (without any value assigned), PARALLEL, is defined on top. It tells the compiler to use the MPI functions. When it is omitted some dummy functions are substituted to the MPI calls and communications are skipped. This is useful for testing the behavior of a particular process in a multiprocess job without actually going parallel.
- (2) Nproc is the total number of processes. If it is not defined by the user, Nproc it is defined automatically (as a variable) when MDP\_MPI.h is included and its value is determined at runtime.
- (3) Two libraries MDP\_Lib2.h and MDP\_MPI.h are included. The standard headers (stdio.h, math.h, complex.h, iostream.h) are included by MDP\_Lib2.h.
- (4) ME is a global macro meaning mpi.my\_id, i.e. the identification number of the running process. Its value is assigned by the call to mpi.open\_wormholes when running with PARALLEL. Otherwise is has to be specified by the user after that call.
- (5) All of the initialization functions have been replaced by a single one:  
`void mpi_wormhole_class::open_wormholes(int&, char**);`  
It is compulsory even if the program does not run in parallel. It takes as input the same arguments of main().<sup>6</sup>
- (6) MPI\_Send and MPI\_Recv have been replaced by mpi.put and mpi.get respectively. They check automatically the size and type of their arguments.
- (7) At the end of the program one has to close\_wormholes. Actually if one forgets about this the destructor of mpi does take care of it.

The programmer is forced to use the names Nproc and ME for the total number of processes and the process identification number, respectively. They are keywords for MDP 1.2.

### 3.2.1. get() and put()

The functions put/get have the following syntax

```

template <class T> mpi_wormhole_class::put(T&,int);
template <class T> mpi_wormhole_class::put(T&,int,mpi_request&);
template <class T> mpi_wormhole_class::get(T&,int)

```

The first argument (passed by reference) is the object to be sent (put) or where to store the received one (get). The second argument is the destination (put) or the source (get). Forget about mpi\_request for the moment, it is optional.

All the variables related with message passing are members of mpi. In this way variables that are not explicitly necessary remain hidden inside it (for example the communicator and the communication tag).

---

<sup>6</sup>This is due to the fact that when running in parallel (using the runmpi command) a script submits the same program to the different processors with different command parameters, which are passed as arguments to main(). This is how a process knows its own identification number: ME.

The functions `put` and `get` that replaced `MPI_Send` and `MPI_Recv` are Object Oriented in the sense that they can be used to send any kind of object.<sup>7</sup> The size and the tag of the communication are automatically computed and do not have to be specified by the programmer. For example with the same syntax a `Complex` number can be put/get instead of an integer. The only price paid for this is that a process, X, cannot send more than one object to another process, Y, until the first object has been received. This is not such a price because in the kind of computations we are interested in, we want to minimize the number of put/get and we want to pack as much data as possible in each single communication, instead of making many small ones.

To avoid confusion, if the third argument of `put()` is not specified this function will not return until the send process has been completed (synchronous send).<sup>8</sup> Sometimes one wants the `put()` command to return even if the send is not completed, so that the process is free to do more computations, and eventually check at a later stage if the previous `put()` has completed its task (asynchronous send). One typical case is when the object to be send was dynamically allocated and it cannot be deallocated until the communication is concluded. The right way to implement this send is writing a code like the following:

```
/* mystruct is any built in variable
   or user defined structure */
mystruct *myobj=new mystruct;
mpi_request request;
put(*myobj,destination,request);
/* do something else */
mpi.wait(request);
delete myobj;
```

The `wait()` member function of `mpi` stops the process until the `put()` command has terminated its task. Its only argument is the object belonging to the class `mpi_request` that was passed to the corresponding `put()`. The function `wait()` can also take a pointer to an array of requests and wait for all of them. The general definition of this function is

```
void mpi_wormhole_class::wait(mpi_request&);
void mpi_wormhole_class::wait(mpi_request*, int);
```

the second argument is the length of the array.

It is also possible to put/get arrays of objects by using the overloaded functions

```
template <class T> mpi_wormhole_class::put(*T,long,int)
template <class T> mpi_wormhole_class::put(*T,long,int,mpi_request&)
template <class T> mpi_wormhole_class::get(*T,long,int)
```

where the first argument is a pointer to the first element of the array, the second is the length, the third is the destination/source.

### 3.2.2. Put/get matrices

We have said that with the put/get commands it is not possible to pass objects that contain a pointer to dynamically allocated memory and this is the case for a `Matrix` object. This does not mean that one cannot put/get a `Matrix`. The correct way to send a `Matrix` is

---

<sup>7</sup> Assuming it does not contain a pointer to dynamically allocated memory.

<sup>8</sup> Actually in `mpi.put` an synchronous send is internally implemented using `MPI_Isend + MPI_Wait`. This is because `MPI_Send` is ambiguous in some cases: the same program can have different behavior depending on the size of the objects that are passed. Our implementation avoids any ambiguity.

```
Matrix A;
/* do something with A */
mpi.put(A.address(),A.size(),destination);
```

The correct way to receive a `Matrix` is

```
Matrix B;
/* dimension B as A */
mpi.get(B.address(),B.size(),destination);
```

It is extremely important to stress that when one puts/gets a matrix one only sends/receives its elements. Therefore (in the last example) `B` must have the same dimensions of `A` at the moment `get` is called, otherwise one gets wrong results.

### 3.2.3. Global communications

As an example of global communication we will consider the following code

```
// Program: example12.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    int a;
    if(ME==1) {
        a=4*8;
        mpi.add(a);
    };
    if(ME==0) {
        a=5*7;
        mpi.add(a);
        printf("%i\n",a);
    };
    mpi.close_wormholes();
};
```

The member function:

```
void mpi_wormhole_class::add(float&);
```

sums the first argument passed by all the processes. In this way each process knows the result of the sum operation.

To sum an array of float the corresponding sum for arrays of float is implemented as:

```
void mpi_wormhole_class::add(float*,long);
```

(the second argument is the length of the array). The same function `add` also works for `double`, `long`, `int`, `Complex` and `Matrix` and arrays of these types. To add `Complex` numbers:

```
Complex x;
mpi.add(x);
```

To add Matrix objects

```
Matrix A;
mpi.add(A);
```

Some more member functions of `mpi` for collective communications are:

- `barrier()` which sets a barrier and all processes stop at that point until all the processes reach the same point.
- `abort()` which forces all the processes to abort.
- `template<class T> broadcast(T a, int p)` which broadcast the object `a` (belonging to an arbitrary class `T`) of process `p` to all other processes.
- `template<class T> broadcast(T *a, long n, int p)` which broadcast the array `a` of objects `T` of process `p` to all other processes. `n` here is the number of elements of the array.

Here is an example of how to use these collective communications

```
// Program: example13.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    int i,j;
    for(i=0; i<5; i++) {
        if(ME==0) j=i;
        else      j=0;
        if(i%2==0) mpi.barrier();
        mpi.broadcast(j,0);
        printf("I am process %i, i=%i, j=%i\n", ME, i,j);
    }
    mpi.close_wormholes();
}
```

This programs prints

```
I am process 0, i=0, j=0
I am process 0, i=1, j=1
I am process 0, i=2, j=2
I am process 0, i=3, j=3
I am process 0, i=4, j=4
I am process 1, i=0, j=0
I am process 1, i=1, j=1
I am process 1, i=2, j=2
I am process 1, i=3, j=3
I am process 1, i=4, j=4
```

## 4. Introducing lattices and fields

### 4.1. class generic\_lattice

By a lattice we mean a generic set of points and a set of functions to move from one point to another. In MDP 1.2 a lattice is implemented as any subset of a regular n-dimensional grid with a given (arbitrary) topology.

A lattice of such a kind is implemented as an object belonging to the class `generic_lattice` using the command:

```
generic_lattice mylattice(ndim,mybox,mypartitioning,
                         mytopology,seed,next_next);
```

or the command

```
generic_lattice mylattice(ndim,ndir,mybox,mypartitioning,
                         mytopology,seed,next_next);
```

where:

- `mylattice` is a user defined variable that will contain the actual lattice structure.
- `ndim` is the dimension of the basic regular grid.
- `ndir` is the number of directions. It can be omitted and, by default, it is assumed to be equal to `ndim`.
- `mybox` is a user defined `ndim`-ensional array of `int` that contains the size of the basic regular grid.
- `mypartitioning` is a user defined function that for each site of the basic grid returns the number of the process that stores it (or it returns `NOWHERE` if the point has to be excluded from the lattice). If `mypartitioning` is omitted all sites of the original grid (specified by `mybox`) will be part of the final lattice, and they will be equally distributed between the processes according with the value of coordinate 0 of each point.<sup>9</sup>
- `mytopology` is a user defined function that for each site of the final (remaining) grid, and for each dimension in the `mydim`-ensional space, returns the coordinates of the neighbor sites in the up and down direction. If `mytopology` is omitted the lattice will, by default, have the topology of an `ndim`-ensional torus, i.e. the same of the basic grid plus periodic boundary conditions.
- `seed` is an integer that is used to construct a site dependent seed to initialize the random generators associated to each lattice sites. If omitted it is assumed to be 0.
- `next_next` is an integer that may have only three values (1, 2 or 3). It essentially fixes the thickness of the boundary between different processes. If omitted the default value is 1. Its use will not be discussed further.

For the moment we will restrict ourselves to the study of lattices with the basic (torus) topology, postponing the rules concerning the specifications for `mypartitioning` and `mytopology`.

Here is a short program to create a 2D  $8^2$  lattice (but no field yet associated to it) running in parallel on 4 processes.

```
// Program: example14.C
#define PARALLEL
#define Nproc 4
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);
```

---

<sup>9</sup> For example, consider a lattice of size 8 in the 0 direction distributed on 4 processes. Process 0 will contain sites coordinate 0 equal to 0 and 1. Process 1 will contain sites coordinate 0 equal to 2 and 3. An so on.

```

int mybox[ ]={8,8};
generic_lattice mylattice(2,mybox);
mpi.close_wormholes();
};

```

When it runs (since `Nproc` is set to 4) the lattice will be automatically subdivided into 4 sublattices of size {2, 8}. `mylattice` contains information about the sites and how they are distributed among the different processes. Moreover the different processes automatically communicate to each other this information so that each of them knows which local sites will have to be passed to the neighbor processes. All the communications to establish the topology are done only once and the information are inherited by all the fields defined on `mylattice`. The program produces the following output

```

PROCESS 0 STARTING
PROCESS 2 STARTING
PROCESS 1 STARTING
PROCESS 3 STARTING
=====
Starting [ Matrix Distributed Processing ] ...
This program is using the packages: MDP_Lib2 and MDP_MPI
Created by Massimo Di Pierro (mdp@FNAL.GOV) version 17-11-1999
=====
Going parallel ... YES

Initializing a generic_lattice...
Communicating...
Initializing random per site...
Done. Let's begin to work!
PROCESS 1 ENDING AFTER 0.010 sec.
PROCESS 2 ENDING AFTER 0.008 sec.
PROCESS 3 ENDING AFTER 0.011 sec.
=====
Fractional time spent in communications by processor 0 is 0.03
Fractional time spent in communications by processor 1 is 0.04
Fractional time spent in communications by processor 2 is 0.02
Fractional time spent in communications by processor 3 is 0.03
Ending this program.
Any runtime error below this point means mess with mpi and/or
deallocateon of memory.
=====
PROCESS 0 ENDING AFTER 0.012 sec.

```

Note that the output automatically includes the execution time (using the wall clock) and the fractional time spent in communication by each processor (0.03 means 3%). The latter is computed by measuring the total time spent inside the communication function `update()`.

Two other member functions of `generic_lattice` are

```

long generic_lattice::global_volume();
long generic_lattice::local_volume();

```

The first return the size of the total lattice volume (i.e. the total number of sites), and the second returns the size of the portion of volume stored by the calling process.

The member function

```
long generic_lattice::size();
```

returns the total volume of the box containing the lattice and

```
long generic_lattice::size(int mu);
```

returns the size, in direction  $\mu$ , of the box containing the lattice.

#### 4.2. class site

To access a site one can define a `site` variable using the syntax

```
site x(mylattice);
```

This tells the compiler that `x` is a variable that will contain the coordinate of a site and a reference to the lattice `mylattice` on which the point is defined.

Let's look at a modification of the preceding code that create a lattice and, for each site `x`, prints out the identification number of the process that stores it:

```
// Program: example15.C
#define PARALLEL
#define Nproc 5
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    int mybox[]={10,10};
    generic_lattice mylattice(2,mybox);
    site x(mylattice);
    forallsites(x)
        printf("Site=(%i,%i) is stored by %i\n", x(0), x(1), ME);
    mpi.close_wormholes();
}
```

Note that

```
x(mu)
```

returns the coordinate `mu` of `x`.

This simple program allows one to print a map of the lattice. The output of this code depends on the number of processors on which it is running.<sup>10</sup> The loop command

```
forallsites(x) /* do something with x */
```

spans first all the even sites and then all the odd sites.

---

<sup>10</sup> The buffers of the different processors are copied to the standard output at a random time partially scrambling the output of the different processes. For this reason it is in general a good rule to print only from process `ME==0`.

It is possible to set the value of a `site` variable `x` to a particular site of given coordinates ((3, 7), for example) using the command

```
x.set(3,7);
```

but one must be careful and execute this statement only on the process which contains the site (3, 7) or, eventually, in a process that contains a neighbor of this site. In this case, in fact, the process will also contain a copy of the original site therefore it will be able to access the location. By default each process will store its own sites plus a copy of next-neighbor sites in each direction and a copy of next-next neighbor sites (moving in two different directions).<sup>11</sup> They will be referred to as boundary sites. Boundary sites are different for different processes.

It is possible to check if a site is a local site in the following way:

```
x.set(3,7)
if(x.is_in()) /* then do something */
```

The member function `is_in()` returns TRUE if the site is local. A better way to code it would be

```
if(on_which_process(3,7)==ME) {
    x.set(3,7);
    /* do something */
}
```

Other member functions of `site` are

- `is_in_boundary()` returns TRUE if a site is in the boundary of that process.
- `is_here()` returns TRUE if the site `is_in()` or `is_in_boundary()`.
- `is_equal(int,int,int...)` returns TRUE if the site is equal to the site specified by the coordinates that are arguments of `is_equal()` ( $x_0, x_1, x_2, \dots$ ). **Accessing a site that `!is_here()` crashes the program.**

To move from one site to another is quite simple:

```
x=x+mu; // moves x up in direction mu (integer) of one step
x=x-mu; // moves x down in direction mu (integer) of one step
```

Note that `mu` is an integer and in general

$$x + mu \neq x \neq x - mu \quad (3)$$

even when `mu` is zero.

Here is a test program to explore the topology:

```
// Program: example16.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);
    int mybox[] = {10, 10};
```

---

<sup>11</sup> This is required, for example, by the clover term in QCD.

```

generic_lattice mylattice(2,mybox);
site x(mylattice);
int mu=0;
if(ME==0) {
    x.set(0,0);
    do {
        printf("x=(%i,%i)\n", x(0), x(1));
        if(x.is_in_boundary()) error("I found the boundary");
        x=x+mu;
    } while(TRUE);
}
mpi.close_wormholes();
};

```

This program stops after 5 iterations if `Nproc==2`, after 4 if `Nproc==3`, after 3 if `Nproc==4`, after 2 if `Nproc==5` and so on. It will never stop on a single process job (`Nproc==1`) because of periodic boundary conditions. Since all sites with the same `x(0)`, by default, are stored in the same process, if one moves in direction `mu=1` the job will never stop despite the number of processes.

#### 4.3. class generic\_field

A `generic_field` is the simplest field one can define on a given `generic_lattice`.

Suppose, for example, one wants to associate a given structure, `mystruct`, to each site of `mylattice` (in the example a structure that contains just a `float`) and initialize the field variable to zero:

```

// Program: example17.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
struct mystruct {
    float value; /* or any other structure */
};
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    int mybox[]={10,10};
    generic_lattice      mylattice(2,mybox);
    generic_field<mystruct> myfield(mylattice);
    site x(mylattice);
    forallsites(x)
        myfield(x).value=0;
    myfield.update();
    mpi.close_wormholes();
};

```

The command

```
generic_field<mystruct> myfield(mylattice);
```

defines `myfield`, a field of `mystruct` on the sites of `mylattice`. The command `myfield(x)` returns by reference the structure associated to the site `x`.

The function

```
void generic_lattice::update(int,int,int)
```

is the most important of all. For the moment we will restrict to that case when it is called with no arguments. Its task is to perform all the communications necessary in order for each process to get an updated copy of the boundary sites stored by a neighbor process.

After the loop all sites that are local (`forallsites`) are initialized, but sites that are in boundary (since they are copies of sites initialized on a different process) still contain “random” numbers. One way to initialize also the sites in the boundary and avoid some time consuming communications is replacing the lines

```
forallsites(x)
  myfield(x).value=0;
mylattice.update();
```

with

```
forallsitesandcopies(x) myfield(x).value=0;
```

(Note that the command `forallsitesandcopies` only works for a local expression that does not call the local random generator.)

It is also possible to loop on all the sites of a given parity, for example, EVEN (or ODD), in a slow way

```
int parity=EVEN;
forallsites(x)
  if(x.parity()==parity)
    /* then do something */
```

or in a fast way

```
int parity=EVEN;
forallsitesofparity(x,parity)
  /* do something */
```

The second expression is faster because sites with the same parity are stored contiguously in memory.

#### 4.3.1. Local random generator

Once a `generic_lattice` is declared one random generator per site is automatically created and initialized with a function of the site coordinates. The random generator (and its member functions) of site `x` of `mylattice` can be accessed (by reference) with the command

```
Random_generator& generic_lattice::random(site)
```

Program `example17.C` can be modified so that `myfield` is initialized, for example, with a gaussian random number:

```
// Program: example18.C
#define PARALLEL
#define Nproc 2
#include "MDP_Lib2.h"
```

```
#include "MDP_MPI.h"
struct mystruct {
    float value; /* or any other structure */
};

int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);
    int mybox[] = {10, 10};
    generic_lattice      mylattice(2, mybox);
    generic_field<mystruct> myfield(mylattice);
    site x(mylattice);
    forallsites(x)
        myfield(x).value = mylattice.random(x).gaussian();
    myfield.update();
    mpi.close_wormholes();
}
```

#### 4.3.2. Accessing a lattice from a field

It is always possible to access a `generic_lattice` from its field using `lattice()`, a member function of `generic_field` that returns by reference the `generic_lattice` on which the field is defined.

For example, in the preceding program one could substitute

```
myfield(x).value = mylattice.random(x).gaussian();
```

with

```
myfield(x).value = myfield.lattice().random(x).gaussian();
```

and get the same result. This is useful because one can avoid passing a `generic_lattice` object to a function of a `generic_field`.

#### 4.3.3. More on update()

In some cases it may be useful to have a field that contains `n` `mystruct` variables at each site.

A field of this type can be defined with the command

```
generic_field<mystruct> myfield(mylattice, n);
```

Consider, for example, the simple case of a tensor field  $T_{\mu}^{ij}(x)$ , where  $i, j \in \{0, \dots, 5\}$ ,  $\mu \in \{0, \dots, 3\}$ . It can be defined with

```
generic_field<Complex[4][5][5]> T(mylattice);
```

and its elements can be accessed with the natural expression:

```
T(x)[mu][i][j]
```

where `mu`, `i`, `j` are integers.

Alternatively the tensor  $T_{\mu}^{ij}$  can be defined with the command

```
generic_field<Complex[5][5]> T(mylattice, 4);
```

and its elements can be accessed with the expression:

```
T(x,mu)[i][j]
```

Sometimes the second choice is better for the following reason: In many algorithms one often needs to update all the boundary site variables having a given a parity and a given mu. One wants to do the update in one single communications. The `update()` function allows one to do exactly this:

```
int parity=EVEN, int mu=3;
T.update(parity,mu);
```

#### 4.3.4. operator()

Some care is required when declaring a `generic_field`. The structure associated to the sites cannot contain a pointer to dynamically allocated memory, therefore it cannot contain a `Matrix` (or a `DynamicArray`) object. This is not a limitation since the structure associated to the sites can be a multidimensional array and one can always map it into a `Matrix` by creating a new field, inheriting `generic_field` and redesigning `operator()`.

In this subsection we explain how to use `Matrix` to build and interface to the field variables.

Let's go back to the example of a tensor field  $T_\mu^{ij}(x)$ . One can define it as

```
class mytensor: public generic_lattice<Complex[5][5]> {};
```

```
mytensor T(mylattice,4);
```

Instead of accessing its elements as `Complex` numbers one may want to access the two-dimensional array as a `Matrix` object. This is done by overloading the `operator()` of the class `generic_field`:

```
Matrix myfield::operator() (site x, int mu) {
    return Matrix(address(x,mu),5,5);
};
```

where `address()` is a member function of `generic_lattice` that returns the location of memory where the field variables at  $(x, \mu)$  are stored. After this definitions one can simply access the field using the expression:

```
T(x,mu)
```

that now returns a `Matrix`.

Now `mytensor T` looks like a field of  $5 \times 5$  matrices and its elements can be used as discussed in Section 2. For example, one can initialize the tensor using random SU(5) matrices generated independently by the local random generators associated to each site

```
forallsites(x)
    for(mu=0; mu<4; mu++)
        T(x,mu)=T.lattice().random(x).SU(5);
```

or print a particular element

```
x.set(3,5);
if(x.is_in()) print(T(x,2));
```

#### 4.3.5. A few derived fields

Following the directives of the last subsections four more basic fields are implemented in MDP 1.2.

- `Matrix_field`: a field of `Matrix`;
- `NMatrix_field`: a field of  $n$  `Matrix`;
- `Vector_field`: a field of vectors (a vector is seen as a `Matrix` with one single column);

- NVector\_field: a field of  $n$  vectors;

To explain their usage we present here a few lines of code that define a field of  $3 \times 4$  matrices, A, a field of 4-vectors, u, and a field of 3-vectors, v; then for each site compute

$$v(x) = A(x)u(x). \quad (4)$$

This can be implemented as

```
Matrix_field A(mylattice, 3, 4);
Vector_field u(mylattice, 4);
Vector_field v(mylattice, 3);
/* assign values to the fields */
forallsites(x) v(x)=A(x)*u(x);
```

and its generalization to

$$v_i(x) = A_i(x)u_i(x) \quad (5)$$

for i in the range  $[0, N - 1]$

```
int N=10; // user defined variable
NMatrix_field A(mylattice,N,3,4);
NVector_field u(mylattice,N,4);
NVector_field v(mylattice,N,3);
/* assign values to the fields */
forallsites(x) for(i=0; i<N; i++) v(x,i)=A(x,i)*u(x,i);
```

#### 4.4. Input/output

The class generic\_field contains two I/O member functions:

```
void generic_lattice::save(char[], int, int);
void generic_lattice::load(char[], int, int);
```

They take three arguments

- The filename of the file to/from which to save/load the data
- The identification number of the master process that will physically perform the I/O
- The size (in sites) of the buffer used by the master process to communicate with the other processes.

The last two arguments are optional and by default the master process is process 0 and the buffer size is 1024 sites. The identification number of the master process has to correspond to the process running on the processor that is physically connected with the I/O device where the file filename is. The size of the buffer will not affect the result of the I/O operation but may affect the speed (which should increase with the size of the buffer). If the size is too big there may be an “out of memory” problem (that usually results in obscure error messages when compiling with MPI). Note that the I/O functions save/load have to be called by all processes (not just the master one). In fact all processes which contain sites of the lattice are involved in the I/O by exchanging informations with the master.

As an example of I/O we consider the case of a save( ) operation. Each of the processes arranges the local field variables to be saved into packets and sends the packets to the process that performs the I/O (we assume it is process 0). Process i arranges its site into packets ordered according with the global parameterization and sends one packet at the time to process 0. In this way process 0 only receives one packet at the time for each of the processes and already finds the sites stored in the correct order so that it does not need to perform a seek. Process

0 saves the sites according to the global parameterization (which is independent from the lattice partitioning over the parallel processes). Its only task is to sweep all the sites and, for each of them, pick the corresponding field variables from the packet sent by the process that stored it.

We believe this is the most efficient way to implement a parallel I/O for field variables. As a particular case, assuming process 0 has enough memory, one could set the size of the buffer equal to the maximum number of sites stored by a process. In this case the parallel I/O would be done performing only one communication per process.

As an example of efficiency: a field as large as 100 MB can be read and distributed over 8 500 MHz PentiumIII PCs in parallel (connected with Ethernet) in a few of seconds.

The save/load function are inherited by every field based on `generic_field`. For example:

```
/* define mylattice */
class myfield: public generic_field<Complex[7][3]> {};
myfield F(mylattice);
/* do something */
F.save("test.dat");
F.load("test.dat");
```

If a field contains member variables other than the structure at the sites, these member variables are ignored by save/load. The only variables that are saved together with the field are:

- the number of dimensions,
- the size of the box containing the lattice,
- the total number of sites belonging to the lattice,
- the size in bytes of the structure at each lattice site,
- a code to detect the endianess of the computer that saved the data,
- the time and date when the file was created.

We also provide a small program, `inspect.C`, that can open a file and extract this information.

It is always possible to add any other information to the bottom of the file (in binary or ASCII format) without compromising its integrity.

#### 4.5. The internal representation of a lattice

A lattice is a collection of sites and, for sake of simplicity, from now on we will refer to the lattice defined by

```
int ndim=2;
int mybox[]={6,6};
int mypartitioning(int x[], int ndim, int nx[]) {
    if(x[0]<3) return 0;
    if(x[1]<4) return 1;
    return 2;
}
generic_lattice mylattice(ndim,mybox,mypartitioning);
```

This lattice is represented in Fig. 2 and partitioned as in Fig. 5.

To speed up the communication process MDP 1.2 uses different parametrizations for labelling the sites:

- the global coordinates (as shown in Fig. 2),
- a global index (as shown in Fig. 3),
- a local index (as shown in Fig. 8 from the point of view of process 0),

and it provides functions to convert one parameterization into another.

The global parameterization is quite natural. For example: the point  $x[] = \{3, 2\}$  has a global index given by  $x[1]*mybox[0]+x[0]$ ; The two global parametrizations have a one to one correspondence. The local

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)
○	○	○	○	○	○
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)
○	○	○	○	○	○
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)
○	○	○	○	○	○
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)
○	○	○	○	○	○
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)
○	○	○	○	○	○
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)
○	○	○	○	○	○

Fig. 2. Example of a  $6 \times 6$  grid. The points are labelled by their coordinates.

0	1	2	3	4	5
○	○	○	○	○	○
6	7	8	9	10	11
○	○	○	○	○	○
12	13	14	15	16	17
○	○	○	○	○	○
18	19	20	21	22	23
○	○	○	○	○	○
24	25	26	27	28	29
○	○	○	○	○	○
30	31	32	33	34	35
○	○	○	○	○	○

Fig. 3. Example of a  $6 \times 6$  grid. The points are labelled by their global index.

parameterization is more complicated and it represents the order in which each process stores the sites in the memory.

When the lattice is defined the constructor of the class `generic_lattice` spans all the sites using the global coordinates (Fig. 2) and it assigns them the global index (Fig. 3) and a parity (Fig. 4). Then each process checks which sites are local sites (Fig. 5). After the topology is assigned to the lattice (Fig. 6), each process identifies which sites are neighbors of the local sites (Fig. 7 for process 0). At this point each process has all the information needed to build a local parameterization for the sites. All even sites belonging to process 0 are labeled with a progressive number; then all the odd sites from process 0 are labelled in progression. The same procedure is repeated by each

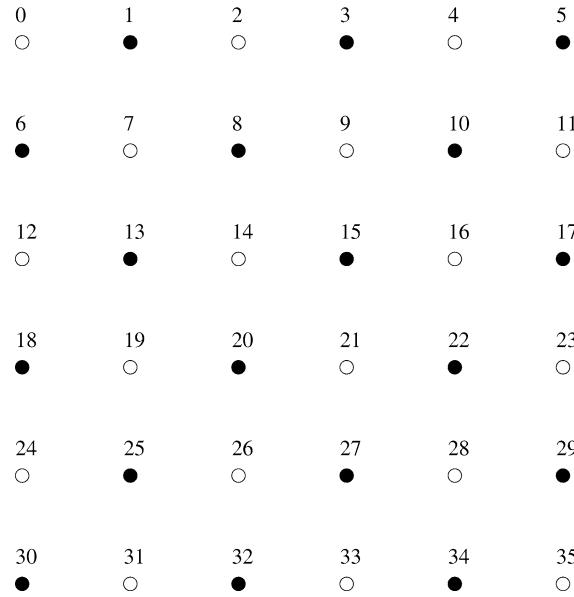


Fig. 4. Example of a  $6 \times 6$  grid. The points are labelled by their global index. Points with even (white) and odd (black) parity are distinguished.

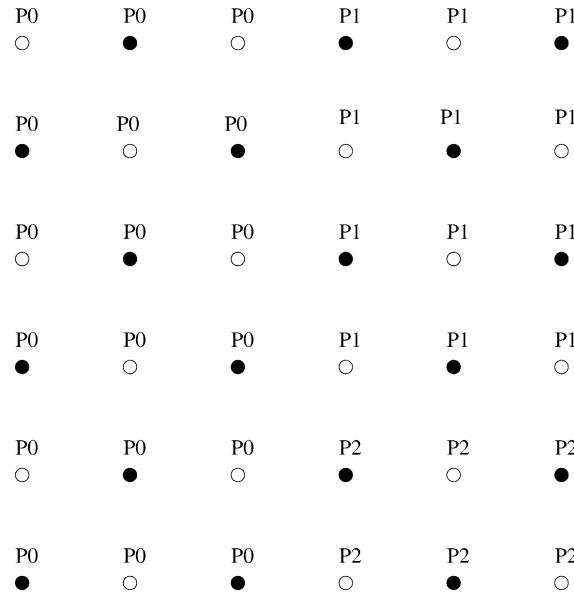


Fig. 5. Example of a  $6 \times 6$  grid distributed on 3 processors (according with some user defined partitioning). Each point is labelled by the identification number of the process that stores it.

process for each process (including itself). The result (for the case in the example) is shown in Fig. 8 (from the point of view of process 0).

When a field is associated to the lattice the field variables are stored by each process according with the local parameterization. This guarantees that:

- Boundary sites that have to be copied from the same process are stored continuously in the memory. This minimizes the need for communications.

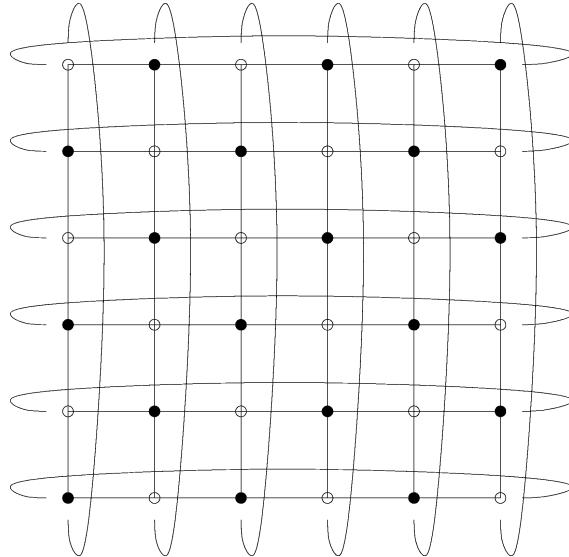


Fig. 6. Example of a  $6 \times 6$  grid endowed with the default (torus) topology. When the topology is assigned to the grid, it is promoted to the rank of lattice.

local ○	local ●	local ○	from P1 ●	○	from P1 ●
local ●	local ○	local ●	from P1 ○	●	from P1 ○
local ○	local ●	local ○	from P1 ●	○	from P1 ●
local ●	local ○	local ●	from P1 ○	●	from P1 ○
local ○	local ●	local ○	from P2 ●	○	from P2 ●
local ●	local ○	local ●	from P2 ○	●	from P2 ○

Fig. 7. Example of a  $6 \times 6$  lattice from the point of view of process 0. Process 0 knows which sites are local, which non-local sites are neighbors of the local sites and from which process to copy them.

- Boundary sites copied from the same process (and the local sites as well) of given parity are also stored continuously in memory. This speeds up loops on sites of given parity.
- When a process X copies a neighbor site from process Y it does not need to allocate a buffer because the data can be received directly in the memory location where it is normally stored.

Given a site variable `x` on `mylattice` one can ask for the mu coordinate using the syntax

`x(mu)`

or the global index

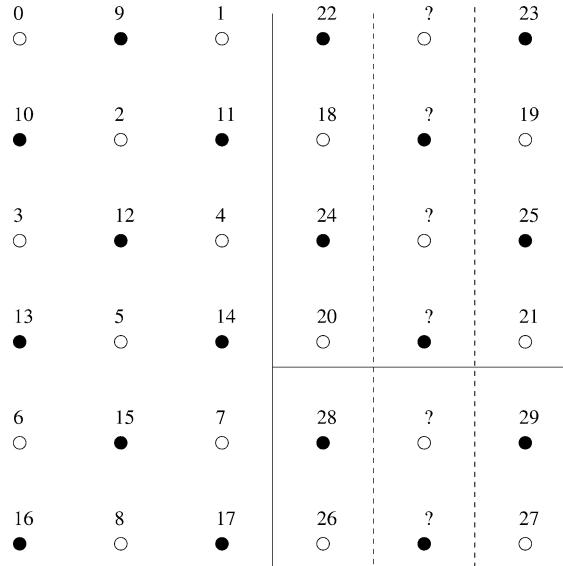


Fig. 8. Example of a  $6 \times 6$  lattice form the point of view of process 0. Process zero assigns a local parameterization to the local sites and to the neighbor sites of the local points. It cannot access the rest of the sites.

`x.global_index()`

or the local index

`x.local_index()`

Moreover one can initialize a site variable specifying the coordinates

`x.set(3, 2);`

the global index

`x.set_global(26);`

or the local index

`x.set_local(7); /* assuming ME is 0 */`

If `x` and `y` are two `site` variables in the same code defined on the same lattice, but using different partitioning. One can assign `y` to `x` using the assignment operator

`x=y;`

This is equivalent to

`x.set_global(y.global_index());`

#### 4.6. Partitioning and topology

One of the most important characteristics of MDP 1.2 is that one is not limited to a box-like lattice (even if it has to be a subset of the points in a box) and the sites of the lattice can be associated to an arbitrary topology. Before stating with weird staff here is the definition of the default function for the lattice partitioning:

```
template<int dim>
int default_partitioning(int *x, int ndim, int *nx) {
    float tpp1=nx[dim]/NPROC;
    int tpp2=(int) tpp1;
    if(tpp1-tpp2>0) tpp2+=1;
    return x[dim]/tpp2;
};
```

Any partitioning function takes three arguments:

- An array  $x[ndim]$  containing the  $ndim$  coordinates of a site.
- The dimensions,  $ndim$ , of the box containing the lattice sites.
- An array,  $nx[ndim]$ , containing the size of box in each dimension.

It returns the identification number of the process that stores the site corresponding to the coordinates in  $x[ ]$ .<sup>12</sup>

One can define a different partitioning just by defining another function that takes the same arguments as the default one and pass it as third argument to the constructor of `generic_lattice`. If the new partitioning function, for a particular input site, returns a number outside the range 0 to  $NPROC-1$ , that site is excluded from the lattice. To this scope the macro `NOWHERE` is defined. If a site has to be excluded from the lattice the partition function should return `NOWHERE`. If one defines a partitioning function that excludes some points from the lattice one is forced to create a user defined topology, otherwise one ends up with sites which are neighbors of sites that are `NOWHERE` and the program fails.

The default torus topology is defined as

```
void torus_topology(int mu, int *x_dw, int *x, int *x_up,
                    int ndim, int *nx) {
    for(int nu=0; nu<ndim; nu++) if(nu==mu) {
        x_dw[mu]=(x[mu]-1+nx[mu]) % nx[mu];
        x_up[mu]=(x[mu]+1) % nx[mu];
    } else x_up[nu]=x_dw[nu]=x[nu];
};
```

Any topology function takes six arguments, they are:

- A direction  $mu$  in the range 0 to  $ndim-1$ .
- A first array of coordinates,  $x_{dw}[ ]$ .
- A second array of coordinates,  $x[ ]$ .
- A third array of coordinates,  $x_{up}[ ]$ .
- The dimensions,  $ndim$ , of the box containing the lattice sites.
- An array,  $nx[ndim]$ , containing the size of box in each dimension.

The coordinates in  $x[ ]$  are taken as input and the topology function fills the arrays  $x_{dw}[ ]$  and  $x_{up}[ ]$  with the coordinates of the neighbor points when moving from  $x[ ]$  one step in direction  $mu$  down and up, respectively.

The topology function will only be called for those sites  $x[ ]$  that belong to the lattice. It has to fill  $x_{dw}[ ]$  and  $x_{up}[ ]$  with the coordinates of points that have not been excluded from the lattice.

---

<sup>12</sup> Note that at this level `site` variables are not used. In fact the partitioning function (as well the topology function) is called before the lattice is defined, and it is used to define it. A site variable can be defined only after a lattice is defined.

#### 4.6.1. Nonperiodic lattice

As an example one can create a lattice with a true physical boundary by saying, for example, that moving up (or down) in a particular direction from a particular subset of sites one remains where one is.<sup>13</sup> In the simple case of a finite box one could define the topology:

```
void box_topology(int mu, int *x_dw, int *x, int *x_up,
                  int ndim, int *nx) {
    torus_topology(mu,x_dw,x,x_up,ndim,nx);
    if(x[mu]==0) x_dw[mu]=x[mu];
    if(x[mu]==nx[mu]-1)) x_up[mu]=x[mu];
}
```

#### 4.6.2. Hexagonal lattice

Another example of a lattice with a weird topology is the hexagonal grid shown in Fig. 9. The trick to do it is having a 3D lattice flat in the third direction, put NOWHERE the centers of the hexagons and define the proper topology for the remaining points. It can be done by the following code

```
int exagonal_partitioning(int x[], int ndim, int nx[]) {
    if((x[0]+x[1])%3==0) return NOWHERE;
    else return default_partitioning<0>(x,ndim,nx);
```

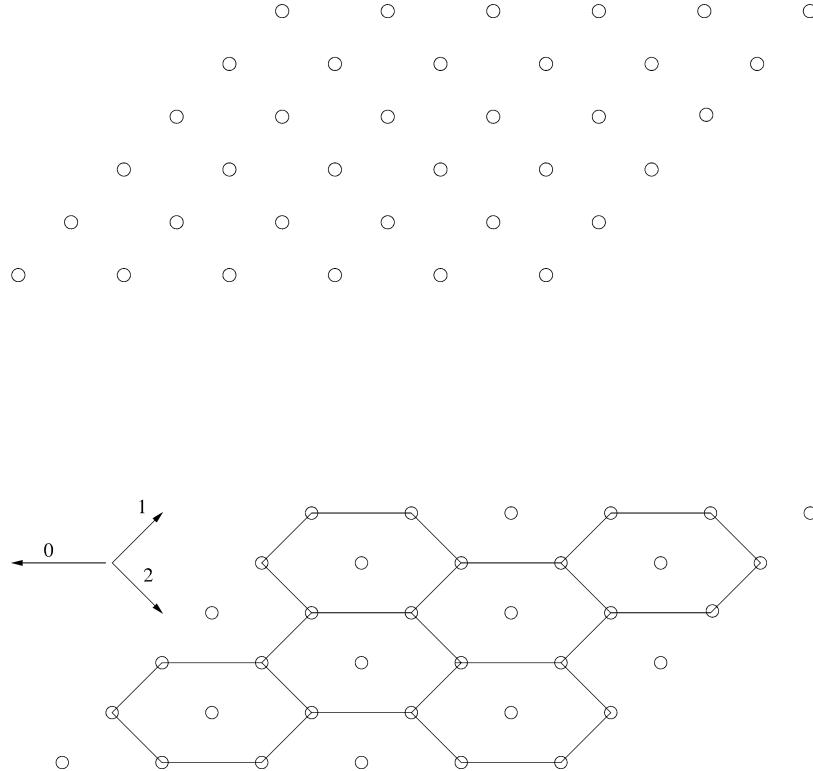


Fig. 9. Example of a  $6 \times 6$  grid to be used to build an hexagonal lattice (top) and the metric associated to this grid (bottom). The points at center of the hexagons are excluded from the lattice by the partitioning function.

<sup>13</sup> This kind of lattice can be used, for example, to study a 3D solid under stress and study its deformations.

```

};

void exagonal_topology(int mu, int *x_dw, int *x, int *x_up,
                      int ndim, int *nx) {
    if((x[0]+x[1])%3==1) {
        switch(mu) {
            case 0: x_up[0]=(x[0]+1)%nx[0]; x_up[1]=x[1]; break;
            case 1: x_up[1]=(x[1]+1)%nx[1]; x_up[0]=x[0]; break;
            case 3: x_up[0]=(x[0]-1+nx[0])%nx[0];
                      x_up[1]=(x[1]-1+nx[1])%nx[1]; break;
        };
        x_up[2]=x[2]; x_dw[0]=x[0];
        x_dw[1]=x[1]; x_dw[2]=x[2];
    };
    if((x[0]+x[1])%3==2) {
        switch(mu) {
            case 0: x_dw[0]=(x[0]-1+nx[0])%nx[0]; x_dw[1]=x[1]; break;
            case 1: x_dw[1]=(x[1]-1+nx[0])%nx[1]; x_dw[0]=x[0]; break;
            case 3: x_dw[0]=(x[0]+1)%nx[0]; x_dw[1]=(x[1]+1)%nx[1]; break;
        };
        x_dw[2]=x[2]; x_up[0]=x[0];
        x_up[1]=x[1]; x_up[2]=x[2];
    };
}
};

int n=2;
int mybox[ ]={3*n, 3*n, 1};

generic_lattice exagonal(3,mybox,exagonal_partitioning,
                         exagonal_topology);

```

One can observe that we need a flat third dimension to allow for three possible direction. For this reason MDP 1.2 allows to define a lattice with a number of directions different from the number of dimensions. To avoid entering in a number of technical details we will avoid discussing this possibility here. What one can do with such a lattice is another matter and will not be discussed. From now on we will concentrate on more regular lattices which tend to have a more intuitive interpretation.

#### 4.6.3. Antiperiodic boundary conditions

There are two possible meaning for antiperiodic boundary conditions: (1) antiperiodic boundary conditions for the lattice (i.e. a Moebius topology). (2) antiperiodic boundary conditions for a field defined on the lattice.

The first case can be handled in the same way as any other topology. The second case has nothing to do with the lattice topology, but is a property of the field and one has to take care of it in the definition of `operator()` for the field in question.

#### 4.7. Memory optimization

Actually the class `generic_field` has more properties of those described. For example each field can be deallocated at any time if memory is needed:

```

struct mystruct { /* any structure */ };
generic_lattice mylattice1(mydim1,mybox1);
generic_field<mystruct> myfield(mylattice1);
/* use myfield */
myfield.deallocate_memory();
generic_lattice mylattice2(mydim2,mybox2);
myfield.allocate_field(mylattice2);

```

Note that one cannot reallocate a field using a different structure at the site. This would be very confusing.

## 5. Examples of parallel applications

We will present here a few example of parallel application that use MDP 1.2. To keep things simple we will not make use of fields of matrices.

### 5.1. Electrostatic potential for a given distribution of charges

We consider here the vacuum space inside a cubic metal box connected to ground and containing a given (static) distribution of charge. We want to determine the electrostatic potential in the box.

The vacuum is implemented as a finite  $20^3$  lattice. Two fields, a charge density  $q(x)$  and a potential  $u(x)$ , are defined on it. In the continuum the potential is obtained by solving the Gauss law equation

$$\nabla^2 u(x) = q(x). \quad (6)$$

Its discretized form reads

$$\sum_{i=0}^2 [u(x + \hat{i}) - 2u(x) + u(x - \hat{i})] = q(x) \quad (7)$$

which can be solved in  $u(x)$

$$u(x) = \frac{1}{6} \left[ q(x) + \sum_{i=0}^2 u(x + \hat{i}) + u(x - \hat{i}) \right]. \quad (8)$$

Therefore the static potential solution is obtained by iterating Eq. (8) on the vacuum until convergence.

As initial condition we assume that only two charges are present in the box:

$$q(x) = 3\delta(x - A) - 5\delta(x - B), \quad (9)$$

where  $A = (3, 3, 3)$  and  $B = (17, 17, 17)$ . Moreover since the box has a finite extension the `box_topology` will be used.

Here is the parallel program based on MDP 1.2.

```

// Program: application1.C
#define PARALLEL
#define Nproc 4
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);
    int mybox[] = {20, 20, 20};

```

```

generic_lattice vacuum(3,mybox,
                     default_partitioning<0>,
                     box_topology);
generic_field<float> u(vacuum);
generic_field<float> q(vacuum);
site x(vacuum);
site A(vacuum);
site B(vacuum);
A.set(3,3,3);
B.set(17,17,17);
float precision, old_u;
forallsitesandcopies(x) {
    u(x)=0;
    if(x==A)      q(x)=3;
    else if(x==B) q(x)=-5;
    else          q(x)=0;
};
do {
    precision=0;
    forallsites(x) if((x(0)>0) && (x(0)<mybox[0]-1) &&
                       (x(1)>0) && (x(1)<mybox[1]-1) &&
                       (x(2)>0) && (x(2)<mybox[2]-1)) {
        old_u=u(x);
        u(x)=(q(x)+u(x+0)+u(x-0)+u(x+1)+u(x-1)+u(x+2)+u(x-2))/6;
        precision+=pow(u(x)-old_u,2);
    };
    u.update();
    mpi.add(precision);
} while (sqrt(precision)>0.0001);
u.save("potential.dat");
mpi.close_wormholes();
};

```

The saved values can be used to produce a density plot representing the potential in the volume.

## 5.2. Total impedance in net of resistors

Another problem we want to solve is that of determining the total resistance between two arbitrary points (A and B) on a semi-conducting finite cylindrical surface. The cylinder is obtained starting from a torus topology and cutting the torus in one direction (say 0). The total resistance is probed by connecting the terminals of a current generator ( $J$ ) to the points A and B and evaluating the potential at points A and B using Ohm's law

$$R_{AB} = \frac{u(A) - u(B)}{J}. \quad (10)$$

The surface of the semiconductor is implemented as a periodic net of resistances so that each site has four resistances connected to it, Fig. 10. For simplicity we assume that all the resistances are the same (i.e. the material is homogeneous). A different choice would be equally possible.

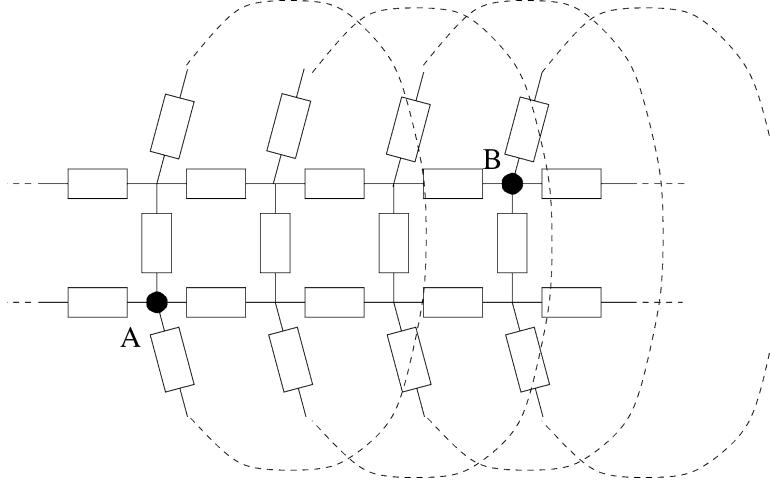


Fig. 10. Example of a cylindrical net of resistors.

The potential  $u(x)$  is computed using the Kirchoff law at each vertex

$$\sum_{i=0,1} R_i(x - \hat{i})[u(x - \hat{i}) - u(x)] + R_i(x)[u(x + \hat{i}) - u(x)] + J = 0 \quad (11)$$

that, solved in  $u(x)$ , leads to

$$u(x) = \frac{\sum_{i=0,1} R_i(x - \hat{i})u(x - \hat{i}) + R_i(x)u(x + \hat{i}) + J}{\sum_{i=0,1} R_i(x - \hat{i}) + R_i(x)}. \quad (12)$$

This equation is iterated on each site until convergence.

Here is the parallel program based on MDP 1.2.

```
// Program: application2.C
#define PARALLEL
#define Nproc 4
#include "MDP_Lib2.h"
#include "MDP_MPI.h"
void open_cylinder(int mu, int *x_dw, int *x, int *x_up,
                   int ndim, int *nx) {
    torus_topology(mu,x_dw,x,x_up,ndim,nx);
    if((mu==0) && (x[0]==0)) x_dw[0]=x[0];
    if((mu==0) && (x[0]==nx[0]-1)) x_up[0]=x[0];
}
float resistance(int x0, int x1, int mu) {
    return (Pi/100*x0);
}
int main(int argc, char **argv) {
    mpi.open_wormholes(argc,argv);
    int mybox[]={100,20};
    generic_lattice cylinder(2,mybox,default_partitioning<0>,
                            open_cylinder);
    generic_field<float> u(cylinder);
```

```

generic_field<float> r(cylinder,2);
site x(cylinder);
site A(cylinder);
site B(cylinder);
float precision, old_u;
float c, J=1, deltaJ, deltaR, Rtot, local_Rtot;
A.set(15,7);
B.set(62,3);
forallsitesandcopies(x) {
    u(x)=0;
    r(x,0)=resistance(x(0),x(1),0);
    r(x,1)=resistance(x(0),x(1),1);
}
do {
    precision=0;
    forallsites(x) {
        old_u=u(x);
        if(x==A) { c+=J; printf("%i\n", ME); };
        if(x==B) c-=J;
        deltaJ=c; deltaR=0;
        /* the next two lines take care of the finite cylinder */
        if(x+0!=x) { deltaJ+=u(x+0)*r(x,0); deltaR+=r(x,0); };
        if(x-0!=x) { deltaJ+=u(x-0)*r(x-0,0); deltaR+=r(x-0,0); };
        deltaJ+=u(x+1)*r(x,1); deltaR+=r(x,1);
        deltaJ+=u(x-1)*r(x-1,1); deltaR+=r(x-1,1);
        u(x)=deltaJ/deltaR;
        precision+=pow(u(x)-old_u,2);
    };
    u.update();
    mpi.add(precision);
} while (sqrt(precision)>0.00001);
Rtot=0;
if(A.is_in()) Rtot+=u(A)/J;
if(B.is_in()) Rtot-=u(B)/J;
mpi.add(Rtot);
if(ME==0) printf("R_A-R_B=%f\n", Rtot);
mpi.close_wormholes();
};

```

### 5.3. Ising model

One more full application we wish to consider is the study of the total magnetization of a spin system under a magnetic field (this is also known as the Ising model) as function of the temperature. As usual we define a lattice that represents the geometry of the spin system, and associate to the sites a field  $s(x) \in \{-1, +1\}$ . This is schematically represented in Fig. 11. The Hamiltonian of this system is

$$H = -\beta \sum_{x,y} s(x)\Delta(x, y)s(y) - \kappa \sum_x s(x)h(x), \quad (13)$$

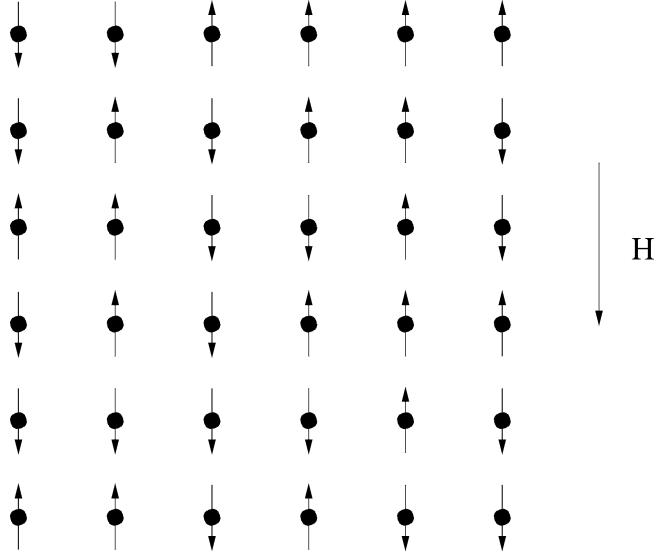


Fig. 11. Example of a 2D spin system.

where  $h(x)$  is an external magnetic field,  $\beta = 1/T$  is the inverse temperature and  $\kappa$  is a constant typical of the material (in principle one could promote  $\kappa$  to be a field for non-homogeneous materials.  $\Delta(x, y)$  is 1 if  $|x - y| = 1$ , 0 otherwise. In the example we consider a two dimensional spin system but the extension to arbitrary dimensions is trivial: just change mybox. Moreover we implement a simple montecarlo\_multihit algorithm [7] to minimizes the action in presence of thermal fluctuations.

Here is the parallel program based on MDP 1.2.

```
// Program: application3.C
#define PARALLEL
#include "MDP_Lib2.h"
#include "MDP_MPI.h"

// /////////////////////////////////
class scalar_field: public generic_field<float> {
public:
    int ndim,nc;
    scalar_field(generic_lattice &a) {
        allocate_field(a, 1);
        ndim=a.ndim;
    };
    float &operator() (site x) {
        return *(address(x));
    };
};

// ///////////////////////////////
class ising_field: public generic_field<int> {
public:
    int ndim,nc;
```

```

float beta, kappa;
ising_field(generic_lattice &a) {
    allocate_field(a, 1);
    ndim=a.ndim;
};

int &operator() (site x) {
    return *(address(x));
};

friend void set_cold(ising_field &S) {
    site x(S.lattice());
    forallsites(x) S(x)=1;
};

friend void montecarlo_multihit(ising_field &S, scalar_field &H,
int n_iter=10, int n_hits=3) {
    int iter, parity, hit, new_spin, mu;
    float delta_action;
    site x(S.lattice());
    for(iter=0; iter<n_iter; iter++) {
        for(parity=0; parity<=1; parity++) {
            forallsitesofparity(x,parity) {
                for(hit=0; hit<n_hits; hit++) {
                    delta_action=S.kappa*H(x);
                    for(mu=0; mu<S.ndim; mu++)
                        delta_action-=S(x+mu)+S(x-mu);
                    new_spin=(S.lattice().random(x).plain()>0.5)?1:-1;
                    delta_action*=S.beta*(new_spin-S(x));
                    if(exp(-delta_action)>S.lattice().random(x).plain())
                        S(x)=new_spin;
                };
                S.update(parity);
            };
        };
    };
};

friend float average_spin(ising_field &S) {
    float res=0;
    site x(S.lattice());
    forallsites(x) res+=S(x);
    mpi.add(res);
    return res/(S.lattice().nvol_gl);
};

int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);

    int conf,Ntherm=100,Nconfig=100;
    int mybox[ ]={64,64};
}

```

```

generic_lattice mylattice(2, mybox);
ising_field      S(mylattice); /* ths spin field +1 or -1      */
scalar_field      H(mylattice); /* the external magnetic field */
site              x(mylattice);
JackBoot          jb(Nconfig,1);
jb.plain(0);

S.beta=0.5;           /* inverse square temperature           */
S.kappa=0.1;          /* coupling with the total extarnal field */

/* setting initial conditions */
forallsites(x) {
    S(x)=1; /* initial spins set to +1 */
    H(x)=0; /* external magnetic field set to zero */
};

S.update();
H.update();

if(ME==0)
    printf("Beta\tMagnetization\terror\n");

/* looping on values of beta */
for(S.beta=0.20; S.beta<0.60; S.beta+=0.01) {
    /* termalizing configuration */
    for(conf=0; conf<Ntherm; conf++)
        montecarlo_multihit(S,H);

    /* generating Nconfig configurations for each value of beta */
    for(conf=0; conf<Nconfig; conf++) {
        montecarlo_multihit(S,H);
        jb(conf,0)=average_spin(S);
    };
    if(ME==0)
        printf("%f\t%f\t%f\n",
            S.beta, abs(jb.mean()), jb.j_err());
};

mpi.close_wormholes();
return 0;
};

```

The output of this program (for  $h(x) = 0$ ) has been plotted and reported in Fig. 12. It clearly shows that when the temperature is high ( $\beta = 1/T$  is low) the average magnetization fluctuates around zero while, when the temperature is low ( $\beta$  if high), all the spins tend to become parallel giving origin to a total magnetization different from zero. The critical temperature where the phase transition (the jump) occurs is, for the 2D model

$$T_c = \frac{1}{\beta_c} = \frac{2}{\log(1 + \sqrt{2})} \simeq 2.269. \quad (14)$$

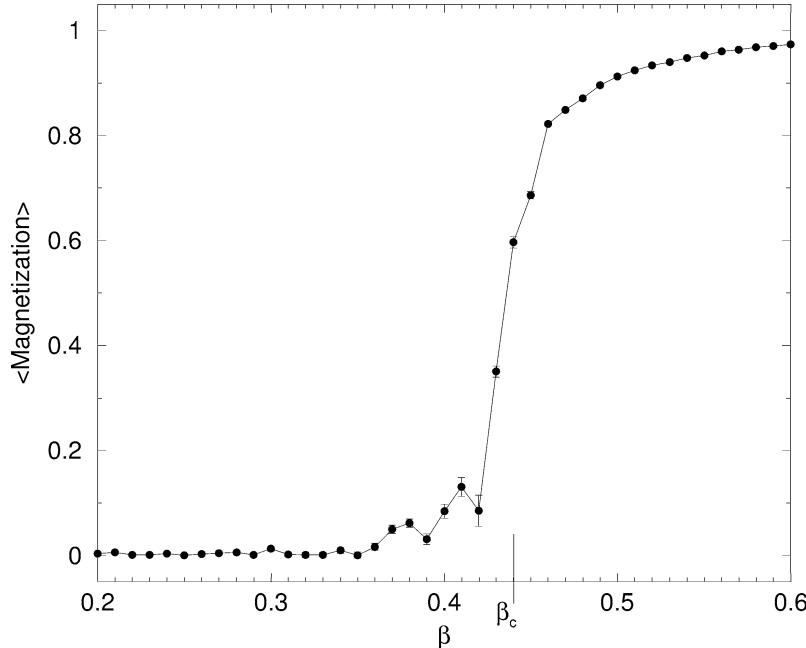


Fig. 12. Total average magnetization in a two-dimensional spin system as function of the inverse temperature. The jump corresponds to a phase transition.

Note that the only reference to the number of dimensions is in the function `main()` in the lines

```
int mybox[ ]={64, 64};  
generic_lattice mylattice(2, mybox);
```

These are the only lines to change to generalize the program to more than 2D. For bigger volumes one may also want to change the total number of computed configurations `Nconfig` and the steps to thermalization `Ntherm`.

#### 5.4. Solid under stress

Many other parallel applications could be developed relatively easy using MDP 1.2. For example, one may be interested in studying the deformation of a solid under stress. The solid can be approximated with a finite lattice of given arbitrary shape and each site would be connected to its next-neighbor and next-next-neighbor sites with springs. One needs at least 4 fields on this lattice:  $M$ ,  $K$ ,  $G$ ,  $Q$  and  $P$ , where  $M$  is the mass density of the solid;  $K$  the spring constant;  $G$  a parameter determining the internal friction of the solid;  $Q$  the physical (spatial) coordinates of each site,  $P$  the momentum associated to each site. The problem of determining the deformations of this solid can be solved by iterating a discretized form of the Hamilton equations

$$P_{t+\Delta t}(x) - P_t(x) = \frac{\partial H(P, Q, K, G)}{\partial Q(x)} \Delta t = F(x, Q, P, K, G) \Delta t, \quad (15)$$

$$Q_{t+\Delta t}(x) - Q_t(x) = \frac{\partial H(P, Q, K, G)}{\partial P(x)} \Delta t = P(x)/M(x) \Delta t. \quad (16)$$

These equations translates into

```
float delta_t=0.000001; /* a small nonzero positive number */  
forallsites(x)
```

```

for(mu=0; mu<3; mu+) {
    F[mu]/* some function of K,G,Q,P */
    P(x,mu)=P(x,mu)+F(mu)*delta_t;
    Q(x,mu)=Q(x,mu)+P(x,mu)/m*delta_t;
}
P.update();
Q.update();

```

The expression of the function  $F$  can be quite complicated depending on how one models the solid.

### 5.5. Lattice QCD

MDP 1.2 has been used to develop a parallel package for large scale Lattice QCD simulations called FermiQCD [8,9]. Here we only list some of its main features.

The typical problem in QCD (Quantum Chromo Dynamics) is that of determining the correlation functions of the theory as a function of the parameters. From the knowledge of these correlation functions one can extract particle masses and matrix elements to compare with experimental results from particle accelerators.

On the lattice, each correlation function is computed numerically as the average of the corresponding operator applied to elements of a Markov chain of gauge field configurations. Both the processes of building the Markov chain and of measuring operators involve quasi-local algorithms. Some of the main features implemented in FermiQCD are:

- works on a single process PC,
- works in parallel with MPI,
- arbitrary lattice partitioning,
- parallel I/O (partitioning independent),
- arbitrary space-time dimension,
- arbitrary gauge group  $SU(n)$ ,
- anisotropic Wilson gauge action,
- anisotropic Wilson fermionic action,
- anisotropic Clover improved action,
- D234 improved action,
- Kogut–Susskind improved action,
- ordinary and stochastic propagators,
- minimal residue inversion,
- stabilized biconjugate gradient inversion,
- twisted boundary conditions for large  $\beta$  numerical perturbation theory (all fields),
- reads existing CANOPY data,
- reads existing UKQCD data,
- reads existing MILC data.

In Table 2 we show a few examples of FermiQCD Object Oriented capabilities (compared with examples in the standard textbook notation for Lattice QCD [9]).

## 6. Timing and efficiency issues

The issue of efficiency of a parallel program is not a simple one, in fact there are many factors to take into account. In particular the efficiency scales with the number of parallel processes, with the size of the data that need to be communicated between processes, and with the total volume of the lattice.

Table 2

Example of FermiQCD statements

QCD	FermiQCD
Algebra of Euclidean gamma matrices	
$A = \gamma^\mu \gamma^5 e^{3i\gamma^2}$	Matrix A; A=Gamma[mu]*Gamma5*exp(3*I*Gamma[2]);
Multiplication of a fermionic field for a spin structure	
$\forall x: \chi(x) = (\gamma^3 + m)\psi(x + \hat{\mu})$	/* assuming the following definitions generic_lattice space_time(...); fermi_field chi(space_time,Nc); fermi_field psi(space_time,Nc); site x(space_time); */ forallsites(x) chi(x)=(Gamma[3]+m)*psi(x+mu);
Translation of a fermionic field	
$\forall x, a: \chi_a(x) = U(x, \mu)\psi_a(x + \hat{\mu})$	forallsites(x) for(a=0; a<psi.Nspin; a++) chi(x,a)=U(x,mu)*psi(x+mu,a);

The dependence of the efficiency on these variables cannot be in general factorized because of non-linearities introduced by modern computer architectures (for example, caches effects, finite memory, memory speed/CPU speed, latencies, etc.).

To give an idea of how a parallel computer program based on MDP scales with the number of processes, we consider the following program.

```
// program: laplace.C
#define PARALLEL
#include "MDP_Lib2.h"
#include "MDP_MPI.h"

int main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);
    int box[4]={8,12,12,12};
    generic_lattice space(4,box);
    Matrix_field U(space,3,3);
    Matrix_field V(space,3,3);
    site x(space);
    forallsites(x) {
        V(x)=space.random(x).SU(3);
        U(x)=0;
    };
    U.update();
```

```

V.update();
for(int i=0; i<10000; i++) {
    forallsites(x)
        U(x)=0.125*(cos(U(x)+V(x))+
                      U(x+0)+U(x-0)+
                      U(x+1)+U(x-1)+
                      U(x+2)+U(x-2)+
                      U(x+3)+U(x-3));
    /* uncomment to print convergence
    if(ME==0) {
        x.set(0,0,0,0);
        printf("%f\n", real(U(x,0,0)));
    };
    */
    U.update();
}
V.save("V_field.dat");
U.save("U_field.dat");
mpi.close_wormholes();
return 0;
};

```

It solves a 4-dimensional Laplace equation for a field of  $3 \times 3$  matrices, and performs some parallel IO as well. The reader can work out the details of the program as an exercise.

We ran the same program for different lattice volumes  $V = T \times L^3$  and on different sets of processors ( $N$  = number of processors). Each processor (node) is a 700 MHz PentiumIII PC. The nodes are connected using Myrinet cards/switches.

For each run we measured the total time  $t(T, L, N)$  and we then computed the efficiency according with two possible definitions:<sup>14</sup>

$$e_1(T, L) = 100 \frac{t(T, L, 1)}{N t(T, L, N)}, \quad (17)$$

$$e_2(N, L, x) = 100 \frac{t(x, L, 1)}{t(Nx, L, N)}. \quad (18)$$

The first definition is applicable to computations in which the problem size is fixed and one varies the numbers of processors. The second definition is applicable to computations for which the size of the problem increases with the number of processors available. In both cases the efficiency is normalized to 100%. In an ideal world (without latencies and with instantaneous communications) both these efficiencies should be constant and equal to 100%.

In Figs. 13, 14 we plot  $e_1$  and  $e_2$  for our runs and the maximum time spent in communications by the nodes (as computed by MDP) that we call  $r$ . The normalizations are given by

$$t(60, 4, 1) = 788 \text{ s},$$

$$t(60, 8, 1) = 6417 \text{ s},$$

$$t(60, 12, 1) = 6417 \text{ s},$$

$$t(2, 8, 1) = 201 \text{ s},$$

---

<sup>14</sup> In the context of Lattice QCD simulation, up to an overall normalization, the first definition is the one used at Fermilab to benchmark parallel software, the second is the definition used by the MILC Collaboration.

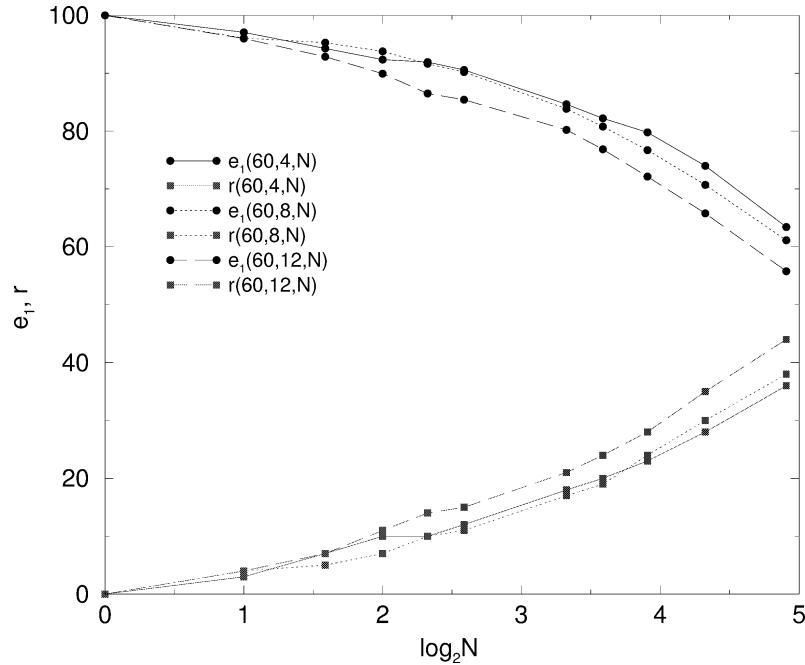


Fig. 13. Efficiency  $e_1$  as function of  $N$  and  $L$ , for  $T = 60$  (the number 60 has been chosen because it has the biggest number of integer factors within the number of available processors).

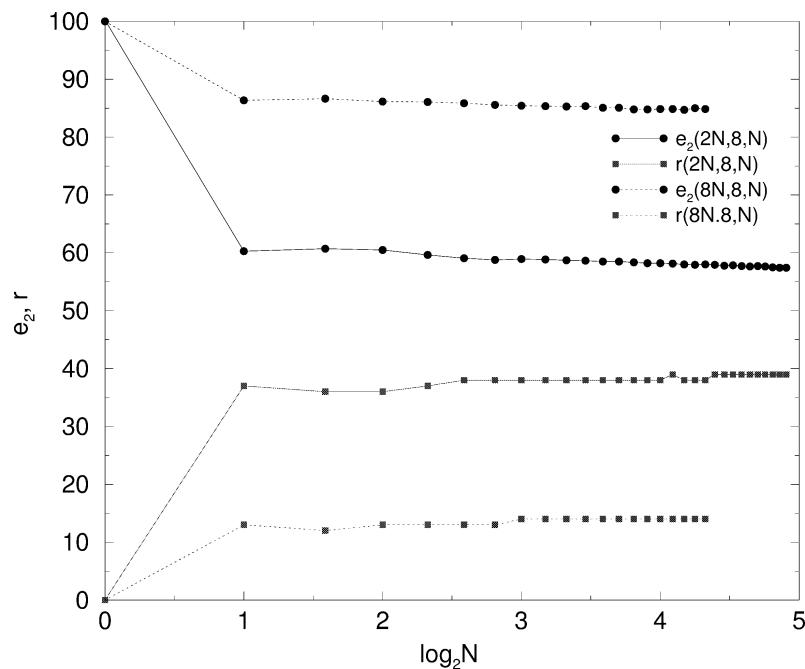


Fig. 14. Efficiency  $e_2$  as function of  $N$ , for  $T = 2N$  and  $L = 8$ .

$$t(8, 8, 1) = 834 \text{ s},$$

These numbers can be further reduced by writing a user defined function that optimizes the critical computation done in the line

$$U(x) = 0.125 * (\cos(U(x)) + V(x)) + \dots$$

From the plots we observe that with good approximation

$$e_i \simeq 100 - r \quad (19)$$

for both the definitions of efficiency.

Eventually, for  $N$  bigger than some  $N_0$ , the total time becomes totally dominated by communication (the time spent in communications goes to 100%) and  $e_1$  starts to scale as  $1/N$ .

Note that the answer to the question “how many processors should I use?” is not addressed in this paper because, in financial terms, this depends on the subjective utility function of each individual user. The only general conclusion is that  $N$  should be less than  $N_0$ .

A different algorithm may scale in a different way even if the qualitative behavior of the efficiency should be the same. Some algorithms may require more communications than others and may result in a bigger loss of efficiency when running in parallel.

## Acknowledgements

I thank Chris Sachrajda, Jonathan Flynn and Luigi Del Debbio of the University of Southampton (UK) where the MDP project started.

The development of the code MDP 1.2 in its final (but not yet definitive) form and its main application, FermiQCD, have greatly benefit from discussions and suggestions with members of the Fermilab Theory Group. In particular I want to thank Paul Mackenzie, Jim Simone, Jimmy Juge, Andreas Kronfeld and Estia Eichten for comments and suggestions. Moreover I freely borrowed many ideas from existing Lattice QCD codes (including the UKQCD code [10], the MILC code [11], QCDF90 [12] and CANOPY [13]). I here thank their authors for making their software available to me.

## References

- [1] G. Satir, D. Brown, C++ The Core Language, O'Reilly and Associates, 1995.
- [2] N. Cabibbo, E. Marinari, A new method for updating  $SU(N)$  matrices in computer simulations of gauge theories, Phys. Lett. 119B (1982) 387.
- [3] J. Shao, D. Tu, The Jackknife and Bootstrap, Springer Verlag, 1995.
- [4] P.D. Coddington, Random Number Generators for Parallel Computers, 1997 (unpublished).
- [5] G.A. Marsaglia, A current view on random number generators, in: L. Balliard (Ed.), Computational Science and Statistics: The Interface, Elsevier, Amsterdam, 1985.
- [6] P.S. Pacheco, Parallel Programming with MPI, San Francisco, CA, Morgan Kaufmann, 1997.
- [7] G. Banhot, The Metropolis algorithm, Rep. Prog. Phys. 51 (1988) 429.
- [8] M. Di Pierro, Matrix distributed processing and FermiQCD, hep-lat/0011083.
- [9] M. Di Pierro, From Monte Carlo integration to lattice quantum chromodynamics: An introduction, hep-lat/0009001.
- [10] The UKQCD collaboration web page: <http://www.ph.ed.ac.uk/ukqcd/>.
- [11] The MILC collaboration web page: <http://physics.indiana.edu/~sg/milc.html>.
- [12] I. Dasgupta, A.R. Levi, V. Lubicz, C. Rebbi, Comput. Phys. Commun. 98 (1996) 365–397.
- [13] M. Fischler, J. Hockney, P. Mackenzie, M. Uchima, Canopy 7.0 manual, Fermilab preprint TM-1881. PDF text available at web site <http://fnalpubs.fnal.gov/archive/tm/TM-1881.pdf>.

**LICENSE for MDP 1.2  
(including examples and applications suchas FermiQCD)**

- MDP 1.2 has been created by Massimo Di Pierro. MDP 1.2 is a property of the author. (The application FermiQCD, also covered by this License is a joined property of the author and of Fermilab).
- MDP 1.2 is free of charge for educational and research institutions worldwide.
- You may copy and distribute exact replicas of MDP 1.2 as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice including the author's name and the disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipient of MDP 1.2 a copy of this License along with the software.
- You may modify your copy or copies of MDP 1.2 or any portion of it and distribute such modifications or work under the terms of Section 1 and 2 above, provided that the modified content carries prominent notices stating that it has been changed, the exact nature and content of the changes, and the date of any change.
- BECAUSE MDP 1.2 IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR IT. EXCEPT WHEN OTHERWISE STATED IN WRITING. THE AUTHOR PROVIDES MDP 1.2 "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. THE ENTIRE RISK OF USE OF THIS SOFTWARE IS WITH THE RECIPIENT. SHOULD MDP 1.2 OR ONE OF ITS APPLICATIONS PROVE FAULTY, INACCURATE, OR OTHERWISE UNACCEPTABLE YOU ASSUMES THE COST OF ALL NECESSARY REPAIR OR CORRECTION.
- To request MDP 1.2 (including the examples and applications described in this paper) please sign this license and fax or mail this page to the author:

Massimo Di Pierro  
Email: mdp@fnal.gov  
Fax: 001-630-840-5435  
Address: MS 106, Fermilab, Batavia, IL 60510-0500 (USA)

Name: .....  
Date: .....  
Email: .....  
Signature: .....



ELSEVIER

Nuclear Physics B (Proc. Suppl.) 106 (2002) 1034–1036

---

---

NUCLEAR PHYSICS B  
PROCEEDINGS  
SUPPLEMENTS

---

www.elsevier.com/locate/npe

# FermiQCD: A tool kit for parallel lattice QCD applications

[ <http://latticeqcd.fnal.gov/software/fermiqcd/> ]

Massimo Di Pierro<sup>a\*</sup>

<sup>a</sup> Fermilab, Kirk and Pine St., Batavia, Illinois 60563, USA

We present here the most recent version of FermiQCD, a collection of C++ classes, functions and parallel algorithms for lattice QCD, based on Matrix Distributed Processing. FermiQCD allows fast development of parallel lattice applications and includes some SSE2 optimizations for clusters of Pentium 4 PCs.

## 1. Introduction

FermiQCD is a collection of classes, functions and parallel algorithms for lattice QCD [1], written in C++. It is based on Matrix Distributed Processing<sup>2</sup> (MDP) [2]. The latter is a library that includes C++ methods for matrix manipulation, advanced statistical analysis (such as Jackknife and Bootstrap) and optimized algorithms for interprocess communications of distributed lattices and fields. These communications are implemented using Message Passing Interface (MPI) but MPI calls are hidden to the high level algorithms that constitute FermiQCD.

FermiQCD works also on single processor computers and, in this case, MPI is not required.

## 2. FermiQCD overview

The basic fields defined in FermiQCD are:

### class gauge\_field:

List of implemented algorithms:

- heatbath algorithm
- anisotropic heatbath
- $O(a^2)$  heatbath

These algorithms work for arbitrary gauge groups  $SU(N_c)$ , for arbitrary lattice dimensions and topologies. FermiQCD also supports arbitrarily

twisted boundary conditions for large  $\beta$  computations and studies of topology.

### class fermi\_field:

List of implemented algorithms:

- multiplication by  $Q = (\not{D} + m)$ , for Wilson and Clover actions, for isotropic and anisotropic lattices
- minimal residue inversion for  $Q$
- stabilized biconjugate gradient (BiCGStab) inversion for  $Q$
- Wupperthal smearing for the field
- stochastic propagators

These algorithms work for arbitrary gauge groups  $SU(N_c)$  and for arbitrary topologies in 4 dimensions. The multiplication by  $Q$ , clover (isotropic and anisotropic), for  $SU(3)$ , is optimized using Pentium 4 SSE2 instructions in assembler language. This implementation is based on the assembler macro functions written by Martin Lüscher [3]

### class fermi\_propagator:

This is an implementation of ordinary quark propagators. A `fermi_propagator` can be generated using any of the inversion algorithms of a `fermi_field`.

### class staggered\_field:

Kogut-Susskind (KS) fermion. List of implemented algorithms:

\*Poster presented at Lattice 2001, Berlin

<sup>2</sup><http://www.phoenixcollective.org/mdp>

- multiplication by  $Q$ , for unimproved and  $O(a^2)$  (Asqtad) improved actions [4]
- BiCGStab inversion for  $Q$
- BiCGStab inversion for  $Q$  using the UML decomposition [5]
- function `make_meson`

These algorithms work for arbitrary gauge groups  $SU(N_c)$  and for an arbitrary even number of dimensions (except `make_meson`). The multiplication by  $Q$ , both improved and unimproved, for  $SU(3)$ , is optimized using Pentium 4 SSE2 instructions in assembler language. In the unimproved case only half of the SSE2 registries are used and there is room for an extra factor two in speed. The function `make_meson` builds any meson propagator (made out of staggered quarks) for arbitrary Spin⊗Flavour structure. This algorithm is described in ref. [6]

#### `class staggered_propagator:`

This is an implementation of the staggered propagator consisting of 16 sources contained in a  $2^4$  hypercube at the origin of the lattice. A `staggered_propagator` can be used to propagate any hadron from the hypercube at the origin of the lattice to any other hypercube without extra inversions.

All fields in FermiQCD inherit the standard I/O methods of MDP (`save` and `load`) and the file format is independent on the lattice partitioning over the parallel processes. These I/O functions, as well as all the FermiQCD algorithms, are designed to optimize interprocess communications.

### 3. Example

We present here, as an example, a full program that generates 100  $SU(3)$  gauge configurations ( $U$ ), starting from a hot one. On each configuration it computes a pion propagator (`pion`) made of  $O(a^2)$  improved quark propagators and prints it out. These propagators are computed using the SSE2 optimized clover action and the BiCGStab inversion algorithm. The program works in parallel.

---

```
#define PARALLEL
#include "fermiqcd.h"

void main(int argc, char **argv) {
    mpi.open_wormholes(argc, argv);

    int t,a,b,conf;
    int nc=3, box[4]={16,8,8,8};
    generic_lattice L(4,box);
    gauge_field U(L,nc);
    fermi_propagator S(L,nc);
    site x(L);
    float pion[16];
    U.param.beta=5.7;
    S.param.kappa=0.1345;
    S.param.cSW=1.5;

    default_fermi_action=
        mul_Q_Luscher;
    default_inversion_method=
        BiCGStab_inversion;

    set_hot(U);
    heatbath(U,100);
    for(conf=0; conf<100; conf++) {
        heatbath(U,30);
        compute_em_field(U);
        generate(S,U);
        for(t=0; t<16; t++) pion[t]=0;
        forallsites(x)
            for(a=0; a<4; a++)
                for(b=0; b<4; b++)
                    pion[x(TIME)]+=
                        real(trace(S(x,a,b)*
                        hermitian(S(x,b,a))));
        mpi.add(pion, 16);
        if(ME==0) for(t=0; t<16; t++)
            printf("%i %f\n", t, pion[t]);
    }
    mpi.close_wormholes();
}
```

---

Comments:

- $L$  is the user-defined name of the lattice ( $16 \times 8^3$ )

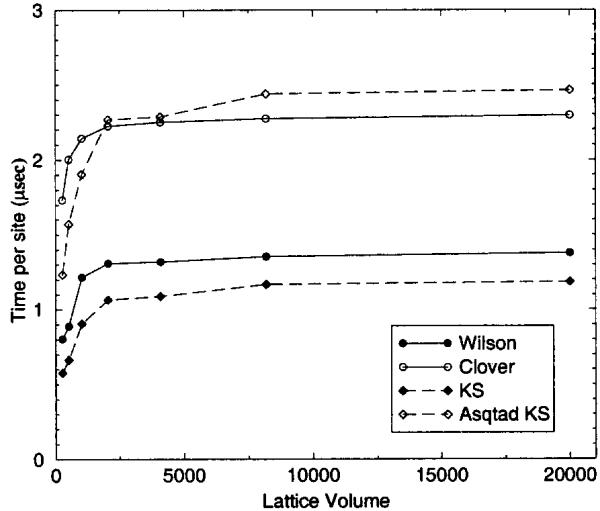


Figure 1. Time per site in  $\mu$ sec for `mul_Q_Luscher` (Wilson, clover, KS and Asqtad KS in single precision).

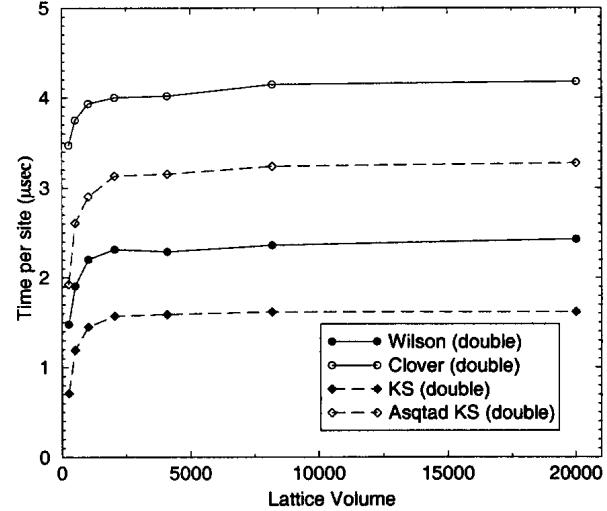


Figure 2. Time per site in  $\mu$ sec for `mul_Q_Luscher` (Wilson, clover, KS and Asqtad KS in single precision).

- $U$ ,  $S$  and  $x$  are the gauge field, the fermi propagator and an auxiliary site variable defined on the lattice  $L$
- `default_fermi_action` is a pointer to the function that implements the action to be used. `mul_Q_Luscher` is one of the the built-in clover actions, optimized for Pentium 4.
- `default_inversion_method` is a pointer to the function that implements the inversion algorithm (minimal residue or BiCGStab)
- `compute_em_field` computes the chromo-electro-magnetic field required by the action<sup>3</sup>.
- `generate` computes the quark propagator  $S$  on the given gauge configuration  $U$ .
- `forall_sites(x)` is a parallel loop on  $x$ . Each processor loops on the local sites.
- `mpi.add(pion,16)` sums the vector `pion[16]` in parallel.

<sup>3</sup>The chromo-electro-magnetic field is a member variable of the gauge field. FermiQCD has almost no global variables except pointers to the functions that implement the algorithms.

- `if(ME==0)` guarantees that only one processor performs the output.

#### 4. Benchmarks

In fig. 1 and fig. 2 we report some benchmarks for the multiplication by  $Q$  of a fermionic field, for the different actions (using a single CPU Pentium 4 PC running at 1.4 GHz, Linux 2.4 and gcc 2.95.3).

This work was performed at Fermilab (U.S. Department of Energy Lab (operated by the University Research Association, Inc.), under contract DE-AC02-76CHO3000.

#### REFERENCES

1. M. Di Pierro, hep-lat/0009001. (Lattice QCD tutorial with examples in FermiQCD)
2. M. Di Pierro, hep-lat/0004007. (Updated tutorial on MDP) To be published on CPC.
3. M. Lüscher, *these proceedings*
4. G. P. Lepage, Phys.Rev. D59 (1999) 074502
5. K. Orginos *et al.* (MILC), Phys.Rev. D59 (1999) 014501
6. M. Di Pierro, *in preparation*

## Nonperturbative tuning of $O(a^2)$ improved staggered fermions

Massimo Di Pierro<sup>a\*</sup> Paul Mackenzie<sup>a</sup>

<sup>a</sup> Fermilab, PO Box 500, Batavia, IL 60563, USA

We perform a nonperturbative tuning of the coefficients in the  $O(a^2)$  improved action for staggered fermions. The mass splitting for the pions of different doubler flavor is used as a measure of the symmetry breaking effects introduced by  $O(a^2)$  discretization errors. We find that the flavor nondegeneracy can be somewhat reduced but not eliminated by such a tuning, indicating the need for new terms in the action to reduce the nondegeneracy.

### 1. INTRODUCTION

Staggered fermions offer the possibility of doing unquenched calculations on current computers with far less simulation time than Wilson type fermions. In their simplest form, however, they suffer from several well-known problems which must be addressed before they can be used effectively [1]. One significant problem with ordinary staggered fermions is the large flavor nondegeneracy, worst in the pion sector. At tree level, it arises from transitions between doubler quarks of different flavors induced by gluons of momentum  $\pi$  [2,3]. In Ref. [4], Lepage showed how to turn the “fat-link” improvement of the MILC collaboration [5], which suppresses the coupling of quarks to these gluons, into a tree level  $O(a^2)$  improved action by proper tuning of the coefficients and the inclusion of an additional term.

Monte Carlo calculations showed a large reduction in pion flavor nondegeneracy with this action [6]. Significant flavor breaking still remains, however, so further improvement is desirable for high precision calculations. In this work, we perform a nonperturbative determination of gluonic corrections to the tadpole improved tree level  $a^2$  improved staggered action (called the “Asqtad” action by the MILC collaboration). We find a small improvement in pion flavor breaking, but not the large reduction that is still desirable. Therefore, incorporation of additional operators into the action is needed [1].

\*Talk presented by Massimo Di Pierro

### 2. THE IMPROVED STAGGERED ACTION

We define a modified Asqtad (mAsqtad) action, built in two steps: First, from the naive fermionic action, one modifies the definition of the covariant derivative:

$$D_\mu^{\{c_i\}} \psi(x) = V_\mu \psi(x + \mu) - V_{-\mu} \psi(x - \mu) \quad (1)$$

$$- \frac{1 + c_5}{24u_0^2} [(U_\mu)^3 \psi(x + 3\mu) - (U_{-\mu})^3 \psi(x - 3\mu)]$$

where  $V_\mu$  is a fat link defined as

$$V_\mu \equiv \frac{5}{8}(1 + c_0)U_\mu$$

$$+ \frac{1 + c_1}{16u_0^2} \sum_\nu U_{\pm\nu} U_\mu U_{\mp\nu} +$$

$$+ \frac{1 + c_2}{64u_0^4} \sum_{\nu, \rho} U_{\pm\rho} U_{\pm\nu} U_\mu U_{\mp\nu} U_{\mp\rho}$$

$$+ \frac{1 + c_3}{384u_0^6} \sum_{\nu, \rho, \sigma} U_{\pm\sigma} U_{\pm\rho} U_{\pm\nu} U_\mu U_{\mp\nu} U_{\mp\rho} U_{\mp\sigma}$$

$$- \frac{1 + c_4}{16u_0^4} \sum_\nu (U_{\pm\nu})^2 U_\mu (U_{\mp\nu})^2 \quad (2)$$

(the indices in the sums are always different from  $\mu$  and among each other). The choice of the coefficients  $c_i = 0$  corresponds to the Asqtad action.

Second, the fermion  $\phi(x)$  is mapped into a scalar field  $\chi(x')$  by the relation

$$\phi(x)_\alpha^a = \sum_{A \in [2^4]} (\gamma_1^{A_1} \gamma_2^{A_2} \gamma_3^{A_3} \gamma_4^{A_4})_\alpha^a \chi(x + A) \quad (3)$$

where  $\alpha$  is the spin index,  $a$  is the flavor index,  $[2^4]$  is a 4D hypercube. Note that eq. (3) is invertible

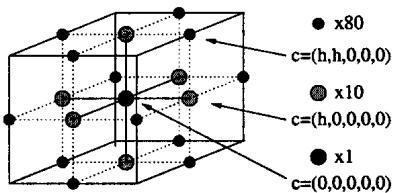


Figure 1. Three dimensional representation of the five dimensional space of coefficients and the points we considered.

only if the fermion and the scalar live on different lattices, i.e. if the former lives on the blocked lattice of the latter [7]. We use this prescription to build the **15**-plet of pions of  $SU(4)$  flavor.

Because of the remnant discrete flavor symmetry there are only seven inequivalent pions. We identify the seven inequivalent pions by their  $SU(4)$  flavor structure

$$\xi = \gamma^5, \gamma^0\gamma^5, \gamma^3\gamma^5, \gamma^1\gamma^2, \gamma^3\gamma^4, \gamma^3, \gamma^3 \quad (4)$$

Our goal is that of tuning the coefficients  $c_i$  around zero to reduce the mass splitting among these pions.

### 3. Computation

Our computation was performed on 113  $O(a^2)$  improved gauge configurations at  $\beta = 7.4$  and  $u_0 = 0.8629$  on a  $24 \times 8^3$  lattice. We preceded in the following way:

We chose a finite set of points in the space of the coefficients. For each point, we performed a lattice computation of the Goldstone pion and we fine tuned the quark mass  $m$  in order to reproduce a mass for the Goldstone pion,  $M_{\gamma^5}$ , equal to  $(0.49 \pm 0.01)a^{-1}$  (this is an arbitrary number). This fine tuning is required since the mass renormalizes in different ways for the different choices of coefficients and we need to impose a physical renormalization condition. Then, for each point, we computed the whole spectrum of pions using the corresponding fine tuned value for the quark mass.

We chose 91 points in the space of coefficients, namely  $c_i = 0$ ,  $c_i = \pm h\delta_{ij}$ ,  $c_i = \pm h\delta_{ij} \pm h\delta_{ik}$  for

every value of  $j$  and  $k \neq j$  ( $h = 0.5$ ). These points are represented in Fig. 1. This choice enables us to evaluate numerically the first ( $M'_{\xi,i}$ ) and second derivative ( $M''_{\xi,ij}$ ) of the pion masses in respect to each coefficient of the action (keeping fixed the renormalization condition, i.e. the mass of the Goldstone pion).

This work amounts to more than 10000 fermionic inversions of the action and more than 2000 fits of pion propagators; it was performed on the Fermilab QCD80 cluster.

### 4. Results

We expand the pion masses as functions of the coefficients in the action, in Taylor series up to second order

$$M_\xi(c_i) = M_\xi^0 + \sum_i M'_{\xi,i}c_i + \frac{1}{2} \sum_{i,j} M''_{\xi,ij}c_i c_j \quad (5)$$

and we define

$$F_\xi(c_i) = M_\xi^2(c_i) - M_{\gamma^5}^2 \quad (6)$$

where  $M_{\gamma^5} = \text{const.}$  because of our choice of renormalization condition. In Fig. 2 we report sections of the function  $F_\xi$ . Each plot shows the value of  $F$  of all pions,  $\xi$ , when we vary one single coefficient.

The first result that is already visible from the plots is that the spectrum is very mildly dependent on the action and there is no obvious direction in the space of coefficients where the pion mass splitting gets reduced for all pions at once.

We report here the values for the meson masses corresponding to the Asqtad action (for a fine tuned light quark mass of  $m = 0.0347a^{-1}$ )

$$\begin{aligned} M_{\gamma^5}^0 &= 0.592 \pm 0.003 & M_{\gamma^0\gamma^5}^0 &= 0.801 \pm 0.017 \\ M_{\gamma^3\gamma^5}^0 &= 0.758 \pm 0.006 & M_{\gamma^1\gamma^2}^0 &= 1.021 \pm 0.030 \\ M_{\gamma^3\gamma^4}^0 &= 0.923 \pm 0.012 & M_{\gamma^3}^0 &= 1.092 \pm 0.032 \\ M_{\gamma^4}^0 &= 0.977 \pm 0.015 \end{aligned}$$

and their first derivatives

$$\begin{aligned} M'_{\gamma^0\gamma^5} &= (-0.022, -0.007, -0.008, 0.095, -0.036) \\ M'_{\gamma^3\gamma^5} &= (-0.040, -0.062, -0.020, 0.108, -0.007) \\ M'_{\gamma^1\gamma^2} &= (-0.005, -0.026, -0.016, 0.078, -0.035) \end{aligned}$$

$$M'_{\gamma^3 \gamma^4} = (-0.036, -0.074, -0.029, 0.103, -0.010)$$

$$M'_{\gamma^3} = (+0.015, +0.006, -0.014, 0.071, -0.019)$$

$$M'_{\gamma^4} = (-0.022, -0.074, -0.034, 0.067, -0.022)$$

One should notice that the signs of the first derivatives are consistent for the different pions, but they are very small. They are of the same order of magnitude as the error on  $M_\xi^0$ . This means that within a more than reasonable range ( $c_i \in [-1, +1]$ ) the pion's splitting does not vary more than  $2\sigma$  where  $\sigma$  is its statistical error.

We conclude that the flavor nondegeneracy of the tadpole improved tree level improved staggered action may be somewhat improved by a nonperturbative tuning of the coefficients, by perhaps 10% or less. However, for further dramatic reduction in flavor breaking, new terms must be added to the action, as in Ref. [1].

This work was performed at Fermilab (U.S. Department of Energy Lab (operated by the University Research Association, Inc.), under contract DE-AC02-76CHO3000.

## REFERENCES

1. H. Trottier, G. P. Lepage, P. B. Mackenzie, Q. Mason, and M. Nobes, *in these proceedings*.
2. G. P. Lepage, Nucl. Phys. Proc. Suppl. **60A** (1998) 267 [hep-lat/9707026].
3. J. F. Lagae and D. K. Sinclair, Phys. Rev. **D59** (1999) 014511 [hep-lat/9806014].
4. G. P. Lepage, Phys. Rev. **D59** (1999) 074502 [hep-lat/9809157].
5. MILC collaboration (Kostas Orginos *et al.*), Phys. Rev. **D59** (1999) 014051 [hep-lat/9805009].
6. MILC Collaboration (Kostas Orginos *et al.*), Phys. Rev. **D60** (1999) 054503 [hep-lat/9903032].
7. Massimo Di Pierro, *in preparation*.

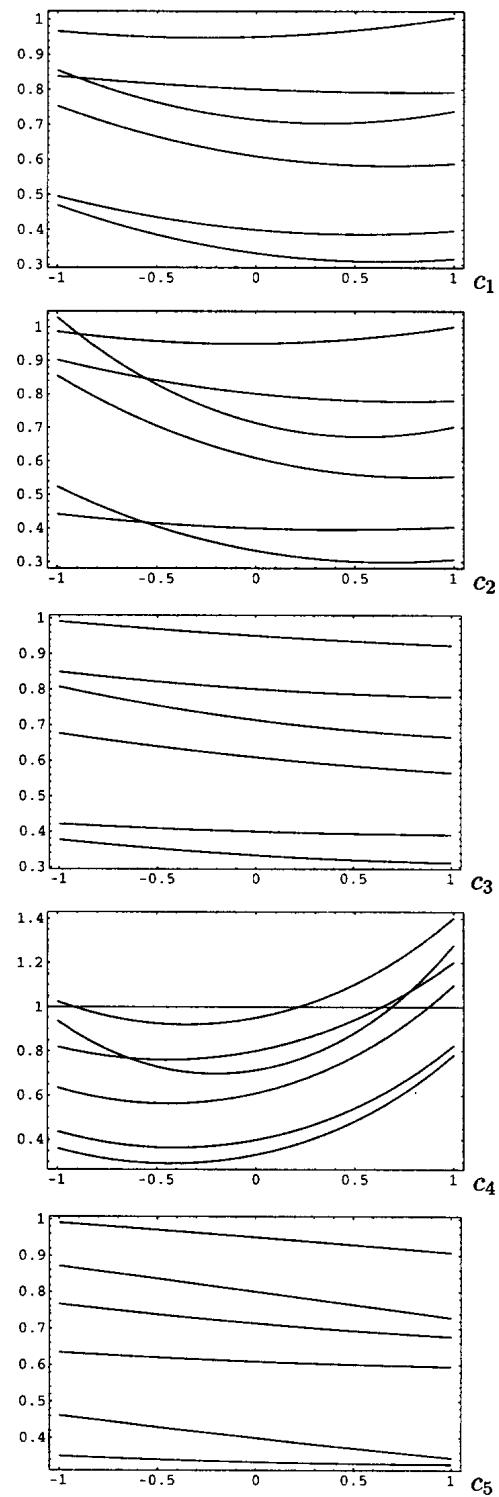


Figure 2.  $F_\xi$  as function of  $\{c_i\}$ .

# A Bird's Eye View of Matrix Distributed Processing

Massimo Di Pierro

School of Computer Science, Telecommunications and Information Systems  
DePaul University, 243 S. Wabash Av., Chicago, IL 60604, USA

**Abstract.** We present Matrix Distributed Processing, a C++ library for fast development of efficient parallel algorithms. MDP is based on MPI and consists of a collection of C++ classes and functions such as lattice, site and field. Once an algorithm is written using these components the algorithm is automatically parallel and no explicit call to communication functions is required. MDP is particularly suitable for implementing parallel solvers for multi-dimensional differential equations and mesh-like problems.

## 1 Introduction

Matrix Distributed Processing (MDP) [1] is a collection of classes and functions written in C++ to be used as components for fast development of efficient parallel algorithms. Typical algorithms include solvers for partial differential equations, mesh-like algorithms and various types of graph-based problems. These algorithms find frequent application in many sectors of physics, engineering, electronics and computational finance.

MDP components can be divided into two main categories:

- Non parallel components: Linear Algebra components (class `mdp_complex`, class `mdp_array`, class `mdp_matrix`) and Statistical Analysis components (class `Measure`, class `Jackboot`)
- Parallel components: (class `mdp_lattice`, class `mdp_site`, class `mdp_field`, etc.)

In this paper we will focus exclusively on the Linear Algebra and the Parallel components<sup>1</sup>.

MDP is based on MPI and can be used on any machine with an ANSI C++ and support for the MPI communication protocol. No specific communication hardware is required but a fast network switch is suggested. MDP has been tested on Linux PC clusters, SUN workstations and a Cray T3E.

The best way to introduce MDP is to write a program that solves a typical problem:

---

<sup>1</sup> The parallel components can interoperate with other third party C/C++ linear algebra packages and can be used to parallelize existing applications with minimal effort.

**Problem:** Let's consider the following differential equation:

$$\nabla^2 \varphi(x) = f(x) \quad (1)$$

where  $\varphi(x)$  is a field of  $2 \times 2$  Complex matrices defined on a 3D space (space),  $x = (x_0, x_1, x_2)$  limited by  $0 \leq x_i < L_i$ , and

$$\begin{aligned} L &= \{10, 10, 10\}, \\ f(x) &= A \sin(2\pi x_1/L_1), \\ A &= \begin{pmatrix} 1 & i \\ 3 & 1 \end{pmatrix} \end{aligned} \quad (2)$$

The initial conditions are  $\varphi_{initial}(x) = 0$ . We will also assume that  $x_i + L_i = x_i$  (torus topology).

**Solution:** In order to solve eq. (1) we first discretize the Laplacian ( $\nabla^2 = \partial_0^2 + \partial_1^2 + \partial_2^2$ ) and rewrite it as

$$\sum_{\mu=0,1,2} [\varphi(x + \hat{\mu}) - 2\varphi(x) + \varphi(x - \hat{\mu})] = f(x) \quad (3)$$

where  $\hat{\mu}$  is a unit vector in the discretized space in direction  $\mu$ . Hence we solve it in  $\varphi(x)$  and obtain the following a recurrence relation

$$\varphi(x) = \frac{\sum_{\mu=0,1,2} [\varphi(x + \hat{\mu}) + \varphi(x - \hat{\mu})] - f(x)}{6} \quad (4)$$

The following is a typical MDP program that solves eq. (1) by recursively iterating eq. (4). The program is parallel but there are no explicit call to communication functions:

```

00  #include "mdp.h"
01
02  void main(int argc, char** argv) {
03      mdp.open_wormholes(argc,argv);      // open communications
04      int L[]={10,10,10};                // declare volume
05      mdp_lattice    space(3,L);        // declare lattice
06      mdp_site       x(space);         // declare site variable
07      mdp_matrix_field phi(space,2,2); // declare field of 2x2
08      mdp_matrix     A(2,2);           // declare matrix A
09      A(0,0)=1;   A(0,1)=I;
10      A(1,0)=3;   A(1,1)=1;
11
12      forallsites(x)                  // loop (in parallel)
13          phi(x)=0;                  // initialize the field
14          phi.update();              // communicate!
15
16      for(int i=0; i<1000; i++) {      // iterate 1000 times

```

```

17      forallsites(x)           // loop (in parallel)
18          phi(x)=(phi(x+0)+phi(x-0)+ 
19                      phi(x+1)+phi(x-1)+ 
20                      phi(x+2)+phi(x-2)- 
21                      A*sin(2.0*Pi*x(1)/L[1]))/6; // equation
22          phi.update();        // communicate!
23      }
24      phi.save("field_phi.mdp"); // save field
25      mdp.close_wormholes();    // close communications
26  }

```

**Notes:**

- Line 00 includes the MDP library.
- Lines 03 and 25 respectively open and close the communication channels over the parallel processes.
- Line 04 declares the size of the box  $L = \{L_0, L_1, L_2\}$
- Line 05 declares a lattice, called **space**, 3-dimensional, on the box  $L$ . MDP supports up to 10-dimensional lattices. By default a lattice object is a mesh with torus topology. It is possible to specify an alternative topology, boundary conditions and any parallel partitioning for the lattice. Notice that each lattice object contains a parallel random generator.
- Line 06 declares a variable site, called **x**, that will be used to loop over lattice sites (in parallel).
- Line 07 declares a field of  $2 \times 2$  matrices, called **phi**, over the lattice **space**. MDP is not limited to fields of matrices. It is easy to declare fields of any user-defined structure or class.
- Lines 08 through 10 define the matrix **A**.
- Lines 12 and 13 initialize the field **phi**. Notice that **phi** is distributed over the parallel processes and **forallsites** is a parallel loop.
- Line 14 performs communications so that each process becomes aware of changes in the field performed by other processes (*synchronization*).
- Lines 16 through 24 perform 1000 iterations to guarantee convergence. In real life applications one may want to implement some convergence criteria as stopping condition.
- Line 17 loops over all sites in parallel.
- Lines 18 through 21 implement eq. (4). Notice the similarity in notation. Here **phi(x)** is a  $2 \times 2$  complex matrix
- Line 22 performs *synchronization*.
- Line 24 saves the field. Notice than any field, including the user defined ones, inherit methods **save** and **load** from a basic class **mdp\_field**.
- It should also be noted that all MDP classes and functions are both type and exception safe. Moreover MDP components can be used without knowledge of C pointers and pointer arithmetics.

## 2 Linear Algebra

MDP includes a Linear Algebra package. The basic classes are:

- class `mdp_real`, that should be used in place of float or double.
- class `mdp_complex`, (just another implementation of complex numbers).
- class `mdp_array`, for vectors and/or multidimensional tensors.
- class `mdp_matrix`, for any kind of complex rectangular matrix.

The most notable difference between our linear algebra package and other existing packages is its natural syntax.

For example:

```
mdp_matrix A,B;
A=Random.SU(3);
B=exp(inv(A))*hermitian(A+5);
```

reads like

$$\begin{aligned} A \text{ and } B &\text{ are matrices} \\ A &\text{ is a random } SU(3) \text{ matrix} \\ B &= e^{(A^{-1})}(A + 5 \cdot \mathbf{1})^H \end{aligned} \tag{5}$$

Notice that each matrix can be resized at will and is resized automatically when a value is assigned.

## 3 Lattice, Site, and Field

An `mdp_lattice` is a container for *topology* and *partitioning* information about the sites. In more abstract terms a lattice is any collection points (vertices) embedded in a multi-dimensional space and connected with directional links. The set of links determines the lattice topology and the boundary conditions. The term partitioning refers to the function that assigns each site (vertex) to one of the parallel process. A lattice, by default, is a mesh.

Each lattice is partitioned over the parallel processes at runtime. There is a default topology and default partitioning but it is possible pass any topology and partitioning functions to the `mdp_lattice` constructor.

A lattice also contains a parallel random number generator: each site of each lattice has its own independent random number generator.

On each lattice it is possible to allocate one or more fields. Some fields are built-in, for example: `mdp_complex_field`, `mdp_vector_field`, `mdp_matrix_field`, etc. All of them extend (inherit from) `mdp_field<mdp_complex>`.

Class `mdp_complex` can be used to declare any type of field. For example:

```
class W {
public: int w[10];
};
int L[]={30,30};
mdp_lattice plane(2,L);
mdp_field<W> psi(plane);
```

declares a  $30 \times 30$  lattice (`plane`) and a field (`psi`), that lives on the `plane`. The field variables of `psi`, `psi(x)` assuming `x` is an `mdp_site` of `plane`, belong to class `W`.

Each user-defined field can be saved:

```
psi.save("filename");
```

loaded

```
psi.load("filename");
```

and synchronized

```
psi.update();
```

as any of the built-in fields.

## 4 Optimization Issues

Once an `mdp_lattice` object is declared the constructor of class `mdp_lattice` performs the following operations:

- Declares a parallel random number generator associated to each site (it uses the Marsaglia random number generator).
- Builds tables containing topology and partitioning information that will be used by the fields to optimize (minimize) communication. Basically each site determines which other sites are its neighbors and where they are located (on which parallel process). When a new field is created on the lattice the field will use these tables to create buffers for the communications.

Once a field object is declared the constructor of class `mdp_field` (or derived field) performs the following operations:

- Each process allocates memory to store the local sites (i.e. sites that will be managed by the process itself).
- Each process loops over every other process and determines if the other process allocated sites that are neighbors of the local sites (this information is already stored in the tables maintained by the lattice object). If this occurs the two processes are said to *overlap*: they have sites in common that need to be synchronized.
- Each process allocates buffers to store copies of the sites that are not local but are neighbors of the local ones and need to be synchronized with the overlapping processes (in this paper we are assuming only next-neighbor synchronization but actually MDP supports also extended synchronization such as next-to-next-neighbor and more). Buffers are created according with some conditions: sites synchronized with the same overlapping process are stored contiguously in memory so that communication can be performed in a single *send/receive*. These buffers are created independently by each field.

Notice that two processes may be overlapping in respect to a given lattice and not overlapping in respect to a different lattice in the same program.

Every time a field changes, for example in a parallel loop such as

```
forallsites(x) phi(x)=0;
```

the program notifies the field that its values have been changed by calling

```
phi.update();
```

The method `update` performs all required communication to copy site variables that need to be synchronized between each couple of overlapping processes. These communications are optimal in the sense that:

- Each process, at each one time, is involved only in one send and one receive.
- Two different processes communicate only if they are overlapping in respect of the lattice associated to the field.
- If two different processes are overlapping, they perform a single send/receive of all sites variables that are synchronized between the two.
- Only the sending process needs to create a temporary buffer. The receiving process receives the site variables in the same buffer where they are normally stored without reordering (and without need for a temporary buffer).

We will refer to our set of communication rules as a “communication policy” (it is possible, in principle, to change this policy to deal with non-standard network solutions). Although our communication policy does not overlap communication with computation it has the advantage of minimizing network jam and calls to send/receive. Hence this communication policy is almost insensitive to network latency and is dominated by network bandwidth. Benchmarks are application dependent since parallel efficiency is greatly affected by the lattice size, by the amount of computation performed per site, processor speed and type of interconnection. In many typical applications, like the one described in the preceding example, the drop in efficiency is less than 10% up to 8 nodes (processes) and less than 20% up to 32 (our tests are usually performed on a cluster of Pentium 4 PCs (2.2GHz) running Linux and connected by Myrinet).

## 5 Conclusions

MDP is a powerful and reliable tool for developing efficient parallel numerical applications. Even if, on the one side, MDP is still undergoing development, on the other side, all of the features here described are fully functional and have been tested in real-life applications. For example MDP constitutes the core of the FermiQCD project [2] developed by the University of Southampton (UK) and Fermilab (Department of Energy). FermiQCD is collection of parallel algorithms for Quantum Chromo Dynamics computations. The typical FermiQCD problem is equivalent to solving iteratively a system of stochastic differential equations

in a 4-dimensional space. Typical field variables are vectors of complex matrices. **FermiQCD** programs are used in production runs in parallel on 8 or more nodes.

We believe MDP could be a useful tool for scientists developing parallel numerical applications. MDP version 2.0 (current) is open source and is free for research and educational purposes.

#### Project web pages:

- <http://www.pheonixcollective.org/mdp/mdp.html> (license and source code)
- <http://www.fermiqcd.net> (Lattice QCD applications)

## References

1. M. Di Pierro, “Matrix Distributed Processing: ...”, Computer Physics Communications, **141** (2001), pp. 98-148 [<http://xxx.lanl.gov/abs/hep-lat/0004007>]. *Note: this paper describes version 1.3 of MDP. The current version is 2.0*
2. M. Di Pierro, “**FermiQCD**”, Nucl. Phys. Proc. Suppl. **106** (2002) 1034-1036 [<http://xxx.lanl.gov/abs/hep-lat/0110116>]



ELSEVIER

Nuclear Physics B (Proc. Suppl.) 119 (2003) 586–591

**NUCLEAR PHYSICS B  
PROCEEDINGS  
SUPPLEMENTS**

www.elsevier.com/locate/npc

## Charmonium with three flavors of dynamical quarks\*

Massimo di Pierro<sup>a</sup>, Aida X. El-Khadra<sup>ab</sup>, Steven Gottlieb<sup>ac</sup>, Andreas S. Kronfeld<sup>a</sup>, Paul B. Mackenzie<sup>a</sup>, Damian P. Menscher<sup>b</sup>, Mehmet B. Oktay<sup>b</sup>, and James N. Simone<sup>a</sup>

<sup>a</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

<sup>b</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>c</sup> Department of Physics, Indiana University, Bloomington, IN 47405

We present a calculation of the charmonium spectrum with three flavors of dynamical staggered quarks from gauge configurations that were generated by the MILC collaboration. We use the Fermilab action for the valence charm quarks. Our calculation of the spin-averaged 1P–1S and 2S–1S splittings yields a determination of the strong coupling, with  $\alpha_{\overline{\text{MS}}}(M_Z) = 0.119(4)$ .

### 1. INTRODUCTION

The current experimental program of precision flavor physics at the  $B$  factories, at CESR-c, and at the Tevatron needs accurate lattice QCD calculations of the relevant hadronic matrix elements to yield stringent constraints on the CKM sector of the standard model. Precision lattice QCD results in turn require that the systematic errors associated with lattice calculations be brought under control to the desired accuracy. The most important sources of systematic error in lattice calculations include the incomplete inclusion of sea quarks (quenched approximation), lattice spacing artifacts, perturbative errors, and the chiral extrapolation. We are planning a series of lattice calculations of the phenomenologically most important quantities in the  $B$ ,  $D$ ,  $\bar{c}c$  and  $\bar{b}b$  systems. We will address the first two sources of systematic error by performing simulations with highly improved actions on gauge configurations with  $n_f = 2 + 1$  improved staggered quarks [1]. This effort must be complemented by the corresponding perturbative matching calculations of the improvement coefficients and current renormalizations. This should be possible with recent advances in automated perturbation theory [2].

The MILC collaboration has generated dynamical gauge configurations [3] using improved staggered and gluon actions. Their configurations include three (or  $2 + 1$ ) flavors of light staggered fermions at several different light quark masses ranging from  $m_s$  to  $m_s/5$ . Hence, the systematic errors usually associated with the quenched approximation should be absent with these configurations. Furthermore, since numerical simulations with rather light quark masses are feasible with staggered actions, the issue of chiral extrapolations may be carefully studied.

The heavy quark action used in this work is based on Ref. [4]. It is related to NRQCD, but uses the four component fields and operators of the Wilson action rather than the two component fields and operators of NRQCD. Similar to NRQCD, the space-like and time-like components of the operators are uncoupled, and the coefficients of the operators are mass dependent. This action smoothly interpolates between an ordinary light quark action as  $am \rightarrow 0$  and NRQCD when  $am > 1$ , but is applicable at all values of  $am$ . Our formalism can be regarded as a summation of terms of the form  $(am)^p$  to all orders in the normalizations of operators, which is useful when  $m \gg \Lambda_{\text{QCD}}$ . To  $O(a)$ , our action uses the same operators as the clover action [5]. Starting at  $O(a^2)$ , the operators are somewhat differ-

\*Talk and poster presented by P. Mackenzie and D. Menscher.

ent from those in the analogous light quark action. The reason is that a two-hop correction to the Wilson time derivative operator cannot be used because it introduces ghost states for heavy quarks. Its effects must instead be duplicated with Hamiltonian-style operators [6]. The action may be particularly useful for the charm quark on lattice spacings with  $am_c < 1$ . It has smaller discretization errors than standard light quark actions since the  $(am_c)^p$  errors, which it resums and eliminates, are much larger than the remaining  $(a\Lambda_{\text{QCD}})^p$  discretization errors. Since it has a well-defined  $a \rightarrow 0$  limit, it may be more convergent for  $am < 1$  than NRQCD, which does not.

Now that we are calculating with dynamical quarks, lattice spacings obtained from the simplest quantities should all agree within errors. We start our work with a study of the charmonium spectrum, to test our methods and lattices. As a byproduct, we obtain a new determination of the strong coupling.

## 2. DETAILS OF THE CALCULATION

MILC uses the Asqtad action [3] for the staggered fermions which contains errors of  $O(\alpha_s a^2)$ , and an improved gluon action with  $O(\alpha_s^2 a^2)$  errors. For the charm quarks we use the Sheikholeslami-Wohlert action [5] (which has  $O(\alpha_s a)$  errors) with the Fermilab interpretation [4]. The quenched gauge configurations are generated with the Wilson gauge action. Table 1 lists the simulation parameters for all three lattices. The quenched lattice has a slightly smaller lattice spacing than the MILC lattices.

As usual, we calculate charmonium two-point functions using smeared source and sink operators. For this purpose, when working on MILC lattices, we use the Richardson potential [7] model wave functions shown in Figure 1. The quenched propagators were generated using Coulomb wave functions.

### 2.1. Fits

As shown in Refs. [8,9], constrained fits allow for better control of the systematic error due to excited state contributions, because they allow us

Table 1  
Simulation parameters for the three lattices.

Size	$16^3 \times 32$	$20^3 \times 64$	$20^3 \times 64$
$n_f$	0	3	2 + 1
$\beta$	5.9	6.85	6.76
configs	300	174	298
$am_s$	$\infty$	0.05	0.05
$am_l$	$\infty$	0.05	0.01
$\kappa_{\text{ch}}$	0.1227	0.113	0.113
wf's	$\delta, 1S, 2S$	$\delta, 1S$	$\delta, 1S$

to fit the correlators to a large number of states without loss of accuracy. Furthermore, with constrained fits one is also able to use all of the time slices in the fits without adjusting  $t_{\min}$  and  $t_{\max}$  to determine the best fit. We investigate this issue by comparing constrained and unconstrained fits. We fit meson correlators to the form

$$G(t; \{Z_n\}, \{E_n\}) = \sum_n Z_n^2 (e^{-E_n t} + e^{-E_n(T-t)}). \quad (1)$$

We force our energy levels to be ordered by defining

$$\Delta E_n = E_n - E_{n-1} \equiv \exp(\epsilon_n). \quad (2)$$

In our fits we use Bayesian statistics,

$$\chi^2 \rightarrow \chi^2_{\text{aug}} \equiv \chi^2 + \chi^2_{\text{prior}}, \quad (3)$$

where  $\chi^2_{\text{prior}}$  is used to constrain  $E_0$ ,  $\epsilon_n$ , and  $Z_n$  to a predetermined range.

For simplicity, we test our fit program on the quenched lattice. Figures 2–3 show our fit results for the  $\eta_c$  ground and first excited state energies as functions of the number of states in the fit ( $n$ ) obtained from the  $\eta_c$  propagator with a  $\delta$ -function source and sink. We observe that the fit results — particularly the error bars — for the ground and first excited state energies are stable under adding more states to the fit when constrained fits are used, but not in the case of unconstrained fits. The unconstrained fits do not use Eq. 3, but they do use the energy ordering constraint of Eq. 2, which is probably the reason for the relative stability of the central fit values even in the unconstrained case.

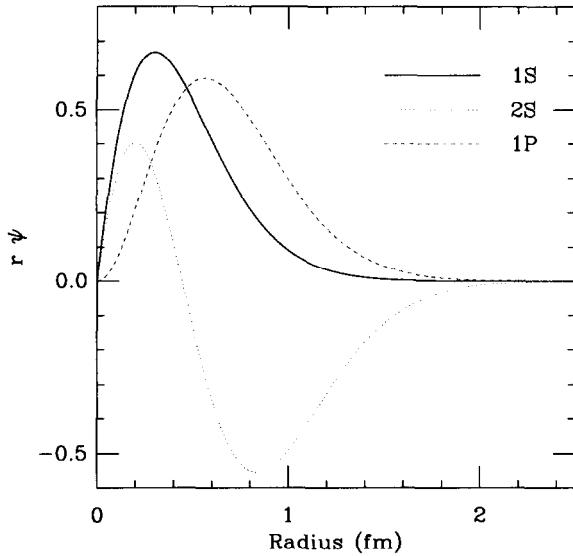


Figure 1. Wavefunctions for the  $1S$ ,  $2S$ , and  $1P$  states in charmonium.

The prior constraints must be chosen so that they do not unduly influence the physical results. Figure 4 shows the dependence of the ground state fit on the prior width. The fit includes four states and the ranges for all priors are varied together with the ground state energy prior width. We see that the fit results are stable, once the prior width is large enough. Furthermore, the error on the ground state energy is unaffected by the variation of the prior width, after the plateau is reached. Table 2 lists typical choices for the energy prior values and ranges used in our fits. The prior values for the  $Z_n$  are chosen by match-

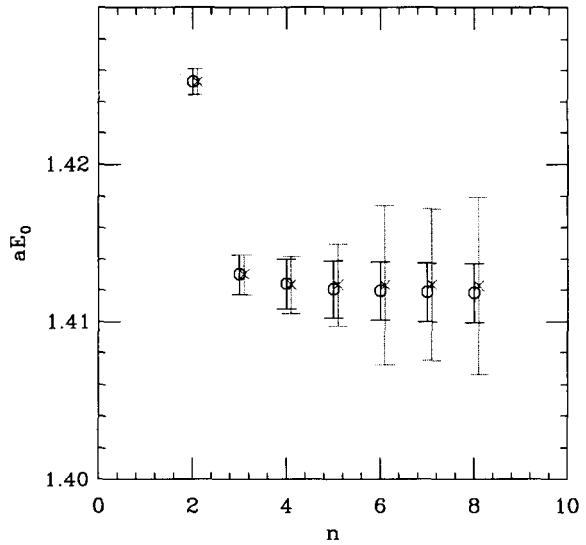


Figure 2. Comparison of fit results from constrained and unconstrained fits. Shown is the  $\eta_c$  ground state energy—obtained from the local-local correlator as a function of the number of states in the fit ( $n$ ). ○: constrained fit, ×: unconstrained fit

ing the hadron propagators evaluated at  $t = 1$  to Eq. 1; the range is usually set to a factor of three of the central value.

The results discussed in the next section are obtained from fits to multiple correlators, making use of the different source and sink operators listed in Table 1. These fits are generally consistent with fits to the delta-function correlators, albeit with smaller statistical errors.

### 3. THE SPECTRUM

Figure 5 summarizes our results for the charmonium spectrum. Our results for the hyperfine,  $1P-1S$ , and  $2S-1S$  splittings are shown in Figures 6–8 as functions of the light quark mass. We observe very little light quark mass dependence in the hyperfine splitting, and our chirally extrapolated result still disagrees with experiment, although the inclusion of dynamical quarks has removed one possible cause of a small hyperfine

Table 2  
Typical energy prior values and ranges.

$n_f$	0	3	$2 + 1$
$\eta_c$ : $E_0$	$1.4 \pm 0.2$	$1.9 \pm 0.2$	$1.9 \pm 0.2$
$\Delta E_n$	$0.4 \pm 0.2$	$0.4 \pm 0.2$	$0.4 \pm 0.2$
$h_c$ : $E_0$	$1.7 \pm 0.2$	$2.3 \pm 0.2$	$2.3 \pm 0.2$
$\Delta E_n$	$0.4 \pm 0.2$	$0.5 \pm 0.2$	$0.4 \pm 0.2$

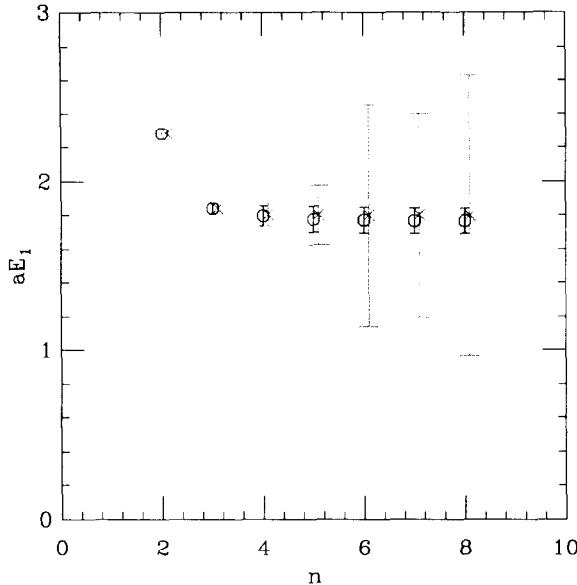


Figure 3. Same as Figure 2, but for the first excited state, the  $\eta'_c$ .

splitting: the fact that the short distance coupling constant is too small in the quenched approximation. We note that the charm quark action is only  $O(a)$  improved. The leading order operator which controls the spin splitting is  $\bar{\psi}\sigma_{\mu\nu}F_{\mu\nu}\psi$ . Its coefficient is being included only at tadpole improved tree-level, and it is plausible that the observed discrepancy is a result of both  $O(\alpha_s a)$  errors and  $O(a^2)$  lattice spacing artifacts. We will be able to study this issue further once our  $O(a^2)$  improved action [6] is ready for numerical simulations.

The spin-averaged 1P-1S and 2S-1S splittings are considerably less sensitive to the leading order lattice artifacts; they are used to determine the lattice spacing. Figures 7–8 indicate that the dependence of the spin-averaged splittings on the light quark mass is mild.

The extent to which the lattice spacings from the 1P-1S and 2S-1S splittings disagree with each other is an indication of residual systematic errors in our simulation, which are a combination of higher order lattice spacing and sea quark effects. For this comparison we compute the ratio

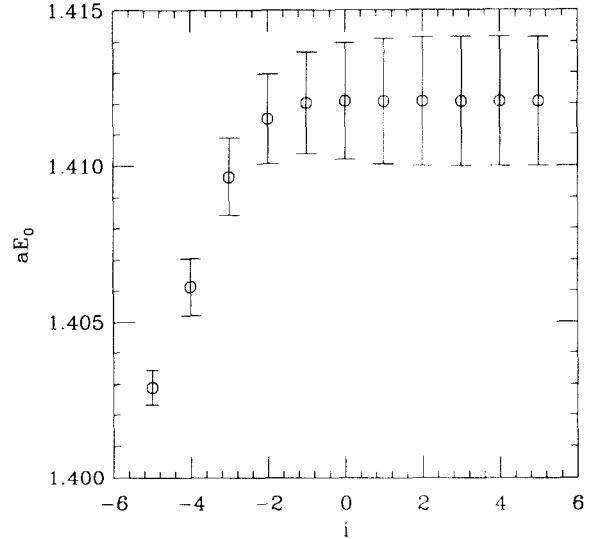


Figure 4. Variation of the  $\eta_c$  ground state energy fit result with the prior width.  $i$  is defined via  $\sigma = 2^i \sigma_0$ , where  $\sigma$  and  $\sigma_0$  are the varied and standard choice of prior widths respectively.

of lattice spacings,

$$R \equiv \frac{\frac{\Delta M(2S-1S)}{\Delta M(1P-1S)}^{\text{lat}}}{\frac{\Delta M(2S-1S)}{\Delta M(1P-1S)}^{\text{exp}}} \quad (4)$$

Table 3 compares the lattice spacings obtained on all three lattices. On the quenched lattice the deviation of  $R$  from unity is about 1.5 standard deviations. On the two MILC lattices, the deviation of  $R$  from unity is less significant, because of the still somewhat large statistical errors. To

Table 3

The lattice spacings from the spin-averaged 1P-1S and 2S-1S splittings for the three lattices with statistical error bars.

$n_f$	$a^{-1}(1P-1S)$ (GeV)	$a^{-1}(2S-1S)$ (GeV)	$R$
0	$1.783^{+0.062}_{-0.030}$	$1.637^{+0.079}_{-0.059}$	$1.089^{+0.057}_{-0.052}$
3	$1.70^{+0.16}_{-0.18}$	$1.52^{+0.15}_{-0.11}$	$1.12^{+0.14}_{-0.17}$
2 + 1	$1.564^{+0.058}_{-0.053}$	$1.426^{+0.073}_{-0.070}$	$1.097^{+0.073}_{-0.068}$

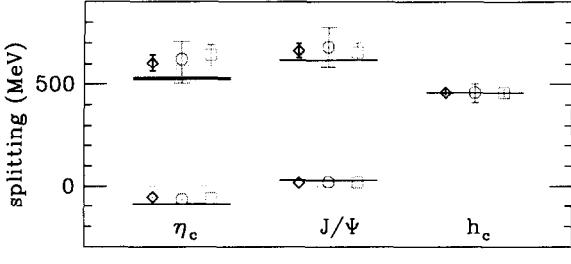


Figure 5. The charmonium spectrum in comparison.  $\diamond$ :  $n_f = 0$ ,  $\circ$ :  $n_f = 3$ ,  $\square$ :  $n_f = 2 + 1$ .

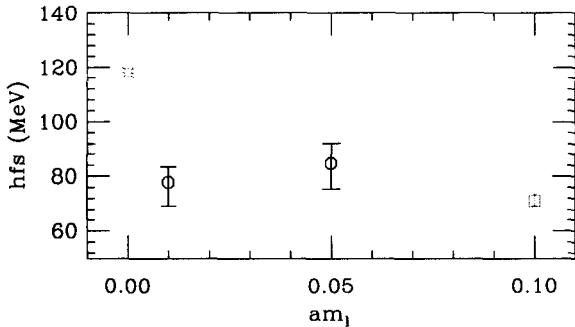


Figure 6. The hyperfine splitting  $vs.$   $am_l$  (circles). Shown also is the quenched result (square), positioned in the plot at finite  $am_l$  for illustration, and the experimental result (burst) positioned at  $am_l = 0$ .

clarify the situation we need to reduce the statistical errors of the results on the MILC lattices.

Now that dynamical fermions are included in the calculations, lattice spacings from the best-understood quantities should be consistent. We note that although the lattice spacings obtained from the 1P–1S and the 2S–1S splittings are consistent with each other, the lattice spacing from the 1P–1S splitting is consistent with several determinations of the lattice spacing from the  $\Upsilon$  system [10], while the one obtained from the 2S–1S splitting is not. It is possible that higher order discretization effects are responsible for this.

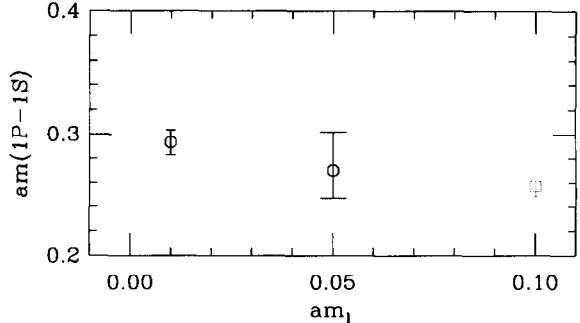


Figure 7. The 1P–1S splitting  $vs.$   $am_l$  (circles). Shown also is the quenched result (square), positioned in the plot at finite  $am_l$  for illustration.

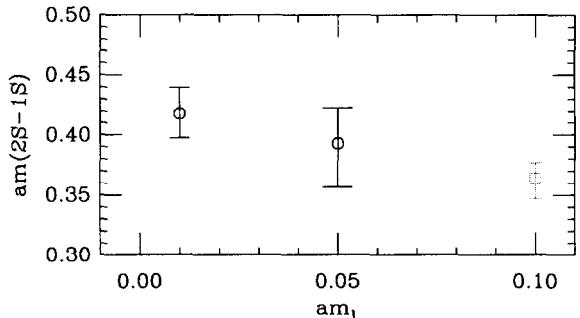


Figure 8. The 2S–1S splitting  $vs.$   $am_l$  (circles). Shown also is the quenched result (square), positioned in the plot at finite  $am_l$  for illustration.

A more interesting possibility is that due to the fact that the 2S is so close to the  $D\bar{D}$  threshold, physical effects from the coupling to  $D\bar{D}$  channels may have a more dramatic effect in the 2S than in other states.

#### 4. THE STRONG COUPLING

The spin-averaged splittings discussed in the previous section are used to determine the strong coupling  $\alpha_s$ . The 2 + 1 flavor lattices are our most realistic. We take the 1P–1S splitting as our

most reliable determination of the lattice spacing because of the possible threshold effects in the 2S state. Following the procedure of Ref. [11], we obtain the strong coupling from the plaquette. For our actions we have [12]

$$-\ln\langle\text{Tr } U_P\rangle = \quad (5) \\ 3.0682 \alpha_P(q^*) [1 - \alpha_P(0.770 + 0.09681 n_f)]$$

where  $q^* = 3.33/a$  is the BLM scale for  $n_f = 3$ . The coupling  $\alpha_P$  is defined to coincide through one-loop order with  $\alpha_V$ , the coupling defined from the heavy quark potential. Using

$$\alpha_{\overline{\text{MS}}}(q) = \alpha_P \left( e^{5/6} q \right) \left[ 1 + \frac{2}{\pi} \alpha_P + O(\alpha_P^2) \right], \quad (6)$$

we obtain

$$\alpha_{\overline{\text{MS}}}(M_Z) = 0.119 \pm 0.004. \quad (7)$$

The difference between our value and the value reported in Ref. [12] arises mainly from differing implicit treatment of  $O(\alpha^3)$  corrections, which will soon be known. The main sources of uncertainty are  $O(\alpha^3)$  corrections (3%), discretization errors (2%), and statistical errors (1%). All three should be significantly reduced soon.

## 5. CONCLUSIONS AND OUTLOOK

We present preliminary results of a calculation of the charmonium spectrum on gauge configurations generated by the MILC collaboration using  $O(a^2)$  improved actions for the gluons and the  $n_f = 2+1$  dynamical staggered fermions. We use the  $O(a)$  improved clover action with the Fermilab interpretation for the charm valence quarks. Since the MILC configurations were generated with the correct number of sea quarks, no extrapolation in  $n_f$  is necessary. Furthermore, the nondegenerate strange and light quark masses in the MILC lattices allow us to consider the chiral limit. Comparing results at  $am_s = am_l = 0.05$  and  $am_s = 0.05, am_l = 0.01$ , we find only mild light quark mass dependence for all spectral quantities we consider. Finally, our calculation yields a new determination of the strong coupling where systematic errors due to sea quark effects are under control.

For future work, we are planning to improve the statistical accuracy of this work. The improvement of the heavy quark action beyond  $O(a)$  is in progress [6].

We thank the MILC collaboration for the use of their configurations. We thank Peter Lepage for helpful conversations. This work was supported in part by the Department of Energy. We thank the Fermilab Computing Division and the Sci-DAC program for their support. Fermilab is operated by Universities Research Association Inc., under contract with the DOE.

## REFERENCES

1. T. Blum et al., Phys. Rev. D55 (1997) R1133; J. Lagae and D. Sinclair, Phys. Rev. D59, (1998) 104511; G. P. Lepage, Phys. Rev. D59 (1999) 074502; K. Orginos, D. Toussaint, and R. Sugar, Phys. Rev. D60 (1999) 054503; C. Bernard et al., Phys. Rev. D61 (2000) 111502(R).
2. M. Nobes and H. Trottier, these proceedings, hep-lat/0209017.
3. C. Bernard et al.(MILC collaboration), Phys. Rev. D64 (2001) 054506.
4. A. El-Khadra, A. Kronfeld and P. Mackenzie, Phys. Rev. D55 (1997) 3933.
5. B. Sheikholeslami and R. Wohlert, Nucl. Phys. B259 (1985) 572.
6. M. B. Oktay et al., these proceedings, hep-lat/0209150.
7. J.L. Richardson, Phys. Lett. B82 (1979) 272.
8. G.P. Lepage et al., Nucl. Phys. Proc. Suppl. 106 (2002) 12.
9. C. Morningstar, Nucl. Phys. Proc. Suppl. 109 (2002) 185.
10. A. Gray et al., these proceedings, hep-lat/0209022.
11. P. Lepage and P. Mackenzie, Phys. Rev. D48 (1993) 2250.
12. C. Davies et al., these proceedings, hep-lat/0209122.

## The Second Moment of the Pion Light Cone Wave Function

Luigi Del Debbio<sup>a</sup>, Massimo Di Pierro<sup>b</sup> \* and Alex Dougall<sup>c</sup>

<sup>a</sup>Dipartimento di Fisica, Universitá di Pisa, P.zza Torricelli 2, Pisa, Italy

<sup>b</sup>Fermilab, PO Box 500, Batavia, IL 60563, USA

<sup>c</sup>Theoretical Physics, Dept. of Mathematical Sciences, Univ. of Liverpool, Liverpool, L69 3BX, UK

We present a preliminary result for second moment of the light cone wave function of the pion. This parameter is the subject of a discrepancy between theoretical predictions (coming from lattice and sum rules) and a recent experimental result (that remarkably agrees with purely perturbative predictions). In this work we exploit lattice hypercubic symmetries to remove power divergences and, moreover, implement a full 1-loop matching for all the contributing operators.

### 1. INTRODUCTION

The light cone wave function of the pion,  $\phi(x, Q^2)$  is the probability amplitude of finding a parton of a pion (moving with momentum  $p$  in a light cone frame) with parallel momentum equal to  $xp$  and transverse momentum less than  $Q$ . This wave function incorporates non-perturbative physics and plays an important role in exclusive hard scattering processes and in non-leptonic decays of heavy mesons. One example of application is the electromagnetic form factor of the pion,  $F(q^2)$ , defined by

$$\langle \pi(p') | \bar{q} \gamma_\mu q | \pi(p) \rangle = F((p - p')^2)(p + p')_\mu \quad (1)$$

This form factor can, in fact, be written in terms of  $\phi$  and  $T_H$  (the perturbative scattering amplitude for the constituents) as

$$F(Q^2) = \int \phi^\dagger(x, Q^2) T_H(x, y, Q^2) \phi(y, Q^2) dx dy \quad (2)$$

More formally the light cone wave function can be defined as [1] [2]

$$\phi_{\alpha\beta}^{ab}(x, Q^2) = F.T. \langle 0 | T\{q_\alpha^a(z_1), \bar{q}_\beta^b(z_2)\} | \pi \rangle \quad (3)$$

where F.T. indicates a Fourier transform on  $z_1$  and  $z_2$  assuming

\*Talk presented by Massimo Di Pierro

- the sum of the parallel components of the momenta of the two partons is equal to the total pion momentum  $p$ .
- the transverse components have been integrated out up to momentum  $Q$ .

The  $n$ -th moment of  $\phi$  is defined as

$$\langle \xi^n \rangle = \int_0^1 \xi^n \phi(\xi, Q^2) d\xi \quad (4)$$

In a typical lattice determination of such quantity the cut-off is provided by the lattice spacing,  $Q \simeq a^{-1}$ .

We finally wish to remark that  $\langle \xi^0 \rangle = 1$  is fixed by a normalization condition and  $\langle \xi^1 \rangle = 0$  because the wavefunction is symmetric under G-parity. Therefore  $\langle \xi^2 \rangle$  is the first non trivial moment of  $\phi$ .

This second moment can be related to

$$\langle 0 | O_{\mu\nu\rho} | \pi(p) \rangle = f_\pi \langle \xi^2 \rangle p_\mu p_\nu p_\rho + \dots \quad (5)$$

where the ellipsis indicates divergent terms,

$$O_{\mu\nu\rho} = \bar{q} \gamma_\mu \gamma_5 \overset{\leftrightarrow}{D}_\nu \overset{\leftrightarrow}{D}_\rho q' \quad (6)$$

and

$$q \overset{\leftrightarrow}{D} q' \stackrel{def}{=} q \overset{\rightarrow}{D} q' + q \overset{\leftarrow}{D} q' \quad (7)$$

## 2. COMPUTATION

Our computation was carried on 154 quenched gauge configurations generated by the UKQCD collaboration using the Wilson gauge action. We use the Clover  $O(a^2)$  improved action for the light quarks.

The parameters of our computation are:

- Volume equal to  $24^3 \times 48$ .
- $\beta = 6.2$  which corresponds to an inverse lattice spacing of  $a^{-1} = 2.67 \pm 0.10 \text{ GeV}$ .
- $\kappa$  values 0.13460, 0.13510, 0.13530 (corresponding to pseudoscalar masses of 748, 574 and 490 MeV respectively)
- $\kappa_{crit} = 0.13582$  and  $c_{SW} = 1.61$

For technical reasons we choose not to improve the operators and do not smear the light quarks. The latter choice is motivated by the fact that the local axial current seems to have better superposition with the pion than other smeared operators we tried.

In order to calculate any moment of the pion light-cone wavefunction, we are required to study the matrix elements of lowest twist local operators between pion and vacuum. In a continuum world these operators are classified by their representation under the group of Lorentz, parity and charge conjugation. In the lattice-discretized world the Lorentz group is broken to  $\mathcal{H}_4 \in O(4)$ , the hypercubic group. Hence lowest twist local operators can mix with higher dimensional operators and introduce power divergences.

We choose a particular combination of the lattice operators that:

- transforms under an irreducible representation of the hypercubic group
- does not mix with higher dimensional operators.

In particular we choose  $O_{\mu[\nu,\rho]}$  with  $\mu \neq \nu \neq \rho \neq \mu$ , which transforms as  $(\frac{1}{2}, \frac{1}{2}) \otimes \mathbf{8}^+$  under  $\mathcal{H}_4$ . The  $\mathbf{8}^+$  irreducible representation would naively generate a term proportional to

$$p_\mu \left[ \frac{p_\nu^2 + p_\rho^2}{2} - (\epsilon^{\mu\nu\rho\sigma} p_\sigma)^2 \right] \quad (8)$$

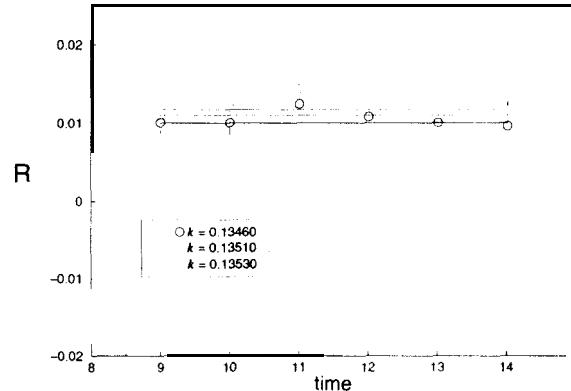


Figure 1. The Plot shows  $R$  as function of  $t$ , the timeslice, and  $\kappa$ . The best fits are included (for each value of  $\kappa$ ).

This contribution vanishes for our matrix elements because of its parity. We introduce the following definition:

$$R = \frac{C_2^O(t, \mathbf{p})}{p_1 p_2 C_2^A(t, \mathbf{P})}_{\mathbf{p}=(1,1,0)} \quad (9)$$

where  $C_2^Q(t, \mathbf{p})$  is the spatial Fourier transform (at momentum  $\mathbf{p}$ ) of  $\langle Q(x) \bar{q} \gamma_5 \psi'(0) \rangle$ ,  $0$  is a short hand notation for  $O_{\mu[\nu,\rho]}$  and  $A = \bar{q} \gamma_5 q'$  is the usual axial current. For large  $t = x_0 \rightarrow \infty$ ,  $C_2(t)$  asymptotically approaches

$$C_2 \simeq \frac{Z_A}{2E(\mathbf{p})} \langle 0 | Q(0) | \pi(\mathbf{p}) \rangle e^{-E(\mathbf{p})t} \quad (10)$$

Using eq. (5), eq. (9) reduces to

$$R \simeq \langle \xi^2 \rangle^{lattice} - \frac{Z^A}{Z^O} \overline{\langle \xi^2 \rangle^{\text{MS}}} \quad (11)$$

the desired second moment (without divergences) times a corrective matching factor to be determined perturbatively.

## 3. MATCHING

A big part of this work was the determination of the matching factor  $Z^O$  for  $0 = O_{\mu\nu\rho}$ . The computation was performed assuming a gluon

mass as regulator and on-shell external quarks with non-zero momentum. All 41 relevant diagrams were expanded in Taylor up to second order in  $p$ , compatibly with eq. (9).

As result of our computation we find that

$$\frac{Z^O}{Z^A} = 1.518 \quad (12)$$

and the coefficients are evaluated at  $\alpha_s(q^*)$  for  $q^* = 2/a$ . We tried varying  $q^*$  by factors of two and this gives us an estimate of the error in the matching of the order of 10% that we include in our final result.

#### 4. RESULTS

The result of our calculation is

$$\langle \xi^2 \rangle_{Q=2.67\text{GeV}}^{\text{lattice}} = 0.185 \pm 0.032 \quad (13)$$

$$\langle \xi^2 \rangle_{Q=2.67\text{GeV}}^{\overline{\text{MS}}} = 0.280 \pm 0.049^{+0.030}_{-0.013} \quad (14)$$

(the first error is statistical and the second is systematic due to matching, quenching error is not included). These numbers should be compared with results from sum rules

$$\langle \xi^2 \rangle_{Q=5\text{GeV}}^{\overline{\text{MS}}} = 0.40 \pm 0.05 \quad (15)$$

with preceding independent lattice results

$$\langle \xi^2 \rangle_{Q\simeq 1\text{GeV}}^{\text{lattice}} = 1.37 \pm 0.20 [3] \quad (16)$$

$$\langle \xi^2 \rangle_{Q\simeq 1\text{GeV}}^{\text{lattice}} = 0.25 \pm 0.10 [4] \quad (17)$$

$$\langle \xi^2 \rangle_{Q=2.4\text{GeV}}^{\text{lattice}} = 0.10 \pm 0.12 [5] \quad (18)$$

and the exact asymptotic value

$$\langle \xi^2 \rangle_{Q=\infty} = 0.2 \quad (19)$$

(confirmed by the Fermilab experiment E791, performed at  $Q \simeq 3 - 4\text{GeV}$ ).

We find that our value for the second moment is smaller than sum rule predictions and is closer to the asymptotic value.

On the one side we conclude that the present computation has better control of statistical and perturbative errors than previous computations. On the other side we strongly feel that dynamical quarks may be playing an important role in this process and quenching errors are yet to be estimated and removed. We also believe that the

same analysis should be repeated for different values of the lattice spacing in order to study the  $Q^2$  dependence of  $\langle \xi^2 \rangle$ .

#### ACKNOWLEDGMENTS

We all wish to thank UKQCD for making the gauge configurations available to us and for giving us access to the Cray T3E (where part of the computation was performed). We also wish to thank Chris Sachrajda for his invaluable contribution and the University of Southampton where this research started. Those of us who no longer work in Southampton thanks their present host institutions for support and funding in continuing this research.

#### REFERENCES

1. V. L. Cherniak and L. R. Zhitnitsky, Phys. Rep. **112** (1984) 173
2. S. Brodsky et al., Phys. Lett. **B 91** (1980) 239
3. S. Gottlieb and A. S. Kronfeld, Phys. Rev. **D 33** (1986) 227
4. G. Martinelli and C. Sachrajda, Phys. Lett. **B 190 (1987) 151**
5. T. Daniel, R. Gupta and D. Richards, Phys. Rev. **D 43 (1991) 3715**
6. L. Del Debbio et al., Nucl. Phys. **B** (proc. suppl.) 83-84 (2000) 235

## $D_s$ spectrum and leptonic decays with Fermilab heavy quarks and improved staggered light quarks

Massimo di Pierro<sup>a</sup>, Aida X. El-Khadra<sup>b</sup>, Steven Gottlieb<sup>c</sup>, Andreas S. Kronfeld<sup>d</sup>, Paul B. Mackenzie<sup>d\*</sup>, Damian P. Menscher<sup>b</sup>, Mehmet B. Oktay<sup>b</sup>, Masataka Okamoto<sup>d</sup>, and James N. Simone<sup>d</sup>

<sup>a</sup>School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, IL 60604

<sup>b</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>c</sup> Department of Physics, Indiana University, Bloomington, IN 47405

<sup>d</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

We present preliminary results for the  $D_s$  meson spectrum and decay constants in unquenched lattice QCD. Simulations are carried out with  $2 + 1$  dynamical quarks using gauge configurations generated by the MILC collaboration. We use the “asqtad”  $a^2$  improved staggered action for the light quarks, and the clover heavy quark action with the Fermilab interpretation. We compare our spectrum results with the newly discovered  $0^+$  and  $1^+$  states in the  $D_s$  system.

### 1. INTRODUCTION

$D$  physics is assuming a larger importance in lattice QCD and in CKM phenomenology with the arrival of the CLEO-c charm factory [1]. CLEO-c will measure the leptonic decay constants  $f_D$  and  $f_{D_s}$  to an accuracy of around 2%. It will measure the amplitudes of the semileptonic decays  $D \rightarrow \pi l\nu$  and  $D \rightarrow K l\nu$  to an accuracy of around 1%. This will produce new determinations of the CKM matrix elements  $V_{cd}$  and  $V_{cs}$  to the accuracy that can be achieved in the required lattice calculations, and new checks of the unitarity triangle. New, precise tests of lattice QCD will come from the amplitude ratios  $f_D/D \rightarrow \pi l\nu$  and  $f_{D_s}/D \rightarrow K l\nu$ . These ratios, which are independent of the CKM matrix, will provide the most precise tests in existence of the types of lattice QCD calculations required to extract CKM matrix elements from  $B$  physics and  $D$  physics.

The recent discovery of the positive parity partners of the  $D_s$  and the  $D_s^*$  [2] has added new interest to the spectrum of the  $D_s$  system. The new

states lie significantly below quark model predictions, and below the  $DK$  threshold, so the states are quite narrow.

### 2. METHODS

We use the “asqtad” order  $a^2$  improved staggered light quark action, and the order  $a$  improved Fermilab heavy quark action. In unquenched calculations, chiral extrapolation is the least well controlled remaining source of error. staggered fermions are the only method currently able to reach the region  $m_l \sim m_s/5$  that seems to be required to control this error. Using unquenched calculations that reach this region, a number of simple quantities that disagree with each other at the 10% level in the quenched approximation come into good agreement [3]. We use the public MILC unquenched configurations, with two light and one strange sea quarks, with properties

- $20^3 \times 64$ ,  $a \sim 1/8$  fm,
- $(m_l, m_s) = (0.007, 0.05), (0.01, 0.05), (0.02, 0.05), (0.03, 0.05)$ , (True  $m_s \approx 0.041$ ),

\*Talk given by Paul Mackenzie, mackenzie@fnal.gov

- $\sim 500$  configurations at each mass, 4 time sources each,

as described in [4].

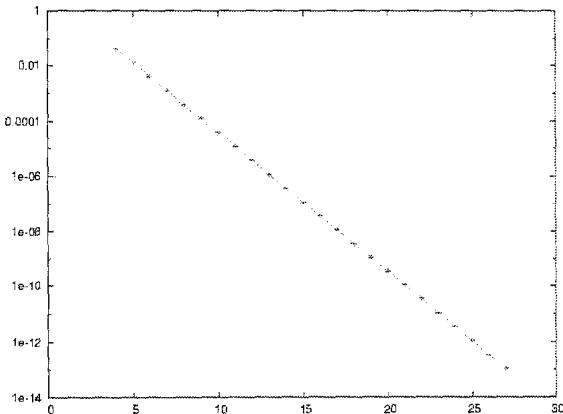


Figure 1. The  $D_s$  correlation function.

With staggered and naive light quarks, the  $0^+$  and  $1^+$  are in the  $D_s$  and  $D_s^*$  correlators automatically and unavoidably because of fermion doubling. Naive fermions possess a doubling symmetry,  $\psi \rightarrow (i\gamma_\mu\gamma_5)(-1)^\mu\psi$ , that implies that quarks come in multiples with identical properties. Local fermion operators connect to all of these doubled modes. Therefore, because of the doubling symmetry, an pseudoscalar current couples to a positive parity scalar state as well:  $\Psi_h\gamma_5\psi \rightarrow \Psi_h\gamma_5(i\gamma_0\gamma_5)(-1)^t\psi = i\Psi_h\gamma_0\psi(-1)^t$ . Oscillating signals for the positive parity states are present in the correlation functions for the  $D_s$  and  $D_s^*$  mesons, but they are small and hard to see, as can be seen in fig. 1, Fig. 2 shows the same correlation function with the leading exponential scaled out. The decay of the oscillating signal gives the mass of the  $0^+$  state. Because it is an excited state, the presence of a plateau is more difficult to ascertain than for a ground state.

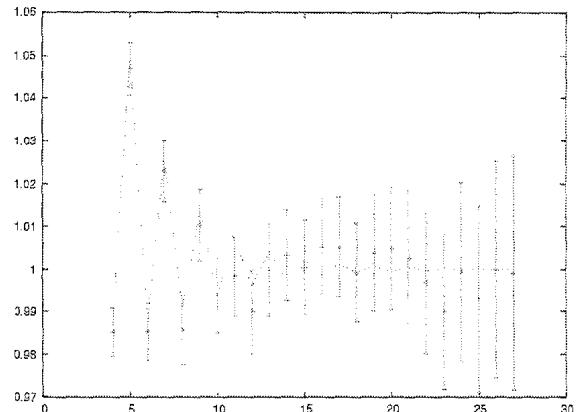


Figure 2. the  $D_s$  correlation function with the leading exponential scaled out. The exponential decay of the oscillating signal gives the mass of the  $0^+$  state.

### 3. SPECTRUM

The  $D_s$  spectrum on the  $m_l, m_s = 0.01, 0.05$  lattices is shown in fig. 3. Very little dependence on the sea quark mass is observed. The  $0^+$  and  $1^+$  states lie somewhat above experiment. However, they are excited states and are somewhat more sensitive to the choices of the priors for the higher states in the Bayes fits than are the ground states. We obtain 0.86 (2) for the hyperfine splitting divided by its experimental value. This may be compared with 0.82 (2) for the hyperfine splitting in charmonium. The most likely source for these errors is the one-loop correction to  $O(a)\bar{\psi}\Sigma \cdot B\psi$  operator in the clover action. Since this operator contributes twice to the charmonium hyperfine splitting and once to the the  $D_s$  hyperfine splitting, it is reasonable to expect a larger effect in charmonium.

### 4. LEPTONIC DECAY AMPLITUDE

This work is part of a larger project with the MILC collaboration. Preliminary results for the amplitude for leptonic decay of the  $D_s$  are shown in fig. 4 as a function of the sea quark mass (using

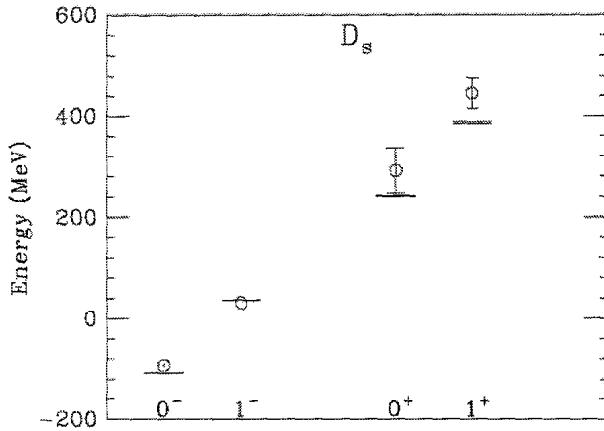


Figure 3. The  $D_s$  spectrum on the  $m_l, m_s = 0.01, 0.05$  lattices.

$m_s = 0.041$ ). There is little dependence on the sea quark mass, as expected. The one-loop Fermilab heavy-staggered light axial current renormalization is in progress [5]. To make an estimate of the one-loop correction, we use the formula  $Z_A^{hl} = \rho_A \sqrt{Z_V^{hh} Z_V^{ll}}$  [6], using  $Z_V^{hh} = 1.33(2)$  and  $Z_V^{ll} = 0.86(5)$  [7]. This leads to a current result for the decay constant of  $f_{D_s} = 240$  MeV  $+/- \mathcal{O}(\alpha)$ . However the order  $\alpha$  correction could be large, potentially as large as 30%. This is to be compared with Ryan's unquenched world average of 250 (30) MeV [8] and the unquenched NRQCD heavy staggered light result of  $289(\alpha^2)$  [9]. The CKM independent quantity  $f_{D_s}/f_{D_s}^{D \rightarrow K}$  can be formed by combining this result with the result for the semileptonic amplitude in ref. [7]. Since high precision is the ultimate goal, we will not quote a result in this early stage of the error analysis.

## ACKNOWLEDGEMENTS

We thank Claude Bernard, Christine Davies, Junko Shigemitsu, Peter Lepage, and Matt Wingate for helpful conversations. We thank the MILC Collaboration for their gauge configurations. These calculations were done on the PC

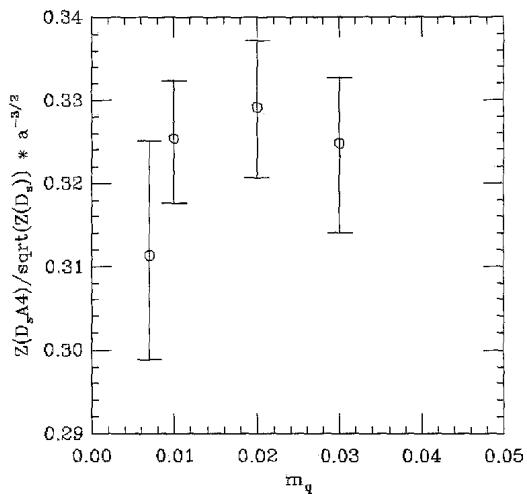


Figure 4. The  $D_s$  leptonic decay amplitude as a function of sea quark mass.

clusters deployed at Fermilab under the Doe SciDAC Program.

## REFERENCES

1. R. A. Briere *et al.*, CLNS-01-1742
2. D. Besson *et al.* [CLEO Collaboration], Phys. Rev. D **68**, 032002 (2003) [arXiv:hep-ex/0305100].
3. C. T. Davies *et al.* [HPQCD Collaboration], arXiv:hep-lat/0304004.
4. C. W. Bernard *et al.*, Phys. Rev. D **64**, 054506 (2001) [arXiv:hep-lat/0104002].
5. Aida El-Khadra and Matthew Nobes, private communication.
6. J. Harada, S. Hashimoto, A. S. Kronfeld and T. Onogi, Phys. Rev. D **67**, 014503 (2003) [arXiv:hep-lat/0208004].
7. M. Okamoto *et al.*, arXiv:hep-lat/0309107.
8. S. M. Ryan, Nucl. Phys. Proc. Suppl. **106**, 86 (2002) [arXiv:hep-lat/0111010].
9. M. Wingate, C. Davies, A. Gray, E. Gulez, G. P. Lepage and J. Shigemitsu, arXiv:hep-lat/0309092.

## High-Precision Lattice QCD Confronts Experiment

C. T. H. Davies,<sup>1</sup> E. Follana,<sup>1</sup> A. Gray,<sup>1</sup> G. P. Lepage,<sup>2</sup> Q. Mason,<sup>2</sup> M. Nobes,<sup>3</sup> J. Shigemitsu,<sup>4</sup>  
H. D. Trottier,<sup>3</sup> and M. Wingate<sup>4</sup>

(HPQCD and UKQCD Collaborations)

<sup>1</sup>*Department of Physics and Astronomy, University of Glasgow, Glasgow, United Kingdom*

<sup>2</sup>*Laboratory for Elementary-Particle Physics, Cornell University, Ithaca, New York 14853, USA*

<sup>3</sup>*Physics Department, Simon Fraser University, Vancouver, British Columbia, Canada*

<sup>4</sup>*Physics Department, The Ohio State University, Columbus, Ohio 43210, USA*

C. Aubin,<sup>5</sup> C. Bernard,<sup>5</sup> T. Burch,<sup>6</sup> C. DeTar,<sup>7</sup> Steven Gottlieb,<sup>8</sup> E. B. Gregory,<sup>6</sup> U. M. Heller,<sup>9</sup> J. E. Hetrick,<sup>10</sup>  
J. Osborn,<sup>7</sup> R. Sugar,<sup>11</sup> and D. Toussaint<sup>6</sup>

(MILC Collaboration)

<sup>5</sup>*Department of Physics, Washington University, St. Louis, Missouri 63130, USA*

<sup>6</sup>*Department of Physics, University of Arizona, Tucson, Arizona 85721, USA*

<sup>7</sup>*Physics Department, University of Utah, Salt Lake City, Utah 84112, USA*

<sup>8</sup>*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

<sup>9</sup>*American Physical Society, One Research Road, Box 9000, Ridge, New York 11961-9000, USA*

<sup>10</sup>*University of the Pacific, Stockton, California 95211, USA*

<sup>11</sup>*Department of Physics, University of California, Santa Barbara, California 93106, USA*

M. Di Pierro,<sup>12</sup> A. El-Khadra,<sup>13,14,15</sup> A. S. Kronfeld,<sup>14</sup> P. B. Mackenzie,<sup>14</sup> D. Menscher,<sup>13</sup> and J. Simone<sup>14</sup>

(HPQCD and Fermilab Lattice Collaborations)

<sup>12</sup>*School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois 60604, USA*

<sup>13</sup>*Physics Department, University of Illinois, Urbana, Illinois 61801-3080, USA*

<sup>14</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

<sup>15</sup>*Kavli Institute for Theoretical Physics, University of California, Santa Barbara, California 93106, USA*

(Received 7 April 2003; published 15 January 2004)

The recently developed Symanzik-improved staggered-quark discretization allows unquenched lattice-QCD simulations with much smaller (and more realistic) quark masses than previously possible. To test this formalism, we compare experiment with a variety of nonperturbative calculations in QCD drawn from a restricted set of “gold-plated” quantities. We find agreement to within statistical and systematic errors of 3% or less. We discuss the implications for phenomenology and, in particular, for heavy-quark physics.

DOI: 10.1103/PhysRevLett.92.022001

PACS numbers: 12.38.Aw, 11.15.Ha, 12.38.Gc

For almost 30 years, precise numerical studies of non-perturbative QCD, formulated on a space-time lattice, have been stymied by our inability to include the effects of realistic quark vacuum polarization. In this Letter, we present evidence that a milestone has been reached: Simulations that include vacuum-polarization effects for three light quarks are now possible. This implies that accurate nonperturbative QCD calculations for a restricted (“gold-plated”) set of quantities are achievable. The set includes, for example,  $B$  and  $D$  meson decay constants, mixing amplitudes, and semileptonic form factors—all quantities of great importance in current experimental studies of heavy quarks. The key to our work is the use of the Symanzik-improved staggered-

quark formalism for the light quarks [1–3], which allows us to include three dynamical light quarks in simulations, with a physical strange quark mass, and  $u$  and  $d$  quark masses that are 3–5 times smaller than in previous studies.

Quark vacuum polarization is by far the most expensive ingredient in a QCD simulation. It is particularly difficult to simulate with small quark masses, such as  $u$  and  $d$  masses. Consequently, most lattice-QCD (LQCD) simulations in the past have either omitted quark vacuum polarization (“quenched QCD”), or they have included effects for only  $u$  and  $d$  quarks, with masses 10–20 times larger than the correct values. This results in uncontrolled systematic errors that can be as large as

30%. The Symanzik-improved staggered-quark formalism is among the most accurate discretizations, and it is much faster in simulations than current alternatives of comparable accuracy. Furthermore, an exact chiral symmetry of the formalism permits efficient simulations with small quark masses. Consequently, realistic simulations are possible now, with all three light-quark flavors. The smallest  $u$  and  $d$  masses we use are still 3 times too large, but they are now small enough that chiral perturbation theory is a reliable tool for extrapolating to the correct masses, at least for the quantities we study.

In this Letter, we show that LQCD simulations, with improved staggered quarks, can deliver nonperturbative results that are accurate to within a few percent. We do this by comparing LQCD results with experimental measurements. In making this comparison, we restrict ourselves to quantities that are accurately measured ( $<1\%$  errors), and that are expected *a priori* to have small systematic errors in LQCD calculations with existing techniques. The latter restriction excludes unstable hadrons and multihadron states (e.g., in nonleptonic decays); both of these are strongly affected by the finite volume of our lattice (2.5 fm across). Unstable hadrons, such as the  $\rho$  and the  $\phi$ , are constantly fluctuating into on-shell or nearly on-shell decay products that can easily propagate to the boundaries of the lattice; similar problems afflict multihadron states. Consequently, we focus here on hadrons that are at least 100 MeV below decay threshold or have negligible widths ( $\pi$ ,  $K$ ,  $B_s$ ,  $J/\psi$ ,  $\Upsilon$ , ...); and we restrict our attention to hadronic masses, and to hadronic matrix elements that have at most one hadron in the initial and final states. These masses and matrix elements can be called “gold-plated”: LQCD calculations of them must work if LQCD is to be trusted at all.

Unambiguous tests of LQCD are particularly important with staggered quarks. These discretizations have the unusual property that a single quark field  $\psi(x)$  creates four equivalent species or “tastes” of quark. Taste is used to distinguish this property, a lattice artifact, from true quark flavor. A quark vacuum-polarization loop in such formalisms contributes 4 times what it should. To remove the duplication, the quark determinant in the path integral is replaced by its fourth root. This construction introduces nonlocalities that are potentially worrisome, but it is the price paid for speed. Much is known that is reassuring: For example, no problems result from fractional roots of the fermion determinant in any order of continuum QCD perturbation theory [4]; phenomena, such as  $\pi^0 \rightarrow 2\gamma$ , connected with chiral anomalies are correctly handled [because the relevant (taste-singlet) currents are only approximately conserved [5]]; the  $CP$  violating phase transition that occurs when  $m_u + m_d < 0$  does *not* occur in this formalism, but the real world is neither in this phase nor near it; the nonperturbative quark-loop structure is correct except for taste-changing interactions. Taste-changing interactions are short dis-

tance, so they can be removed with perturbation theory [6]—at present to order  $a^2 \alpha_s$ . They may also be removed after the simulation with modified chiral perturbation theory [7]. To press further requires nonperturbative studies. The tests we present here are among the most stringent nonperturbative tests ever of staggered quarks (and indeed of LQCD).

The gluon configurations that we used, together with the raw simulation data for pions and kaons, were produced by the MILC collaboration; heavy-quark propagators came from the HPQCD collaboration. The lattices have lattice spacings of approximately  $a = 1/8$  fm and  $a = 1/11$  fm. The simulations employed an  $\mathcal{O}(a^2)$  improved staggered-quark discretization of the light-quark action [2], a “tadpole-improved”  $\mathcal{O}(a^2 \alpha_s)$  accurate discretization of the gluon action [8], an  $\mathcal{O}(a^2, v^4)$  improved lattice version of nonrelativistic QCD (NRQCD) for  $b$  quarks [9], and the Fermilab action for  $c$  quarks [10]. Several valence  $u/d$  quark masses, ranging from  $m_s/2$  to  $m_s/8$ , were needed for accurate extrapolations, as were sea  $u/d$  masses ranging between  $m_s/2$  and  $m_s/6$ . Only  $u$ ,  $d$ , and  $s$  quark vacuum polarization was included; effects from  $c$ ,  $b$ , and  $t$  quarks are negligible ( $<1\%$ ) here.

To test LQCD, we first tuned its five parameters to make the simulation reproduce experiment for five well-measured quantities. The five parameters are the bare  $u$  and  $d$  quark masses, which we set equal, the bare  $s$ ,  $c$ , and  $b$  masses, and the bare QCD coupling. There are no further free parameters once these are tuned.

Setting  $m_u = m_d$  simplifies our analysis, and has a negligible effect ( $<1\%$ ) on isospin-averaged quantities. We tuned the  $u/d$ ,  $s$ ,  $c$ , and  $b$  masses to reproduce experimentally measured values of  $m_\pi^2$ ,  $2m_K^2 - m_\pi^2$ ,  $m_{D_s}$ , and  $m_Y$ , respectively. In each case, the experimental quantity is approximately proportional to the corresponding parameter, approximately independent of the other parameters, and gold plated.

Rather than tune the bare coupling, one normally sets it to a particular value, and determines the lattice spacing  $a$  in its place (*after* the simulation). We adjusted the lattice spacing to make the  $Y-Y'$  mass difference agree with experiment. We chose this mass difference since it is almost independent of all quark masses, including, in fact, the  $b$  mass [11].

Having tuned all free parameters in the simulation, we then computed a variety of gold-plated quantities (in addition to the five used for tuning). Our results are summarized in Fig. 1, where we plot the ratio of LQCD results to experimental results for nine quantities:  $\pi$  and  $K$  decay constants, a baryon mass splitting, a  $B_s$ - $\Upsilon$  splitting, and mass differences between various  $J/\psi$  and  $\Upsilon$  states. On the left, we show ratios from QCD simulations without quark vacuum polarization ( $n_f = 0$ ). These results deviate from experiment by as much as 10%–15%; the deviations can be made as large as 20%–30% by tuning QCD’s input parameters against

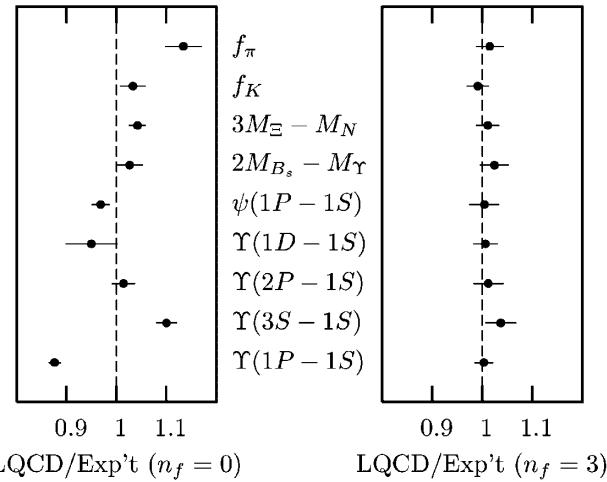


FIG. 1. LQCD results divided by experimental results for nine different quantities, without and with quark vacuum polarization (left and right panels, respectively). The top three results are from our  $a = 1/11$  and  $1/8$  fm simulations; all others are from  $a = 1/8$  fm simulations.

different physical quantities. The right panel shows results from QCD simulations that include realistic vacuum polarization. These nine results agree with experiment to within systematic and statistical errors of 3% or less—with no free parameters.

The quantities used in this plot were chosen to test several different aspects of LQCD. Our results for  $f_\pi$  and  $f_K$  are sensitive to light-quark masses; they test our ability to extrapolate these masses to their correct values using chiral perturbation theory. Accurate simulations for a wide range of small quark masses were essential here. The remaining quantities are much less sensitive to the valence  $u/d$  mass, and therefore are more stringent tests of LQCD since discrepancies cannot be due to tuning errors in the  $u/d$  mass. The  $\Xi$  mass tests our ability to analyze (strange) baryons, while the  $B_s$  mass tests our formalism for heavy quarks. The  $b$  rest mass cancels in  $2M_{B_s} - M_Y$ , making this a particularly clean and sensitive test. The same is true of all the  $Y$  splittings, and our simulations confirm that these are also independent ( $\leq 1\%-2\%$ ) of the sea-quark masses for our smallest masses, and of the lattice spacing (by comparing with  $r_0$  and  $r_1$  computed from the static-quark potential) [12]. The  $Y(P)$  masses are averages over the known spin states; the  $Y(1D)$  is the  $1^3D_2$  state recently discovered by CLEO [13].

Note that our heavy-quark results come directly from the QCD path integral, with only bare masses and a coupling as inputs—five numbers. Furthermore, unlike in quark models or heavy-quark effective theory (HQET),  $Y$  physics in LQCD is inextricably linked to  $B$  physics, through the  $b$ -quark action. Our results confirm that effective field theories, such as NRQCD and the Fermilab

formalism, are reliable and accurate tools for analyzing heavy-quark dynamics.

A serious problem in the previous work was the inconsistency between light-hadron,  $B/D$ , and  $Y/\psi$  quantities. Heavy-quark masses and inverse lattice spacings, for example, were routinely retuned by 10%–20% when going from an  $Y$  analysis to a  $B$  analysis in the *same* quenched simulation [14]. Such discrepancies lead to the results shown in the left panel of Fig. 1. The results in the right panel for  $\pi$ ,  $K$ ,  $\Xi$ ,  $D_s$ ,  $J/\psi$ ,  $B_s$ , and  $Y$  physics mark the first time that agreement has been achieved among such diverse physical quantities using the same QCD parameters throughout.

The dominant uncertainty in our light-quark quantities comes from our extrapolations in the sea and valence light-quark masses. We used partially quenched chiral perturbation theory to extrapolate pion and kaon masses, and the weak decay constants  $f_\pi$  and  $f_K$ . The  $s$ -quark mass required only a small shift; we estimated corrections due to this shift by interpolation (for valence  $s$  quarks) or from the sea  $u/d$  mass dependence (for sea  $s$  quarks). We kept  $u/d$  masses smaller than  $m_s/2$  in our fits, so that low-order chiral perturbation theory was sufficient. Our chiral expansions included the full first-order contribution [15], and also approximate second-order terms, which are essential given our quark masses. We corrected for errors caused by the finite volume of our lattice (1% errors or less), and by the finite lattice spacing (2%–3% errors). The former corrections were determined from chiral perturbation theory; the latter by comparing results from the coarse and fine lattices. Residual discretization errors, due to nonanalytic taste violations [7] that remain after linear extrapolation in  $a^2$ , were estimated as 2% for  $f_\pi$  and 1% for  $f_K$ . Perturbative matching was unnecessary for the decay constants since they were extracted from partially conserved currents. Our final results agree with experiment to within systematic and statistical uncertainties of 2.8%. For the  $n_f = 0$  case, we analyzed only  $a = 1/8$  fm, but corrected for discretization errors by assuming these are the same as in our  $n_f = 3$  analysis.

Figure 2, which shows our fits for  $f_\pi$  and  $f_K$  as functions of the valence  $u/d$  mass, demonstrates that the  $u/d$  masses currently accessible with improved staggered quarks are small enough for reliable and accurate chiral extrapolations, at least for pions and kaons. The valence and sea  $s$ -quark masses were 14% too high in these particular simulations; and the sea  $u/d$  masses were too large— $m_s/2.3$  and  $m_s/4.5$  for the top and bottom results in each pair (fit simultaneously by a single fit function). The dashed lines show the fit functions with corrected valence  $s$  and sea  $u/d/s$  quark masses; these lines extrapolate to our final fit results. The bursts mark the experimental values. Our extrapolations are not large—only 4%–9%. Indeed the masses are sufficiently small that simple linear extrapolations give the same results as our

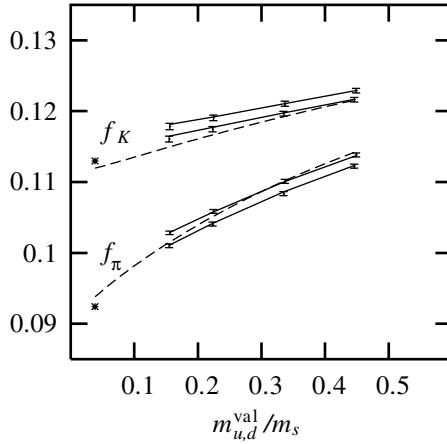


FIG. 2. Chiral fits to LQCD determinations of  $f_\pi$  and  $f_K$  (in GeV) for different values of the valence  $u/d$ -quark mass at  $a = 1/11$  fm.

fits, within few percent errors. These decay constants represent the current state of an ongoing project; a more thorough analysis will be published soon [16].

As a final test of high-precision LQCD, we examine the applicability of perturbation theory, which is essential for connecting most interesting lattice results to the continuum. We tested perturbation theory by extracting values of the coupling from our simulations and comparing them with non-LQCD results. We determined the renormalized coupling,  $\alpha_V(6.3 \text{ GeV})$ , by comparing second-order perturbation theory for the expectation value of a  $1 \times 1$  Wilson loop with (exact) values from the simulations [11,17]. Results for several sea-quark masses are shown in Table I; the masses become more realistic as one moves down the table.

The QCD coupling is particularly sensitive to the tuning of the lattice spacing. We show results for two different tunings: one using the  $Y(1P - 1S)$  splitting, and the other using  $Y(2S - 1S)$ . The two tunings give couplings that are ten statistical standard deviations apart (systematic errors correlate) and 25% too small when sea-quark masses are infinite.

TABLE I. The QCD coupling  $\alpha_V(6.3 \text{ GeV})$  from  $1 \times 1$  Wilson loops in simulations with different  $u/d$  and  $s$  sea-quark masses (in units of the physical  $s$  mass), and using two different tunings for the lattice spacing. The first error shown is statistical, and the second is truncation error which we take to be  $\mathcal{O}(1\alpha_V^3)$  [11].

$a$ (fm)	$m_{u,d}$	$m_s$	$1P - 1S$	$2S - 1S$
1/8	$\infty$	$\infty$	0.177 (1)(5)	0.168 (0)(4)
1/8	0.5	$\infty$	0.211 (1)(9)	0.206 (1)(8)
1/8	1.3	1.3	0.231 (2)(12)	0.226 (2)(11)
1/8	0.5	1.3	0.234 (2)(12)	0.233 (1)(12)
1/8	0.2	1.3	0.234 (1)(12)	0.234 (1)(12)
1/11	0.2	1.1	0.238 (1)(13)	0.236 (1)(13)

With smaller, more realistic sea-quark masses, the two tunings agree to within 1% (as expected from Fig. 1), and the coupling becomes mass independent. Our results, converted to the modified minimal subtraction scheme ( $\overline{\text{MS}}$ ) and evolved perturbatively to scale  $M_Z$ , imply  $\alpha_{\overline{\text{MS}}}^{(5)}(M_Z) = 0.121(3)$ , which agrees with the current world average of 0.117(2) [18]. Unlike previous determinations, ours includes realistic quark vacuum polarization,  $\mathcal{O}(a^2)$  improved actions, and a thorough study of the quark mass dependence (or independence); it is further supported by a wide range of heavy-quark *and* light-quark calculations. A more detailed discussion, with results from other short-distance quantities (they agree), will be presented elsewhere.

Our results suggest that light improved staggered quarks, with NRQCD or Fermilab heavy quarks, enable accurate nonperturbative calculations for gold-plated quantities. Further work is required, however. Chiral extrapolations for nonstrange baryons, for example, are expected to be larger than for pions and kaons, as are finite volume and statistical errors; computations with these hadrons are not yet under control. Also, there are many gold-plated quantities that we have not yet fully analyzed. Heavy-quark mixing amplitudes, and semileptonic decay form factors, for example, are essential to high-precision experiments at CLEO-c and the  $B$  factories; our lattice techniques for these require independent tests.

The larger challenge facing LQCD is to exploit these new techniques in the discovery of new physics. Again,  $B$  and  $D$  physics offer extraordinary opportunities for new physics from LQCD. There are, for example, gold-plated lattice quantities for every Cabibbo-Kobayashi-Maskawa (CKM) matrix element except  $V_{tb}$  (Fig. 3). An immediate challenge is to predict the  $D/D_s$  leptonic and semileptonic decay rates to within a few percent *before* CLEO-c measures them.

$$\left( \begin{array}{ccc} V_{ud} & V_{us} & V_{ub} \\ \pi \rightarrow l\nu & K \rightarrow l\nu & B \rightarrow \pi l\nu \\ & K \rightarrow \pi l\nu & \\ V_{cd} & V_{cs} & V_{cb} \\ D \rightarrow l\nu & D_s \rightarrow l\nu & B \rightarrow D l\nu \\ D \rightarrow \pi l\nu & D \rightarrow K l\nu & \\ V_{td} & V_{ts} & V_{tb} \\ \langle B_d | \bar{B}_d \rangle & \langle B_s | \bar{B}_s \rangle & \end{array} \right)$$

FIG. 3. Gold-plated LQCD processes that bear on CKM matrix elements.  $\epsilon_K$  is another gold-plated quantity.

This work was supported by PPARC, the NSF, and the DOE, and by computing allocations at NERSC, LANL, ORNL, SDSC, NCSA, PSC, FNAL, and Indiana. We thank M. Alford, T. DeGrand, P. Drell, L. Gibbons, J. Hein, M. Peskin, and E. Witten for useful discussions and comments.

- 
- [1] C.W. Bernard *et al.*, Nucl. Phys. (Proc. Suppl.) **B60**, 297 (1998); G. P. Lepage, Nucl. Phys. (Proc. Suppl.) **B60**, 267 (1998); MILC Collaboration, C.W. Bernard *et al.*, Phys. Rev. D **58**, 014503 (1998).
  - [2] G. P. Lepage, Phys. Rev. D **59**, 074502 (1999).
  - [3] MILC Collaboration, K. Orginos *et al.*, Phys. Rev. D **59**, 014501 (1999); MILC Collaboration, K. Orginos *et al.*, Nucl. Phys. (Proc. Suppl.) **B73**, 909 (1999); MILC Collaboration, K. Orginos *et al.*, Phys. Rev. D **60**, 054503 (1999); MILC Collaboration, C.W. Bernard *et al.*, Phys. Rev. D **61**, 111502 (2000).
  - [4] G.G. Batrouni *et al.*, Phys. Rev. D **32**, 2736 (1985).
  - [5] H.S. Sharatchandra, H.J. Thun, and P. Weisz, Nucl. Phys. **B192**, 205 (1981); J. Smit and J.C. Vink, Nucl. Phys. **B298**, 557 (1988).
  - [6] HPQCD Collaboration, Q. Mason *et al.*, Nucl. Phys. (Proc. Suppl.) **B119**, 446 (2003).
  - [7] MILC Collaboration, C. Bernard, Phys. Rev. D **65**, 054031 (2002); C. Aubin *et al.*, Nucl. Phys. (Proc. Suppl.) **B119**, 233 (2003); C. Aubin and C. Bernard, Phys. Rev. D **68**, 034014 (2003); **68**, 074011 (2003).
  - [8] M.G. Alford, W. Dimm, G.P. Lepage, G. Hockney, and P.B. Mackenzie, Phys. Lett. B **361**, 87 (1995).
  - [9] C.T.H. Davies *et al.*, Phys. Rev. D **52**, 6519 (1995).
  - [10] A.X. El-Khadra, A.S. Kronfeld, and P.B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
  - [11] C.T.H. Davies *et al.*, Phys. Rev. D **56**, 2755 (1997).
  - [12] C.T.H. Davies *et al.*, Phys. Rev. D **58**, 054505 (1998).
  - [13] CLEO Collaboration, S.E. Csorna *et al.*, hep-ex/0207060.
  - [14] A. Ali Khan *et al.*, Phys. Rev. D **53**, 6433 (1996).
  - [15] S.R. Sharpe and N. Shores, Phys. Rev. D **62**, 094503 (2000).
  - [16] S. Gottlieb, hep-lat/0310041.
  - [17] C.T.H. Davies *et al.*, Nucl. Phys. (Proc. Suppl.) **B119**, 595 (2003).
  - [18] K. Hagiwara *et al.*, Phys. Rev. D **66**, 010001 (2002).

# Properties of charmonium in lattice QCD with $2 + 1$ flavors of improved staggered sea quarks

Massimo di Pierro<sup>a</sup>, Aida X. El-Khadra<sup>b</sup>, Steven Gottlieb<sup>c</sup>, Andreas S. Kronfeld<sup>d</sup>, Paul B. Mackenzie<sup>d</sup>, Damian P. Menscher<sup>b</sup>, Mehmet B. Oktay<sup>b</sup>, Masataka Okamoto<sup>d</sup>, and James N. Simone<sup>d\*</sup>

<sup>a</sup>School of Computer Science, Telecommunications, and Information Systems, DePaul University, Chicago, IL 60604

<sup>b</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>c</sup> Department of Physics, Indiana University, Bloomington, IN 47405

<sup>d</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

We use the dynamical gluon configurations provided by the MILC collaboration in a study of the charmonium spectrum and  $\psi$  leptonic width. We examine sea quark effects on mass splitting and on the leptonic decay matrix element for light masses as low as  $m_s/5$ , while keeping the strange quark mass fixed and the lattice spacing nearly constant.

## 1. INTRODUCTION

Charmonium states below open flavor threshold are considered gold-plated quantities in lattice QCD. They are almost stable mesons, which should be accurately calculable in lattice QCD once realistic sea-quark effects are included in the simulations. A high precision study of the charmonium system in unquenched lattice QCD is interesting for several reasons. First, it provides us with an important test of the lattice methods, because the methods used for charmonium calculations are similar to those used for CKM determinations in the  $D$  meson system [1,2]. Second, it allows us to test our improved actions which are under development [3], since different splittings in the charmonium system are sensitive to different correction operators in the action. For example, the hyperfine splitting is very sensitive to  $\bar{\psi}\sigma \cdot B\psi$ , while the  $\chi_c$  fine structure is expected to sensitively depend on  $\bar{\psi}\sigma \cdot (D \times E)\psi$ . Finally, together with 2-loop perturbation theory, it yields precise determinations of  $\alpha_s$  and the charm quark mass.

The  $2 + 1$  dynamical gauge configurations gen-

erated by the MILC collaboration contain realistic sea quark effects, since they reach the chiral region ( $m_l \geq m_s/5$ ) – a necessary feature to control chiral extrapolation errors. A first test of lattice QCD calculations which use the MILC configurations was presented in Ref. [9]. For the first time agreement (at the few % level) with experiment was achieved for a variety of different physical systems, involving  $b$ ,  $c$ , and light quarks. This comparison includes our results for the  $1P-1S$  splitting in charmonium which are also presented here.

The work presented here continues our charmonium study [4,5]. Our companion study of the  $D_s$  and  $D$  meson spectra and weak decays is presented in Ref. [1,2].

## 2. METHODS

We are using the MILC collaboration “Asqtad” gluon ensembles [6,7]. The Asqtad action has leading  $\mathcal{O}(\alpha_s^2 a^2)$  gluon uncertainties and leading  $\mathcal{O}(\alpha_s a^2)$  uncertainties for the improved staggered sea quarks.

The gluon ensembles have one flavor approximating the strange quark and two equal-mass

\*Talk presented by J. Simone, simone@fnal.gov

lighter flavors. A matched set of gluon ensembles is available having light masses in the range  $m_s$  to  $m_s/5$  and nearly constant lattice spacing (see Table 1).

Our charm quarks are  $\mathcal{O}(a)$ -improved Wilson fermions in the Fermilab interpretation of heavy quarks. The coefficient of the clover term has the tadpole-improved tree-level value. The bare charm quark mass is tuned by demanding that the  $D_s$  meson kinetic mass equal the experimental value.

### 3. CHARMONIUM SPECTRUM

Fig. 1 shows the overall picture of the charmonium spectrum after sea quark extrapolations. The zero of energy is taken to be the spin average of the  $1S$  masses. The lattice spacing is determined using the  $h_c(1P)$  splitting as input. This lattice spacing is consistent, at the few percent level, with other ways of setting the lattice spacing [9]. We compare the charmonium  $1P-1S$  and bottomonium  $2S-1S$  lattice spacings in Table 1.

Table 1

The bare light sea quark mass and coupling  $\beta$  for the MILC three-flavor gluon ensembles. The strange quark has mass  $am_s = 0.05$ . Lattice spacings are given in GeV. The  $2S-1S$  results are from Reference [8].

$am_l$	beta	cfgs	$a_\psi^{-1}(1P-1S)$	$a_\Upsilon^{-1}(2S-1S)$
0.007	6.76	403	1.55(3)	—
0.01	6.76	593	1.56(2)	1.59(2)
0.02	6.79	460	1.59(3)	1.61(2)
0.03	6.81	549	1.57(3)	1.60(3)

We extrapolate linearly in the light sea quark mass. The mass dependence is mild: linear terms are typically of the same order of magnitude as their statistical error. The mass dependence for the hyperfine splitting, shown in Fig. 2, illustrates a typical extrapolation. Smaller statistical errors and more sea quark masses would be needed to better resolve terms in the chiral expansions.

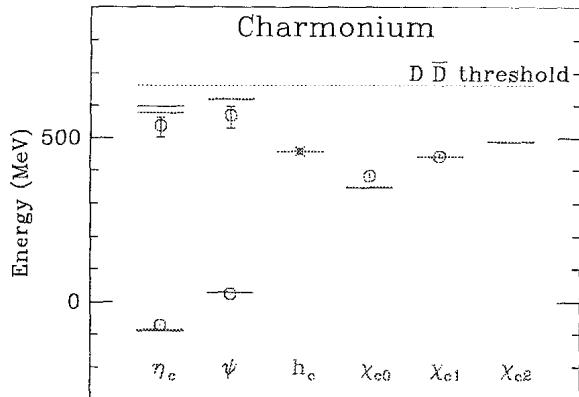


Figure 1. The charmonium spectrum in  $2+1$  flavor lattice QCD after extrapolation to the physical sea quark masses. The lattice spacing is  $a \approx 1/8$  fm. No lattice result for the  $\chi_{c2}(1P)$  splitting is shown since that state was not computed in this study. The dotted indicates the  $D\bar{D}$  threshold energy.

The  $2S$  splittings shown in Fig. 1 have large errors. Statistical uncertainties for these excited states are 20–30 times as large as uncertainties in the ground states with the same  $J^{PC}$ . Ground state and radially excited state energies are obtained from a single fit using Bayesian techniques. We continue to investigate ways to improve the signal for excited states.

#### 3.1. Hyperfine splitting

The hyperfine splitting in charmonium is a gold-plated quantity, which must agree with experiment once all known systematic errors are corrected.

As shown in Fig. 2 we obtain a  $1S$  hyperfine splitting of  $97 \pm 2$  MeV, or in ratio to experiment, theory/expt =  $0.82 \pm 0.02$ . A comparison to our previous quenched result (obtained at similar lattice spacings), theory/expt = 0.6, shows that sea quarks have an appreciable effect on this quantity. The remaining discrepancy is likely due to having only a (tadpole improved) tree-level estimate for the coefficient of the  $\sigma \cdot B$  term in our action. A one-loop calculation of this coefficient is in progress [10].

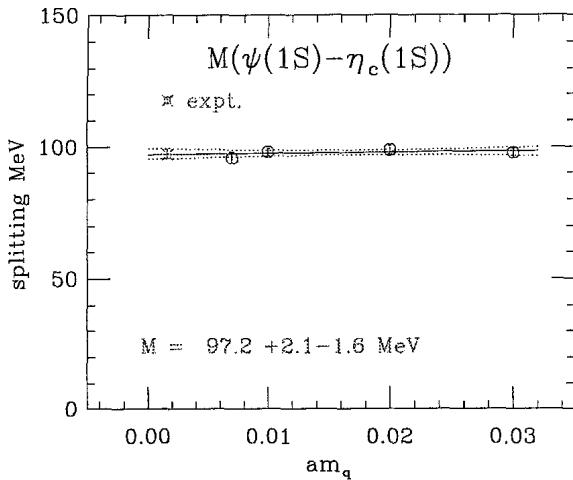


Figure 2. The charmonium 1S hyperfine splitting as a function of the mass of the two light sea quarks. The solid and dotted lines show the linear extrapolation with 68% confidence bounds. The extrapolated value is denoted by the burst symbol. The experimental value lies above the lattice determination.

#### 4. $\psi(1S)$ LEPTONIC WIDTH

We determine the hadronic matrix element,  $V_\psi \equiv \langle 0 | V_j | \psi \rangle$  for the leptonic width from the overlap coefficient of the two-point function of the  $\psi$  propagator annihilated by the local vector current. At present our calculation does not include the  $O(a)$  correction for the vector current.

For the spatial current renormalization, we use the formula  $Z_{V_i} = \rho_{V_i} Z_{V_4}$ , and the nonperturbative result for  $Z_{V_4}$  from Ref. [2]. Since the one-loop correction to  $\rho_{V_i}$  is currently unknown, we have  $\rho_{V_i} = 1$  in our calculation of  $V_\psi$ . Hence, our results for this quantity are very preliminary.

The experimental measurement of the leptonic width implies for the matrix element  $V_\psi^{\text{expt}} = 0.504 \pm 0.018 \text{ GeV}^{3/2}$ .

Fig. 3 shows our preliminary results for  $V_\psi$ ; after sea quark extrapolation we find  $V_\psi^{\text{thy}} = 0.586 \pm 0.015 \text{ GeV}^{3/2}$ . The statistics only uncertainty is dominantly from the lattice spacing determination. We find theory/expt =  $1.16 \pm 0.04$ , combining errors in quadrature.

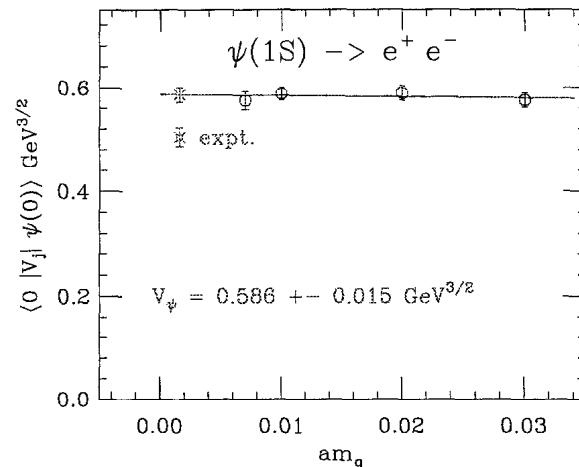


Figure 3. The annihilation matrix element for the decay  $\psi(1S) \rightarrow e^+ e^-$  as a function of the mass of the two light sea quarks. The extrapolated matrix element lies above the value derived from the experimental leptonic decay width.

#### ACKNOWLEDGMENTS

We thank the MILC collaboration for their gauge configurations. These calculations were done on the PC clusters deployed at Fermilab under the DoE SciDAC program. This work was supported in part by the DoE.

#### REFERENCES

1. P. Mackenzie, these proceedings.
2. M. Okamoto, these proceedings.
3. M. Oktay, these proceedings.
4. A. X. El-Khadra *et al.*, Nucl. Phys. Proc. Suppl. **83**, 283 (2000).
5. M. Di Pierro *et al.*, [arXiv:hep-lat/0210051].
6. Steven Gottlieb, these proceedings.
7. C. W. Bernard *et al.*, Phys. Rev. D **64**, 054506 (2001) [arXiv:hep-lat/0104002].
8. M. Wingate, these proceedings.
9. HPQCD Collaboration: C. T. Davies *et al.*, [arXiv:hep-lat/0304004].
10. M. Nobes and H. Trottier, these proceedings.

## Semileptonic decays of $D$ mesons in unquenched lattice QCD\*

Masataka Okamoto<sup>a</sup>, Massimo di Pierro<sup>b</sup>, Aida X. El-Khadra<sup>c</sup>, Steven Gottlieb<sup>d</sup>, Andreas S. Kronfeld<sup>a</sup>, Paul B. Mackenzie<sup>a</sup>, Damian P. Menscher<sup>c</sup>, Mehmet B. Oktay<sup>c</sup>, and James N. Simone<sup>a</sup>

<sup>a</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

<sup>b</sup> School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois 60604

<sup>c</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>d</sup> Department of Physics, Indiana University, Bloomington, IN 47405

We present our preliminary results for semileptonic form factors of  $D$  mesons in unquenched lattice QCD. Simulations are carried out with  $n_f = 2 + 1$  dynamical quarks using gauge configurations generated by the MILC collaboration. For the valence quarks, we adopt an improved staggered light quark action and the clover heavy quark action. Our results for  $D \rightarrow K$  and  $D \rightarrow \pi$  form factors at  $q^2 = 0$  are in agreement with the experimental values.

### 1. INTRODUCTION

In order to extract the CKM matrix element  $|V_{ub}|$  accurately with experimental measurement of the semileptonic decay width, a precision lattice QCD calculation of the  $B \rightarrow \pi$  form factor is required. To check the reliability of lattice calculations of the heavy to light form factors, study of the semileptonic decays of  $D$  mesons, such as  $D \rightarrow K$  and  $D \rightarrow \pi$ , is a good test ground because the corresponding CKM matrices  $|V_{cs}|$  and  $|V_{cd}|$  are known more accurately than  $|V_{ub}|$ . Furthermore, forthcoming experiments by the CLEO-c collaboration will provide more stringent checks of lattice calculations in the  $D$  meson system.

We have started new lattice calculations of heavy quark physics in unquenched QCD[1], and here we report our preliminary results for semileptonic form factors of  $D$  mesons. We are using unquenched gauge configurations with  $n_f = 2 + 1$  improved staggered quarks generated by the MILC collaboration[2], with which the systematic errors due to the quenched approximation should be almost absent. For the valence light quarks, we adopt an improved staggered quark action,

which allow us to simulate at lighter quark mass than previous studies with the Wilson-type light quarks. Hence, our new calculations should have a better control over the chiral extrapolations.

### 2. METHOD

In order to combine the staggered light quark with the Wilson-type heavy quark in heavy-light bilinears, we convert the staggered quark propagator  $g(x, y)$  to the “naive” quark propagator  $G(x, y)$  according to

$$g(x, y) \Omega(x) \Omega^\dagger(y) = G(x, y) \quad (1)$$

with  $\Omega(x) = \gamma_0^{x_0} \gamma_1^{x_1} \gamma_2^{x_2} \gamma_3^{x_3}$ [3]. The 3-point function for the matrix element is then computed as

$$C_{3,\mu}^{D \rightarrow \pi}(t_x, t_y; \mathbf{p}_\pi, \mathbf{p}_D) = \sum_{\mathbf{x}, \mathbf{y}} e^{i(\mathbf{p}_D - \mathbf{p}_\pi)\mathbf{y} - i\mathbf{p}_D \mathbf{x}} \times$$

$$\langle \text{Tr}[g_d^\dagger(y, 0) \Omega^\dagger(y) \gamma_5 \gamma_\mu G_c(y, x) \gamma_5 \Omega(x) g_u(x, 0)] \rangle,$$

where subscripts  $d, c, u$  denote quark flavors. The matrix element can be extracted from the ratio

$$\langle \pi | V_\mu | D \rangle \stackrel{t_x \gg t_y \gg 0}{\sim} \frac{C_{3,\mu}^{D \rightarrow \pi}(t_x, t_y; \mathbf{p}_\pi, \mathbf{p}_D)}{C_2^\pi(t_y, \mathbf{p}_\pi) C_2^D(t_x - t_y, \mathbf{p}_D)} \quad (2)$$

with the  $D$  meson (Wilson-Naive) 2-point function  $C_2^D$  and the pion (Naive-Naive) 2-point function  $C_2^\pi$ . Care is needed, however, for the overall

\*Talk presented by M. Okamoto.

Table 1

Quark mass, statistics and the sink time.

$m_l^{\text{sea}}/m_s^{\text{sea}}$	$m_l^{\text{val}}/m_s^{\text{val}}$	conf	$t_x$
0.01/0.05	0.01/0.0415	$552 \times 4$	20
0.02/0.05	0.02/0.0415	460	20
0.03/0.05	0.03/0.0415	358	22
0.01/0.05	0.0415/0.0415	412	26
$\infty/\infty$	0.0415/0.0415	350	16

normalization of amplitude since the naive quark action describes 16 fermions, which can cause the doubling of these correlation functions.

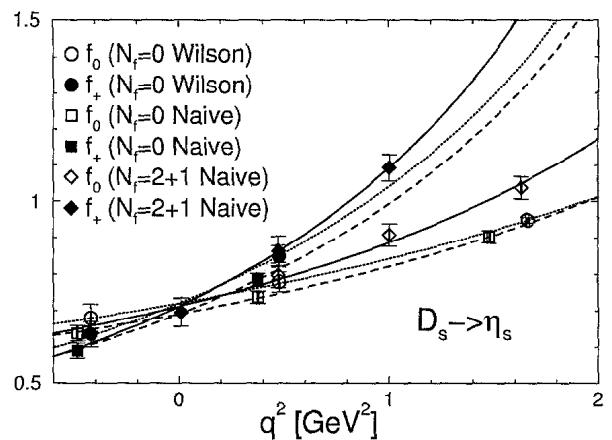
In Ref. [3] it is shown that the Wilson-Naive 2-point function  $C_2^D$  does not have the doubling because contributions of quarks with momentum  $p \sim \mathcal{O}(\pi/a)$  are suppressed by the Wilson term. The same also holds for the 3-point functions which include at least one Wilson propagator such as  $C_{3,\mu}^{D \rightarrow \pi}$ . On the other hand, the Naive-Naive 2-point function  $C_2^\pi$  should have 16 equivalent contributions. Therefore one has to divide it by 16 to get the physical amplitude;  $C_2^{\pi, \text{phys}} = C_2^\pi / 16$ .

### 3. SIMULATION

Unquenched calculations are performed using  $n_f = 2 + 1$  dynamical gauge configurations obtained with an improved staggered quark action on a  $20^3 \times 64$  lattice ( $a^{-1} \approx 1.58$  GeV)[2]. For the valence light quarks we use the same staggered quark action as for the dynamical quarks. The valence light quark ( $u, d$ ) mass  $m_l^{\text{val}}$  is usually set equal to the dynamical light quark mass  $m_l^{\text{sea}}$ . For the valence charm quark we use the clover action with the Fermilab interpretation[4]. The hopping parameter is fixed to  $K_{\text{charm}} = 0.119$ , based on our spectrum study[1]. The  $O(a)$  rotation[4] is performed for the vector current.

The 3-point functions are computed in the  $D$  meson rest frame ( $\mathbf{p}_D = \mathbf{0}$ ) for the light meson momentum  $\mathbf{p}_\pi$  up to  $(1, 1, 1)$  in lattice units, using local source and sink. The sink time is fixed to  $t_x = 20 - 26$  depending on  $m_l^{\text{val}}$ , whereas the source time is set to  $t_0 = 0$  with an exception at  $m_l^{\text{val}} = 0.01$ , where we average over results from four source times  $t_0 = 0, 16, 32$  and  $48$ . Some simulation parameters are summarized in Table 1.

In addition to the unquenched calculations, we

Figure 1.  $D_s \rightarrow \eta_s(s\bar{s})$  form factors.

also perform a quenched simulation at  $m_l^{\text{val}} = m_s^{\text{val}} = 0.0415$  using  $\beta = 5.9$  ( $a^{-1} \approx 1.80$  GeV) configurations on a  $16^3 \times 32$  lattice used in our previous study[5]. Comparison between the quenched result with the staggered light quarks and that with the Wilson-type light quarks[5] allows us to check the validity of our new calculations.

For the vector current renormalization  $Z_{V_\mu}^{cd}$  we follow the method in Ref. [5]. We take  $Z_{V_\mu}^{cd} = \rho_{V_\mu} (Z_V^{cc} Z_V^{dd})^{1/2}$ , where  $Z_V^{qq}$  ( $q = c, d$ ) is the renormalization constant for the flavor-conserving current, which we compute nonperturbatively from the charge normalization condition  $Z_V^{qq} \langle D(\mathbf{0}) | V_4^{qq} | D(\mathbf{0}) \rangle = 2m_D$ . The  $\rho_{V_\mu}$  is set to unity. The one-loop calculation is in progress.

### 4. RESULTS

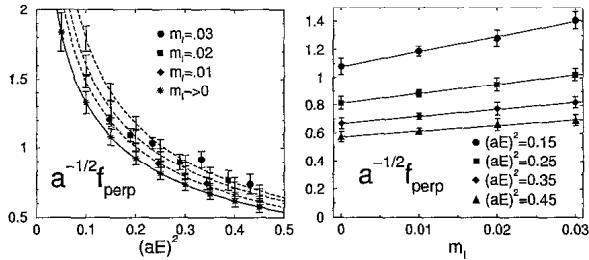
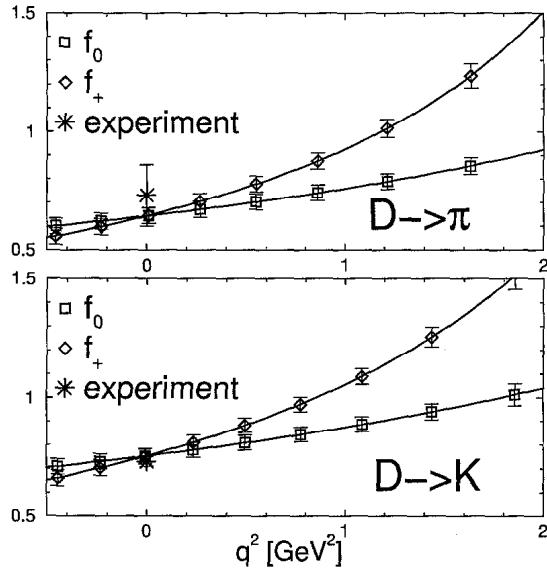
Form factors are defined through

$$\begin{aligned} \langle \pi | V^\mu | D \rangle &= f_+(q^2) \left[ p_D + p_\pi - \frac{m_D^2 - m_\pi^2}{q^2} q \right]^\mu \\ &\quad + f_0(q^2) \frac{m_D^2 - m_\pi^2}{q^2} q^\mu \\ &= \sqrt{2m_D} [v^\mu f_{||}(E) + p_\perp^\mu f_\perp(E)] \end{aligned}$$

with  $q = p_D - p_\pi$ ,  $v = p_D/m_D$ ,  $p_\perp = p_\pi - Ev$  and  $E = E_\pi$ . The second expression using  $f_{||}$  and  $f_\perp$  is more convenient when one considers the heavy quark expansion and the chiral limit.

#### 4.1. $D_s \rightarrow \eta_s$

In Fig. 1 we summarize the results of form factors  $f_0$  and  $f_+$  for the  $D_s \rightarrow \eta_s(s\bar{s})$  decay

Figure 2. Chiral extrapolation for  $f_{\perp}^{D \rightarrow \pi}$ .Figure 3.  $D \rightarrow \pi$  and  $D \rightarrow K$  form factors.

obtained with the naive(staggered) light quarks and previous results[5] with the Wilson-type light quarks. The lines in the figure are fitting curves with a parametrization by Becirevic and Kaidalov (BK)[6]. One can see a nice agreement between the quenched result with the naive quarks (squares, dashed line) and that with the Wilson-type quarks (circles, dotted), showing that our new method works well. We also note that the unquenched result (diamonds, solid) is larger than quenched ones for  $q^2 > 0$ . See also Ref. [7] for a similar comparison for the  $B_s \rightarrow \eta_s$  form factors.

#### 4.2. $D \rightarrow \pi$ and $D \rightarrow K$

To obtain  $D \rightarrow \pi/K$  form factors at the physical quark mass, we need to perform a chiral extrapolation using data in range of  $m_l^{\text{val}} = 0.01-0.03$ . We do this for  $f_{\parallel}$  and  $f_{\perp}$  at fixed pion(kaon) energies  $E_{\pi(K)}$  because the chiral perturbation

formulas for the heavy to light form factors are given in such a way[8]. In order to interpolate and extrapolate the results to common values of  $E_{\pi}$ , we use a fit with the BK parametrization. We then perform a linear chiral extrapolation in  $m_l^{\text{val}}$  at nine values of  $(aE_{\pi})^2$ . One example of these procedures is shown in Fig. 2 for  $f_{\perp}^{D \rightarrow \pi}$ . Finally  $f_{\parallel}$  and  $f_{\perp}$  are converted to  $f_0$  and  $f_+$ .

The  $D \rightarrow \pi$  and  $D \rightarrow K$  form factors are shown in Fig. 3 together with experimental values at  $q^2 = 0$ [9]. Our results at  $q^2 = 0$  are

$$f_+^{D \rightarrow K}(0) = 0.75(3), \quad f_+^{D \rightarrow \pi}(0) = 0.64(3) \quad (3)$$

with statistical errors only, whereas experimental values are  $f_+^{D \rightarrow K}(0) = 0.73(2)$  and  $f_+^{D \rightarrow \pi}(0) = 0.73(13)$  with  $|V_{cs}| = 0.996(13)$  and  $|V_{cd}| = 0.224(16)$ [9]. Our results are in agreement with the experimental values. The analysis including the chiral logarithm and the one-loop renormalization constant is underway.

**Acknowledgments:** We thank the MILC collaboration for the use of their configurations, and the Fermilab Computing Division and the SciDAC program for their support. Fermilab is operated by Universities Research Association Inc., under contract with the DOE.

#### REFERENCES

1. See also, P. Mackenzie and J. Simone, these proceedings.
2. C. Bernard *et al.* (MILC collaboration), Phys. Rev. D **64**, 054506 (2001).
3. M. Wingate *et al.* Phys. Rev. D **67**, 054505 (2003)
4. A. X. El-Khadra, A. Kronfeld and P. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
5. A. X. El-Khadra *et al.* Phys. Rev. D **64**, 014502 (2001)
6. D. Becirevic and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000)
7. C. DeTar, these proceedings.
8. D. Becirevic, S. Prelovsek and J. Zupan, Phys. Rev. D **67**, 054010 (2003)
9. K. Hagiwara *et al.* [Particle Data Group Collaboration], Phys. Rev. D **66**, 010001 (2002); C. Caso *et al.* Eur. Phys. J. C **3**, 1 (1998).



[www.fermiqcd.net](http://www.fermiqcd.net)

Massimo Di Pierro<sup>a\*</sup>, Aida X. El-Khadra<sup>b</sup>, Steven Gottlieb<sup>c</sup>, Andreas S. Kronfeld<sup>d</sup>, Paul B. Mackenzie<sup>d</sup>, Masataka Okamoto<sup>d</sup>, Mehmet B. Oktay<sup>b</sup> and James N. Simone<sup>d</sup> (for the FermiQCD Collaboration)

<sup>a</sup> School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, IL 60604

<sup>b</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>c</sup> Department of Physics, Indiana University, Bloomington, IN 47405

<sup>d</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

## 1. INTRODUCTION

FermiQCD[1][2] is a C++ library for fast development of parallel lattice QCD applications. The expression FermiQCD Collaboration is used as a collective name to indicate both the users of the software and its contributors.

One of the main differences between FermiQCD and libraries developed by other collaborations is that it follows an object oriented design as opposed to a procedural design. FermiQCD should not be identified exclusively with the implementation of the algorithms but, rather, with the strict specifications that define its Application Program Interface. One should think of FermiQCD as a language on its own (a superset of the C++ language), designed to describe Lattice QCD algorithms. The objects of the language include complex numbers (mdp\_complex), matrices (mdp\_matrix), lattices (mdp\_lattice), fields (gauge\_field, fermi\_field, staggered\_field), propagators (fermi\_propagator) and actions. Algorithms written in terms of these objects are automatically parallel.

Some of the advantages of our design approach are the following:

- Programs written in FermiQCD are easy to write, read and modify since the FermiQCD syntax resembles the mathematical syntax used in Quantum Field Theory articles and

books.

- Programs are portable in the sense that they can, in principle, be compiled with any ANSI C++ compiler. Hardware specific optimizations are coded in the library and hidden from the high level programmer.
- The high level programmer does not have to deal with parallelization issues since the underlying objects deal with it. FermiQCD communications are based on MPI.
- Programs are easier to debug because the usage of FermiQCD objects and algorithms does not require explicit use of pointers. All memory management is done by the objects themselves.

FermiQCD was originally designed with one main goal in mind: easy of use. It is true that in many cases this requires a compromise with speed. For example, FermiQCD actions and fields support arbitrary  $SU(n_c)$  gauge group and it is not practical to optimize the linear algebra for any  $n_c$ . However it is convenient to optimize some of the specific cases of interest (such as  $n_c = 3$ ) while maintaining compatibility with the FermiQCD syntax. At present the FermiQCD library includes multiple implementations of each action (Wilson, Clover, Kogut-Susskind, Asqtad[3], Domain Wall, Fermilab[4])

\*presented by M. Di Pierro

and, for each action, at least one implementation is optimized for  $SU(3)$ . In particular, we provide a FermiQCD port of the Clover  $SU(3)$  action coded by M. Lüscher using SSE assembly instructions for Pentium 4[5].

The following code declares two fermionic fields (`phi` and `psi`) and one  $SU(nc)$  gauge field (`U`) on a given lattice (`lattice`), reads `psi` and `U` from disk and computes

$$\varphi = Q[U, \kappa]^{-1} \psi$$

where  $Q[U, \kappa] = D[U] + m[\kappa]$  is the Dirac matrix and  $\kappa$  is the typical lattice parameter that sets the mass scale:

```
fermi_field phi(lattice, nc);
fermi_field psi(lattice, nc);
gauge_field U(lattice, nc);
coefficients light_quark;
psi.load("fermi_field.mdp");
U.load("gauge_field.mdp");
light_quark["kappa"]=0.123;
if(nc==3) default_fermi_action=
    FermiCloverActionSSE2::mul_Q;
mul_invQ(phi,psi,U,light_quark);
```

Note that when  $nc = 3$  the above program uses the SSE optimized Wilson action. If  $nc \neq 3$  a different implementation of the Wilson action is used. FermiQCD programs are transparent to the choice of the action.

On a cluster of 2GHz Pentium 4 PCs connected by Myrinet, one double precision  $SU(3)$  minimum residue step with Clover action takes 4.8 micro seconds per lattice site. The efficiency drops to 75 – 80% on 8 processors. For more benchmarks we refer to the [www.fermiqcd.net](http://www.fermiqcd.net) web page.

FermiQCD programs are implicitly parallel. Each lattice object determines an optimal communication pattern for its sites assuming a next-neighbor, a next-next-neighbor or a next<sup>3</sup>-neighbor interaction in the action. The optimal patterns are determined according to empirical rules that minimize dependence on the network latency and minimize communication load.

## 2. EXAMPLES

### 2.1. Computing the average plaquette

Given a  $\mu\nu$ -plane, the average plaquette, for each gauge configuration, is defined as

$$\langle P_{\mu\nu} \rangle = \frac{1}{V} \text{ReTr} \sum_x U_\mu(x) U_\nu(x + \hat{\mu}) \quad (1)$$

$$[U_\nu(x) U_\mu(x + \hat{\nu})]^H \quad (2)$$

here is the corresponding FermiQCD syntax:

```
forallsites(x)
p=p+real(trace(U(x,mu)*U(x+mu,nu)*
hermitian(U(x,nu)*U(x+nu,mu))));
```

`p=p/lattice.size();`  
In the above code `forallsites` is a parallel loop, `U(x,mu)` is  $n_c \times n_c$  color matrix.

### 2.2. Implementing the Dirac-Wilson action

As another example, we consider here the following Wilson discretization of the Dirac action:

$$\varphi_\alpha = \psi_\alpha - \kappa \sum_\mu (1 + \gamma_\mu)_{\alpha\beta} U_\mu(x) \psi_\beta(x + \hat{\mu}) + \quad (3)$$

$$(1 - \gamma_\mu)_{\alpha\beta} U_\mu^H(x - \hat{\mu}) \psi_\beta(x - \hat{\mu}) \quad (4)$$

which can be translated into the following FermiQCD code:

```
phi=psi;
forallsites(x)
for(int mu=0; mu<4; mu++) {
    for(int b=0; b<4; b++) {
        psi_up(b)=U(x,mu)*psi(x+mu,b);
        psi_dw(b)=hermitian(U(x-mu,mu))*psi(x-mu,b);
    }
    phi=phi-kappa*((1+Gamma[mu])*psi_up+
                    (1-Gamma[mu])*psi_dw);
}
```

Note how “1” in this context is interpreted as a diagonal unitary matrix. The sum on  $\alpha$  is implicit since `psi_up` and `psi_dw` are spin $\times$ color matrices.

### 2.3. Computing the pion propagator

The pion propagator is defined as

$$C_2(x_0) = \sum_{x_1, x_2, x_3} \langle \pi(x) \pi(0) \rangle \quad (5)$$

$$= \text{ReTr} \sum_{x_1, x_2, x_3} \sum_{\alpha, \beta} S_{\alpha\beta}(x) S_{\beta\alpha}^H(x) \quad (6)$$

where  $S$  is a light quark propagator in the background gauge field  $U$ :

$$S_{\alpha\beta}^{ij}(x) = \int [dq d\bar{q}] q(x) \bar{q}(0) e^{Action[q, U]} \quad (7)$$

Here is the FermiQCD syntax for Eq. (6):

```
generate(S,U,light_quark);
forallsites(x)
  for(int a=0; a<4; a++)
    for(int b=0; b<4; b++)
      C2(x(0))+=real(trace(S(x,a,b)*
                           hermitian(S(x,b,a))));
```

where  $S(x, a, b)$  is a color×color matrix. The function `generate` creates the propagator  $S$  by calling the inverter (for example the minimum residue or the stabilized bi-conjugate gradient). Each of the inverters works on any of the provided actions and also on user defined actions.

### 3. CONCLUSIONS

In our experience the cost of developing and debugging software is one of the major costs of Lattice QCD computations. FermiQCD was designed to standardize this software development process and, therefore, reduce the cost for the community.

FermiQCD comes in the form of a library (with examples), rather than a collection of ready made programs, since we understand that different research groups have different requirements. Some ready made programs for specific computations (such as generating gauge configurations, computing meson propagators, etc.) are available as examples, while others are available upon request to the authors. We provide converters for the most common data formats including UKQCD, MILC, CANOPY and various binary formats.

The FermiQCD libraries can be used freely for research purposes and we do not require that users share their programs unless they wish to do so (although we feel that this practice should be encouraged).

On one hand, FermiQCD is a mature project. The present implementation has been tested independently by different groups and it has been

used to develop large scale lattice QCD computations. On the other hand, there is a lot of work that needs to be done, including:

- implementing dynamical fermions
- developing a graphical interface
- optimizing the gauge actions
- porting the low level communications libraries (currently based on MPI) to other protocols such as TCP/IP for Gigabit ethernet and SciDAC API for the QCDOC.
- incorporating grid-like features, including the ability to read and write data in the new international data grid standard for gauge configurations.
- implementing new actions such as the Iwasaki gauge action and Overlap fermions.
- building a collection of ready-to-use programs and related documentation.

The development of FermiQCD has greatly benefited from access to other codes such as UKQCD (from which we borrowed the local random number generator), MILC (for the staggered fermion algorithms), CANOPY (for many design features), M. Lüscher's (for the assembly macros) and C. Michael's (for all-to-all propagators); we thank their authors for making them available to us. We also wish to thank J. Flynn, A. Shams, F. Mescia, L. Del Debbio and T. Rador for their through tests of the inverters and the Clover action.

### REFERENCES

1. M. Di Pierro, [hep-lat/0004007]
2. M. Di Pierro, Nucl. Phys. Proc. Suppl. **106** (2003) 1034 [hep-lat/0110116]
3. MILC collaboration (Kostas Orginos et al.) Phys. Rev. **D59** (1999) 014501 [hep-lat/9805009]
4. M. Oktay *et al.* (in these proceedings)
5. M. Lüscher, Nuc. Phys. Proc. Suppl. **106** (2002) 21 [hep-lat/0110007]

## AN ALGORITHMIC APPROACH TO QUANTUM FIELD THEORY

MASSIMO DI PIERRO

*School of Computer Science,  
 Telecommunications and Information Systems,  
 DePaul University, 243 S Wabash Ave,  
 Chicago, IL 60604, USA  
 mdipierro@cs.depaul.edu*

Received 12 September 2005

The lattice formulation provides a way to regularize, define and compute the Path Integral in a Quantum Field Theory. In this paper, we review the theoretical foundations and the most basic algorithms required to implement a typical lattice computation, including the Metropolis, the Gibbs sampling, the Minimal Residual, and the Stabilized Biconjugate inverters. The main emphasis is on gauge theories with fermions such as QCD. We also provide examples of typical results from lattice QCD computations for quantities of phenomenological interest.

*Keywords:* Lattice; quantum chromodynamics; quantum field theory.

### 1. Introduction

In this paper, we present a brief review of Lattice Quantum Field Theory (LQFT), a way to formulate a Quantum Field Theory (QFT) in algorithmic terms.<sup>a</sup> Most of this work is based on lectures given at Fermilab by the author in 2001.

QFT's are the application of quantum mechanics to fields. They form a very general class of mathematical models that reduces to quantum mechanics in the nonrelativistic limit (speed of light  $\rightarrow \infty$ ), to relativistic mechanics in the decoherence limit (Planck constant  $\rightarrow 0$ ) and to classical physics when both limits are taken.

Any QFT states that:<sup>b</sup>

- each type of elementary particle “A” is associated with a field  $\phi(x)$  so that  $|\phi(x)|^2$  represents the probability of the event “particle A is at the space-time point  $x = (x_0, \mathbf{x})$ ”;

<sup>a</sup>For an introduction to QFT see Refs. 1 and 2 and for LQFT see Refs. 3–6.

<sup>b</sup>From now on we assume units in which both the Planck constant and the speed of light are 1.

- there exists a functional of the field  $\mathcal{S}[\phi, \dots]$ , called action, which describes the dynamics of the system and any correlation amplitude between multiple time ordered events A at  $x$ , A at  $x'$ , A at  $x''$ , etc. and can be computed as

$$\langle 0 | \phi(x) \phi(x') \phi(x'') \cdots | 0 \rangle \stackrel{\text{def}}{=} \int [d\phi] \phi(x) \phi(x') \phi(x'') \cdots e^{i\mathcal{S}[\phi, \dots]}. \quad (1)$$

The square of a correlation amplitude is a regular real correlation function.

The field components  $\phi(x)$  can be scalars, complex, spinors, vectors, tensors, etc., depending on the properties of the particle A.  $x', x', x''$  are points in the space-time. The symbol  $\int [d\phi]$  indicates an integral over a Hilbert space and is known as Path Integral (PI). Each  $\phi$  in the Hilbert space is referred to as a *history* or *path* or *field configuration*.

The symbol  $\phi$  is used here as a template for a generic field and, for now, one may think of it as a real scalar field. Later  $\phi$  will be replaced by the symbol  $U$  when it represents a gauge field and by the symbol  $\psi$  when it represents a spinor, for example, a quark.

Translational invariance combined with the finite speed of light implies that  $\mathcal{S}$  can be written as the integral over the  $s$ -dimensional space-time of a local function of  $\phi$ , the Lagrangian density  $\mathcal{L}$ :

$$\mathcal{S}[\phi, \dots] = \int \mathcal{L}(\phi(x), \partial_\mu \phi(x), \dots) d^s x. \quad (2)$$

The dots indicate additional fields and/or parameters of the theory such as particle masses and coupling constants. The Lagrangian is usually a function of  $\phi(x)$ , its gradient and higher derivatives.

The naive assumption that the Hilbert space in the PI is the space of all possible distributions leads to the problem of divergences of QFT's. In order to understand and cure this problem, one has to properly define this space and provide an algorithmic way to evaluate the integral. That is the main purpose of these notes.

For some theories, such as quantum electrodynamics (QED), it is possible to perform a functional expansion of the integrand of Eq. (1) around a minimum and integrate exactly the individual terms. These terms are the Feynman diagrams. The result is an asymptotic series, the Dyson series, that can be used to approximate the PI. While this is a well defined algorithm it is very difficult to implement, it only works in some cases, and it cannot be improved to arbitrary accuracy because of the asymptotic nature of the Dyson series. The divergences that appear in this case can be subtracted by regulating the theory. For example, one can dump the Fourier modes of the fields with frequency above some cutoff  $p$ .

The perturbative expansion plays a role analogous to the Taylor expansion and it does not provide a general definition of the PI. Moreover, for some theories, the Dyson series may diverge and a different approach is required in order to define and compute Eq. (1).

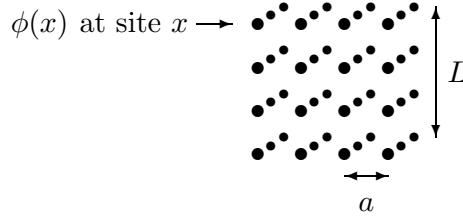
For quantum chromodynamics<sup>7,8</sup> (QCD) and other strongly coupled theories, the lattice formulation has been one of the most successful. We will refer to the latter as lattice quantum chromodynamics (LQCD).

## 2. Overview

### 2.1. Formulation

The formulation of LQFT consists of three stages.<sup>3–5</sup>

**Step 1: Discretization.** The space–time is approximated with a finite hypercubic mesh such that  $\phi(x)$  is only defined on the lattice sites  $x$ . Now the Hilbert space that constitutes the integration domain is well defined. The hypercubic mesh is characterized by the number of dimensions,  $s$ , the lattice spacing,  $a$ , the overall size,  $L = Ka$ , and the boundary conditions. Here is an example of a 3D lattice.



After discretization, for every finite  $a$  the symbol  $\int [d\phi]$  becomes a well-defined multidimensional integral

$$\int [d\phi] \simeq \int d\phi_0 \int d\phi_1 \int d\phi_2 \cdots, \quad (3)$$

where the  $i$ th degree of freedom,  $\phi_i$  is localized at some lattice site.

**Step 2: Wick rotation.** The time coordinate  $x_0$  is Wick rotated  $x_0 \rightarrow ix_0$  thus turning Eq. (1) into

$$\langle 0 | \phi(x) \phi(x') \phi(x'') \cdots | 0 \rangle_E = \int [d\phi] \phi(x) \phi(x') \phi(x'') \cdots e^{-\mathcal{S}_E[\phi]}, \quad (4)$$

where  $\mathcal{S}_E$  is now the Euclidean action. If the exponent term is real, and this is true for most physical systems of practical interest, then Eq. (4) reads as a weighted average of  $\phi(x) \phi(x') \phi(x'') \cdots$  with an exponential weight factor equal to  $\exp(-\mathcal{S}_E)$ .

Equation (4) looks like a correlation in statistical mechanics and the integral can be computed using standard numerical techniques.

**Step 3: Monte Carlo Computation.** Equation (4) is approximated by a finite sum

$$\langle 0 | \phi(x)\phi(x')\phi(x'') \cdots | 0 \rangle_E \simeq \frac{1}{N} \sum_{k=0}^{k < N} \phi_{[k]}(x)\phi_{[k]}(x')\phi_{[k]}(x'') \cdots . \quad (5)$$

Each field configuration  $\phi_{[k]}$  is generated by random sampling from a probability distribution proportional to  $\exp(-\mathcal{S}_E[\phi])$ . As more terms in the sum are considered, the right-hand side of Eq. (5) approaches the left-hand side. The numerical error in this numerical approximation can be controlled and it is usually proportional to  $N^{-\frac{1}{2}}$  where  $N$  is the number of generated configurations.

This lattice approach to the PI can be improved to any arbitrary precision by reducing the discretization step ( $a \rightarrow 0$ ), by considering a larger portion of space-time ( $L \rightarrow \infty$ ), and by generating more configurations ( $N \rightarrow \infty$ ). The extrapolation of  $a$  to zero is referred to as the *continuum limit*.

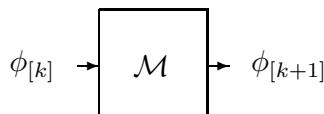
Notice that the set of Monte Carlo configurations which appear in the sum Eq. (5) does not depend on the particular correlation amplitude one is computing and therefore it can be reused for different computations as long as the system is the same (i.e. it is described by the same action). For example, in typical LQCD computations, a large numbers of configurations are generated, stored, and reused for different computations.

Because of the Wick rotation, any correlation computed using the above technique is defined in the Euclidean time, not in the Minkowski time. Some observables are unaffected by the Wick rotation and can be reliably computed, while others require an analytical continuation back to Minkowski time. Some phase information may be lost when combining this analytical continuation with the finite precision of the Monte Carlo method and only those observables that do not require analytical continuation back to the Minkowski time can be reliably extracted from the lattice.<sup>9</sup> Luckily these include the low energy spectrum and the absolute value of matrix elements of operators between on-shell states. In LQCD typical computations include the masses of bound states such as glueballs, mesons, baryons, and matrix elements between these states.

## 2.2. Algorithms

Any Monte Carlo computation can be decomposed into three elementary steps.

**Step 3.1: Markov Chain Monte Carlo (MCMC).** The MCMC is a method to generate the field configurations  $\phi$  by random sampling from a known probability distribution which, in this case, is proportional to  $\exp(-\mathcal{S}_E[\phi])$ .



The idea behind the Markov Chain is that of building a randomized iterative algorithm  $\mathcal{M}$  so that the transition probability  $P_{\mathcal{M}}(\phi'|\phi)$  of going from a configuration  $\phi$  to a configuration  $\phi'$  satisfies the following reversibility condition

$$e^{-S_E[\phi, \dots]} P_{\mathcal{M}}(\phi'|\phi) = e^{-S_E[\phi', \dots]} P_{\mathcal{M}}(\phi|\phi'). \quad (6)$$

Regardless of the starting configuration  $\phi_{[0]}$ , the succession in  $k$  is ergodic with the desired stationary distribution<sup>10</sup>  $\exp(-S_E[\phi, \dots])$ . The simplest MCMC algorithm that satisfies the reversibility condition, Eq. (6), is the Metropolis<sup>11</sup> algorithm shown in Fig. 1. This algorithm makes the next configuration in the chain by either

```

1 Algorithm: Metropolis for MCMC
2 Input:  $\phi$ , Euclidean action  $S$ 
3 Output:  $\phi'$ 
4
5 generate  $\phi'$  at random uniformly in the Hilbert space
6 generate a random number  $z$ 
7 if  $z > \exp(S(\phi) - S(\phi', \dots))$  then
8     replace  $\phi'$  with  $\phi$ 
9 return  $\phi'$ 
```

Fig. 1. Metropolis is the simplest algorithm to generate a Markov Chain Monte Carlo (MCMC). It performs a global change of the input configuration and then an accept–reject step of that change. The latter step ensures the reversibility condition and the ergodicity of the chain. Notice that every time an update is false the output configuration is the same as the input.

```

1 Algorithm: Gibbs sampling for MCMC
2 Input:  $\phi$ , euclidean gauge action  $S$ 
3 Output:  $\phi'$ 
4
5 for all lattice sites  $x$ 
6     store the field variable  $\phi(x)$  in  $\phi^{old}$ 
7     replace the value of  $\phi(x)$  with a random one (uniform in domain)
8     generate a random number  $z$ 
9     if  $z > \exp(-[\text{change in action}])$  then
10        replace  $\phi(x)$  with  $\phi^{old}$ 
11     $\phi' = \phi$ 
12 return  $\phi'$ 
```

Fig. 2. Gibbs sampling is another algorithm to generate the MCMC. It loops over all degrees of freedom of the system and, for each of them, it performs a local change and an accept–reject of the change. If the action is local, Gibbs sampling and algorithms derived from it are more efficient than the Metropolis.

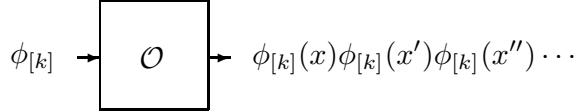
picking a copy of the preceding one, or a totally new configuration generated with uniform probability in the configurations' domain. This choice is implemented as an accept–reject step that depends on the action and guarantees that the transition probability satisfies the reversibility condition.

One algorithm derived from the Metropolis is the Gibbs sampling, Fig. 2. It uses an accept–reject similar to the Metropolis algorithms but differs because only one field variable is updated at one time. If the action is local, and this is usually true for most physical systems, the accept–reject condition is also local, and the overall algorithm is more efficient than the Metropolis in sampling the space.

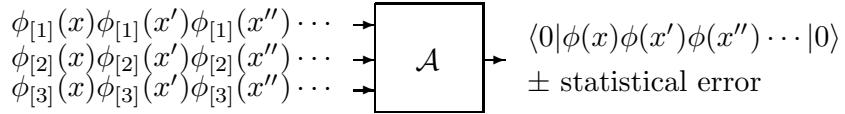
There are many other algorithms that can be used for generating the MCMC configurations and most of them are derived from either the Metropolis or the Gibbs sampling.

**Step 3.2: Measure the operator.** This algorithm measures the desired correlation, the operator  $\mathcal{O} = \phi(x)\phi(x')\phi(x'')\cdots$ , on each MCMC configuration  $\phi_{[k]}$ .

This step is nontrivial because the operator  $\mathcal{O}$  may depend on fields that do not appear in the PI, for example fermions propagating in a background gauge field. If this is true, the algorithm  $\mathcal{O}$  requires the computation of fermion propagators. We will provide more details in a later section.

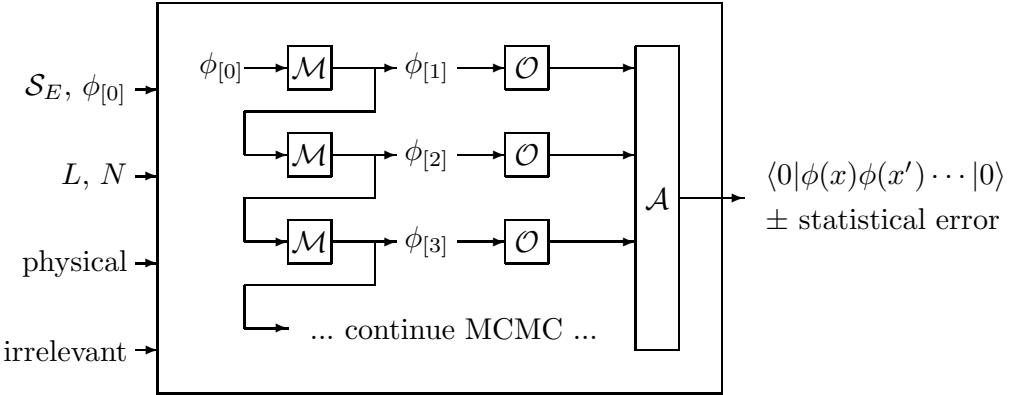


**Step 3.3: Average and Analysis.** The final result is computed by averaging the measurements of each configuration. This average is accompanied by a statistical analysis to determine the error in the result.



Estimating the statistical error is of crucial importance since it must be used as a stopping condition for the MCMC. A naive estimate of the error is given by  $\sigma/\sqrt{N}$  where  $\sigma$  is the standard deviation of the individual measurements  $\mathcal{O}_{[k]}$ . However, this estimate fails when the individual measurements are not Gaussian distributed. The standard algorithm to estimate the statistical error in an average without making any assumption about the underlying distribution is the Bootstrap algorithm.<sup>12</sup>

Every lattice computation is a combination of the above elementary steps as shown below.



Because of the way the configurations  $\phi_{[k]}$  are generated, each of them retains some memory of the preceding configurations in the chain. This correlation dies out with the distance in MCMC steps between configurations. If the evaluation of the operator  $\mathcal{O}$  on each individual configuration is more expensive than the elementary MCMC step  $\mathcal{M}$ , it may be wise to skip a number of configurations and evaluate the operator only on a subset of the total number of MCMC configurations. By skipping  $c$  configurations at each step, for  $c$  large enough,  $\phi_{[k]}$  and  $\phi_{[k+c]}$  will be sufficiently decorrelated and the procedure provides a better sampling of the integration space while saving CPU time. An empirical choice is making  $c$  larger than the maximum distance between lattice sites in lattice units,  $c > sL/a$ . In fact, if one assumes that for a local action, at each elementary MCMC step information propagates only from one lattice site to the next, after  $c$  steps, information should propagate all around the lattice, thus removing the memory of the preceding configuration.

From now on, when referring to a MCMC step, we will indicate the repetition of multiple elementary steps in order to achieve a sufficient decorrelation between effective consecutive configurations.

### 2.3. Input of the algorithms

Any lattice computation takes the following input.

- The action  $\mathcal{S}_E$ . The action determines the dynamics of the system. The same system may be represented by multiple actions that differ from each other because of irrelevant operators. These are high dimensional operators that can be added to the Lagrangian and whose contribution to the action and to the correlation amplitudes vanishes in the continuum limit. Nevertheless, their contribution affects the rate of convergence of correlations to the continuum limit and can be relevant from a numerical point of view. The art of finding these operators and determining their coefficients is called *improvement* and is based on the work of Symanzik.<sup>13</sup>

- The initial field configuration,  $\phi_{[0]}$ , from which the Markov Chain is started. The result of the computation should be independent from this choice because the MCMC loses memory of this initial configuration. One way to verify that this is true is by repeating the computation using different  $\phi_{[0]}$ .
- The lattice volume, represented by the parameter  $L$ . Ideally, one would like to perform computations close to the limit  $L \rightarrow \infty$ . In practice, this is not possible. A finite  $L$  acts as an infrared cutoff which results in systematic errors known as finite size effects. These effects can be quantified by repeating the computation at different values of  $L$ . For large  $L$  boundary conditions become irrelevant but, for finite  $L$ , they do affect the computation. The usual choice consists of periodic boundary conditions in the hypercubic lattice topology:

$$\forall x, \phi(x + L) \stackrel{\text{def}}{=} \phi(x). \quad (7)$$

For fermions often antiperiodic boundary conditions are adopted

$$\forall x, \psi(x + L) \stackrel{\text{def}}{=} -\psi(x). \quad (8)$$

For typical LQCD computations  $L \simeq 32a \simeq 2fm$  and often  $L$  is chosen larger in the time direction than in the space direction. Empirical results indicate that for  $L > 5/m_\pi$  the corresponding systematic error is less than 1%.

- The number of MCMC steps  $N$ . This number is limited by the total computing time available. For different operators,  $\mathcal{O}$ , the Monte Carlo integration converges at different rates with  $N$ . For typical LQCD computations  $N \simeq 1000$ .
- Physical parameters of the Lagrangian, such as masses of elementary particles,  $m$ , and coupling constants,  $g$ . Particles with masses  $m < 1/L$  and  $m > 1/a$  have propagators that, in Euclidean space, exhibit a correlation length larger than the lattice size  $L$  and smaller than the lattice spacing  $a$ , respectively. Elementary particles with these masses cannot be put on the lattice in a naive way. The standard approach for light fermions consists of performing the computation with nonphysical values for the masses (in the allowed region  $1/L < m < 1/a$ ) and then extrapolating the results to the physical values  $m \rightarrow \bar{m} < 1/L$ . This extrapolation is called *chiral extrapolation*. For heavy fermions the extrapolation is just one of the possible approaches and different solutions are discussed in a later section.
- Irrelevant parameters. One has the freedom to add irrelevant operators to the Lagrangian, i.e. operators whose contribution to the action vanishes in the continuum limit. Some choices for the coefficients of these operators are better than others because they improve the rate of convergence of the correlation amplitudes to the continuum limit. The values of these parameters do not affect the result of the computation but they do affect how fast one gets to the result within the required precision.

Most notably, the value of  $a$  is not a direct input of any lattice computation. All the physical input parameters (masses and coupling constants) are bare parameters

that, at constant physics, depend on the lattice spacing. Because of this implicit dependence, the choice of the coupling constant (which we will refer to as  $g$ ) is equivalent to setting the value of  $a$ . In general the relation between  $g$  and  $a$  is described by the Renormalization Group Equation (RGE).

Because the RGE is often only known perturbatively, one does not exactly know *a priori* the value of  $a$  in a lattice computation. Since every lattice quantity is computed in units of  $a$ , any error on  $a$  introduces an uncertainty in those quantities with dimension different from zero. For example the energy spectrum is in units of  $1/a$ . This problem is solved by computing ratios of masses and/or matrix elements that cancel any explicit dependence on  $a$ . An equivalent approach used in LQCD consists of measuring  $a$  by comparing the pion mass,  $m_\pi$ , computed from the lattice in lattice units with the physical pion mass and using this value to convert other quantities in physical units. We will see in the next section how this procedure is equivalent to choosing a renormalization condition.

The RGE indicates that the continuum limit  $a \rightarrow 0$  corresponds to a fixed point of the theory  $g \rightarrow g^*$  (for example for asymptotically free<sup>14</sup> theories  $g^* = 0$ ) therefore the continuum limit is realized numerically by tuning  $g$  closer to the fixed point  $g^*$ . Ideally the output of the computation should plateau as this limit is approached. In LQCD a typical example is the computation of the mass of the scalar over the vector meson for fixed bare quark masses.

#### 2.4. Regularization and the continuum limit

On one hand, the lattice provides a regularization scheme for the continuum PI. On the other hand, the lattice formulation in the continuum limit provides a definition of the continuum PI. We wish to clarify the meaning of the concept of regularization and renormalization in the lattice paradigm.

We start by making two observations.<sup>15,16</sup>

- One can never measure the value of a continuum field  $\phi(x)$  at every point in space–time. One can only measure its integral over the test function that corresponds to a physical detector, which has a finite extension. Therefore, there are mathematical reasons to require that a continuum field be defined in the space of distributions. In particular, one assumes that a particle can be localized to any arbitrary precision and that the corresponding field configuration can be a delta function  $\delta(x)$ .
- One wants to model the short distance physics by introducing a Lagrangian density that contains only local (contact) interactions. Any nontrivial Lagrangian contains terms that are products or powers of fields.

These two observations are incompatible because the product or power of fields is not a well-defined quantity when a field is a delta function. The role of regularization is that of defining this product.

There is a physical reason behind this problem: if one only knows the field via a finite size detector, with a spatial resolution limited to  $\bar{a}$ , then any field fluctuation on a scale smaller than  $\bar{a}$  should not be part of the model. This is why the model itself forces one to introduce some kind of cutoff  $a < \bar{a}$ . The effect of modes with length scale smaller than  $a$  is encoded in the value of the bare coupling constants that one puts in the model. Any change in  $a$  implies a change in the modes that contribute to the bare coupling constants and, therefore, their values have to be scaled accordingly in order to describe the same physical system.

The easiest way to regularize a distribution  $\delta(x)$  is by replacing it with some localized function with finite support, for example

$$\delta(x) \rightarrow \delta_a(x) = \frac{1}{a} [\theta(x + a/2) - \theta(x - a/2)]. \quad (9)$$

This makes the product of distributions  $\delta_a^n(x)$  well defined.

Let us consider now a 1D scalar theory defined in the interval  $[0, L]$ . Its most general correlation amplitude, defined in terms of the PI, is

$$\langle 0 | \phi(x) \phi(x') \phi(x'') \cdots | 0 \rangle \stackrel{\text{def}}{=} \int [d\phi] F[\phi(x), g], \quad (10)$$

where  $F[\phi(x), g]$  is the integrand

$$F[\phi(x), g] \stackrel{\text{def}}{=} \phi(x) \phi(x') \phi(x'') \cdots e^{-S_E[\phi, g]} \quad (11)$$

and  $g$  is a coupling constant that appears in the action. The specific form of the action,  $S_E[\phi, g]$  is unimportant but we assume that the action contains an interaction term of the form

$$g \int \phi^n(x) dx \quad (12)$$

with  $n > 2$ . This makes  $F[\phi, g]$  a nontrivial function of  $\phi$ .

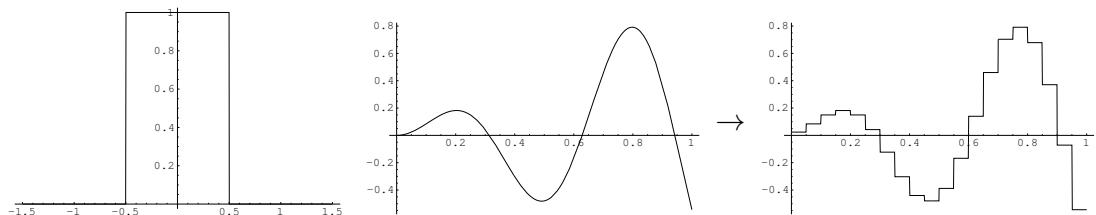


Fig. 3. (left): Example of possible regularization for the delta function. The  $x$  axis is in units of  $a$ , the  $y$  axis is in units of  $1/a$ . (center-right): Example of a continuum field configuration  $\phi$  and its approximation with a linear combinations of regularized delta functions.

For  $a = 0.1$  ( $K = 10$ ):

$$\int^{(a)} [d\phi] F[\phi, g] \stackrel{\text{def}}{=} \int d\phi_0 \cdots \int d\phi_9 F[\phi, g]$$

$$= F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=10)} \\ \text{Step function approximation} \end{array} \right] + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=10)} \\ \text{Step function approximation} \end{array} \right] + \cdots + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=10)} \\ \text{Step function approximation} \end{array} \right] + \cdots$$

$$(13)$$

For  $a = 0.05$  ( $K = 20$ ):

$$\int^{(a)} [d\phi] F[\phi, g] \stackrel{\text{def}}{=} \int d\phi_0 \cdots \int d\phi_{19} F[\phi, g]$$

$$= F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=20)} \\ \text{Step function approximation} \end{array} \right] + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=20)} \\ \text{Step function approximation} \end{array} \right] + \cdots + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=20)} \\ \text{Step function approximation} \end{array} \right] + \cdots$$

$$(14)$$

For  $a = 0.025$  ( $K = 40$ ):

$$\int^{(a)} [d\phi] F[\phi, g] \stackrel{\text{def}}{=} \int d\phi_0 \cdots \int d\phi_{39} F[\phi, g]$$

$$= F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=40)} \\ \text{Step function approximation} \end{array} \right] + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=40)} \\ \text{Step function approximation} \end{array} \right] + \cdots + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=40)} \\ \text{Step function approximation} \end{array} \right] + \cdots$$

$$(15)$$

And at the continuum limit,  $a \rightarrow 0$  ( $K \rightarrow \infty$ )

$$\int [d\phi] F[\phi, g] \stackrel{\text{def}}{=} \lim_{a \rightarrow 0} \underbrace{\int d\phi_0 \cdots \int d\phi_{K-1} F[\phi, g]}_{K \simeq L/a}$$

$$= F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=L/a)} \\ \text{Smooth function approximation} \\ \int \phi^n(x) dx \text{ is finite} \end{array} \right] + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=L/a)} \\ \text{Smooth function approximation} \\ \int \phi^n(x) dx \text{ is finite} \end{array} \right] + \cdots + F \left[ \begin{array}{c} \text{Graph of } \phi \text{ vs } x \text{ (K=L/a)} \\ \text{Smooth function approximation} \\ \int \phi^n(x) dx \rightarrow \infty \end{array} \right] + \cdots$$

$$(16)$$

Fig. 4. Example of lattice regularization of the PI.

Lattice regularization is the way to regularize the integral (10) by approximating the fields with sums of regularized delta functions:

$$\phi(x) \simeq \phi^{\text{latt}}(a, x) \stackrel{\text{def}}{=} \sum_{k=0}^{K-1} \phi_k \delta_a(x - ka), \quad (17)$$

where

$$\phi_k \stackrel{\text{def}}{=} \frac{1}{a} \int_{ka}^{ka+a} \phi(x) dx. \quad (18)$$

This is equivalent to discretizing the space-time on which the fields are defined (as shown in Fig. 3).

After discretization, for every finite  $K = L/a$ ,

$$\int^{(a)} [d\phi] F[\phi(x), g] \stackrel{\text{def}}{=} \underbrace{\int d\phi_0 \int d\phi_1 \cdots \int d\phi_{K-1}}_{K \simeq L/a} F[\phi^{\text{latt}}(a, x), g] + O(a). \quad (19)$$

Here we used the upper index  $(a)$ , which identifies the regularized PI with lattice spacing set to  $a$ .

Divergences associated with the limit  $a \rightarrow 0, K \rightarrow \infty$  (at  $L = Ka = \text{const}$ ) are called ultraviolet, while those associated with the limit  $K, L \rightarrow \infty$  (at  $a = L/K = \text{const}$ ) are called infrared.

Figure 4 shows, in a schematic way, how the path integral, Eq. (16), can be approximated by finite multidimensional integrals, Eqs. (13)–(15), and the fields are defined on a lattice. For each finite  $a$ , the “sum over the paths” (Eqs. (13)–(15)) is well defined since all divergences that may appear can be absorbed in the normalization of the integration measure and, for each path  $\phi$ , the functional  $F[\phi, g]$  is finite. In the limit  $a \rightarrow 0$  those configurations that correspond to a localized particle become more and more peaked and approach a Dirac  $\delta(x)$  function (Eq. (16)). Since the integrand  $F[\phi(x), g]$  is nonlinear, the integrand diverges on those configurations  $\phi(x) \simeq \delta(x)$ .

In order to have a well defined limit  $a \rightarrow 0, K \rightarrow \infty$  (at  $L = Ka = \text{const}$ ) one must require that the result of the regularized path integral is independent from  $a$ . The only way to do it is to making the field normalization and the coupling constant (the  $g$  of Eq. (12)) dependent on  $a$ :

$$g \rightarrow g_R(a, \Lambda) \quad (20)$$

(the constant  $\Lambda$  must be introduced because, in general,  $a$  and  $g$  do not have the same dimensions). This makes the physics independent by the lattice scale  $a$ .

One does this by choosing a particular correlation amplitude (identified by the functional integrand  $F[\phi, g]$ ) and imposing the constraint

$$\frac{d}{da} \left[ \underbrace{\int d\phi_0 \int d\phi_1 \cdots \int d\phi_{K-1}}_{K \simeq L/a} F[\phi(x), g_R(a, \Lambda)] \right] \simeq 0. \quad (21)$$

Equation (21) is the RGE for a lattice regularized theory and it determines the behavior (the running) of  $g_R(a)$ . The appearance of  $\Lambda$  is called *dimensional transmutation*.

$\Lambda$  is the typical length scale of the physics being studied. This scale is in nature and there is no freedom to change it. For QCD, for example, it is best determination is from the LQCD scaling of the coupling constant,<sup>17</sup>  $\Lambda_{\text{QCD}}^{\overline{\text{MS}}} \simeq 259(1)(20)$  MeV ( $1/\Lambda_{\text{QCD}} \simeq 1$  fm).

Notice that this procedure of defining the limit  $a \rightarrow 0$  cannot be carried out for an arbitrary theory since there may be more sources of divergence than coupling constants. If this limit can be defined the theory is said to be renormalizable, if not,  $a$  must be kept finite and the theory should be considered as an effective theory.

We distinguish between the *bare* parameters that appear in the regularized Lagrangian (for a finite value of the cutoff,  $a$ ) and the *dressed* or *renormalized* parameters that are measured by actual experiments. If one takes the limit  $a \rightarrow 0$ , the bare parameters lose any physical meaning and one must carefully define the renormalized ones (one is said to choose a prescription). If one is happy with keeping the cutoff small but finite one is allowed to identify the renormalized and the bare parameters, because these can now be measured. This approach is known as the Kadanoff–Wilson approach to renormalization.

In LQCD Eq. (21) is realized numerically. One repeats the computation of the same quantity, for example the pion mass,  $m_\pi$ , for different values of the coupling constant  $g'$ ,  $g''$ ,  $g'''$ , etc. thus obtaining  $a'\bar{m}_\pi$ ,  $a''\bar{m}_\pi$ ,  $a'''\bar{m}_\pi$ , etc. where  $\bar{m}_\pi$  is the physical pion mass. By comparing the lattice results in lattice units with the physical pion mass one can obtain the value of  $a$  that corresponds to the input values of  $g$ . From the plot one determines  $g(a)$  and identifies the fix point  $g^*$ . The same procedure can be carried on any physical quantity and it should be lead to the same running of  $g$ , up to corrections in  $a$ .

Since we will never be able to probe the physical world at every length scale, every quantum field theory should be considered an effective theory.

## 2.5. Lattice regularization versus momentum cutoff

A continuum field  $\phi$  can be expanded into Laplace components as

$$\phi(x) = \sum_{n=0}^{\infty} b_n e^{ip_n x}, \quad (22)$$

where

$$p_n \stackrel{\text{def}}{=} \frac{2\pi n}{L}; \quad b_n \stackrel{\text{def}}{=} \frac{1}{2\pi} \int_0^L \phi(x) e^{-ip_n x} dx. \quad (23)$$

Similarly a lattice field, Eq. (17) can also be expanded in Laplace components and we obtain

$$\phi^{\text{latt}}(a, x) = \sum_{n=0}^{\infty} b'_n e^{ip_n x}, \quad (24)$$

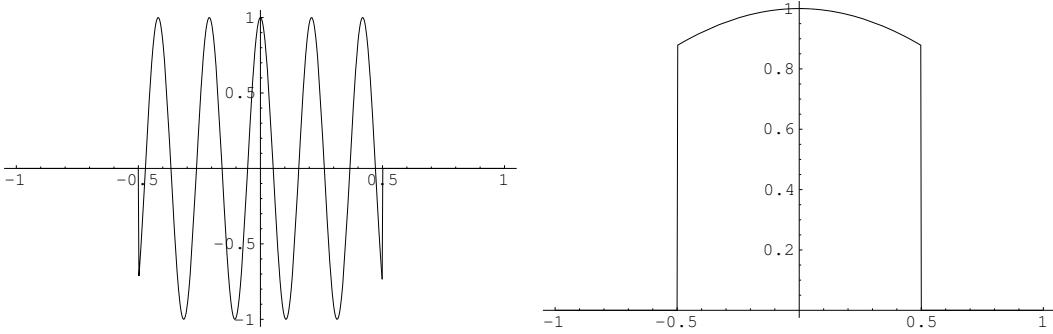


Fig. 5. Different behavior of the integrand of  $b'_n$  for high frequency modes (left) and low frequency modes (right) respectively. The  $x$  axis is in units of  $a$ .

where

$$b'_n = \frac{1}{2\pi} \sum_{k=0}^{K-1} \left[ \phi_k \int_0^L \delta_a(ka - x) e^{-ip_n x} dx \right]. \quad (25)$$

It becomes evident that for  $p_n > 1/a$  the integrand oscillates quickly and the corresponding integral,  $b'_n$ , is small; while for  $p_n < 1/a$  the integral is almost constant and approximately equal to  $e^{-ip_n ka}$ , therefore  $b'_n \simeq b_n$ . The different behaviors of the integrand are shown in Fig. 5. This proves that Eq. (22) can be written as

$$\phi(x) \simeq \phi^{\text{latt}}(a, x) \simeq \phi^{\text{co}}(a, x) \stackrel{\text{def}}{=} \sum_{n=0}^{\infty} \theta\left(\frac{1}{a} - p_n\right) b_n e^{ip_n x}. \quad (26)$$

The superscripts “latt” and “co” are used to identify the lattice and the cutoff regularization schemes, respectively.

In other words the lattice cutoff  $a$  is equivalent to a momentum cutoff  $p_{\max} < 1/\bar{a}$ , an ultraviolet cutoff. Therefore the lattice and the momentum cutoff are alternative but equivalent ways to regularize the PI.

Note that  $b_0$  is the mean value of  $\phi(x)$  and  $p_1 = 2\pi/L$  represents the minimum energy/momentum mode that can propagate on a finite unidimensional volume of length  $L$ . This is a lower limit, an infrared cutoff.

### 3. Pure Gauge Theories

#### 3.1. Action

We consider a pure gauge theory and we restrict the gauge group  $\mathcal{G}$  to U(1) and/or SU( $n$ ). In order to be able to probe the gauge field we assume to have a single particle  $\psi$  of infinite mass and unit charge that we can move on the lattice as we please.

The Aharonov–Bohm experiment<sup>18</sup> teaches us that the gauge field is not directly measurable but if we move the test particle  $\psi$  from one point  $x$  to a point  $x'$  along a path  $C$ , the test charge acquires a phase that can be measured via interference experiments

$$\psi(x') = \exp\left(ig \int_C A_\mu(x) dx^\mu\right) \psi(x), \quad (27)$$

where  $A^\mu(x)$  is the gauge field in the continuum space. On a lattice the shortest possible path is the link connecting two neighbor sites  $x$  and  $x + \hat{\mu}$  ( $+\hat{\mu}$  here indicates a positive vector in direction  $\mu$  and length equal to the lattice spacing  $a$ ) therefore the elementary lattice gauge degrees of freedom are not  $A_\mu$  but

$$U(x, \mu) \stackrel{\text{def}}{=} \exp \left( ig \int_x^{x + \hat{\mu}} A_\mu(x) dx^\mu \right) \simeq 1 + i a g A_\mu \left( x + \frac{1}{2} \hat{\mu} \right) + O(a^2). \quad (28)$$

For later convenience we also define

$$U(x, -\mu) \stackrel{\text{def}}{=} \exp \left( ig \int_{x + \hat{\mu}}^x A_\mu(x) dx^\mu \right) \simeq 1 - i a g A_\mu \left( x - \frac{1}{2} \hat{\mu} \right) + O(a^2). \quad (29)$$

From the definition it follows that  $U(x, -\mu) = U^\dagger(x - \hat{\mu}, \mu)$ .

For the rest of this section, the gauge links  $U(x, \mu)$  will replace our generic template field  $\phi(x)$ . The index  $\mu$  is an integer that labels the vector component of the gauge field.

Under a local gauge transformation  $\psi(x) \rightarrow V(x)\psi(x)$  the field  $U$  transforms as

$$U(x, \mu) \rightarrow V(x)U(x, \mu)V^{-1}(x + \hat{\mu}). \quad (30)$$

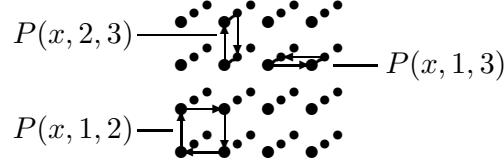
Any path  $C$  on the lattice can be identified with a set of links  $U(x_{[i]}, \mu_{[i]})$  such that

$$\exp \int_C A^\mu(x) dx_\mu \simeq \prod_i U(x_{[i]}, \mu_{[i]}). \quad (31)$$

The most local gauge invariant one can build is called the plaquette and it is a product of 4 links in a loop

$$P(x, \mu, \nu) \stackrel{\text{def}}{=} \text{Tr}[U(x, \mu)U(x + \hat{\mu}, \nu)U(x + \hat{\mu} + \hat{\nu}, -\mu)U(x + \hat{\nu}, -\nu)]. \quad (32)$$

Here are some examples of plaquettes:



The simplest lattice action  $S_E$  one can engineer using the gauge field must therefore be linear in  $P(x, \mu, \nu)$ , real, and invariant under the lattice rotational symmetry. Following the common notation, this simplest action can be written as

$$S_E^{\text{gauge}} \stackrel{\text{def}}{=} \frac{-\beta}{2n} \text{Re} \sum_{x, \mu \neq \nu} P(x, \mu, \nu), \quad (33)$$

where  $n$  is the size of the gauge group ( $n = 1$  for  $U(1)$ ).

Substituting Eqs. (28)–(32) in Eq. (33), one obtains

$$\begin{aligned}
S_E^{\text{gauge}} &\simeq \frac{-\beta}{2n} \operatorname{Re} \sum_{x,\mu\nu} \left[ 1 + iagA_\mu \left( x + \frac{1}{2}\hat{\mu} \right) \right] \left[ 1 + iagA_\mu \left( x + \frac{1}{2}\hat{\nu} \right) \right] \\
&\quad \times \left[ 1 - iagA_\mu \left( x - \frac{1}{2}\hat{\mu} \right) \right] \left[ 1 - iagA_\nu \left( x - \frac{1}{2}\hat{\nu} \right) \right] \\
&\simeq \frac{-\beta}{2n} \operatorname{Re} \sum_{x,\mu\nu} \left[ 1 + iagA_\mu - \frac{ia^2g}{2} \partial_\nu A_\mu \right] \left[ 1 + iagA_\mu + \frac{ia^2g}{2} \partial_\mu A_\nu \right] \\
&\quad \times \left[ 1 - iagA_\mu - \frac{ia^2g}{2} \partial_\nu A_\mu \right] \left[ 1 - iagA_\nu + \frac{ia^2g}{2} \partial_\mu A_\nu \right] \\
&\simeq \frac{-\beta}{2n} \operatorname{Re} \sum_{x,\mu\nu} 1 - \frac{a^4g^2}{4} (\partial_\mu A_\nu - \partial_\nu A_\mu + g[A_\mu, A_\nu])^2 \\
&\simeq c + \frac{\beta}{2n} \sum_{x,\mu\nu} \frac{a^4g^2}{4} F_{\mu\nu} F^{\mu\nu}, \tag{34}
\end{aligned}$$

where  $c$  is an overall irrelevant constant and Eq. (34) was derived from Eq. (33) via a Taylor expansion around  $x + \frac{1}{2}\mu + \frac{1}{2}\nu$ .

Notice that  $\frac{1}{4}F_{\mu\nu}(x)F^{\mu\nu}(x)$  is the ordinary kinetic term for a continuum gauge field and  $a^4 \sum_x$  is  $\int d^4x$  in the continuum limit. Following this analogy,

$$\beta = \frac{2n}{g^2} \tag{35}$$

hence  $g$  is interpreted as the bare gauge coupling constant at the lattice scale  $a$ . For non-Abelian gauge theories asymptotic freedom dictates that  $g$  goes to zero when  $a$  goes to zero.

Equation (33) is called Wilson gauge action.<sup>19</sup> The Wilson gauge action can also be derived by direct discretization of the continuum gauge action by ignoring everything but the lowest order terms in  $a$ .

For large  $\beta$  the trace of the average plaquette is close to  $n$ , perturbative effects dominate and this results in small quantum fluctuations, and small/slow changes in the configurations generated by the MCMC algorithm. For small  $\beta$  the average plaquette is close to 0, nonperturbative effects dominate which results in large/nonlocal quantum fluctuations, and relatively big changes in the configurations generated by the MCMC.

### 3.2. Algorithms

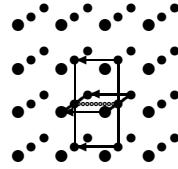
Because of the locality of the Wilson gauge action, for every link  $U(x, \mu)$  one can rewrite the action as

$$S_E^{\text{gauge}} = \frac{-\beta}{2n} \operatorname{Re} \operatorname{Tr} U(x, \mu) R(x, \mu) + \dots, \tag{36}$$

where the dots represent the sum over plaquettes that do not include  $U(x, \mu)$  and

$$\begin{aligned} R(x, \mu) = & \sum_{\nu \neq \mu} U(x + \hat{\mu}, \nu) U(x + \hat{\mu} + \hat{\nu}, -\mu) U(x + \hat{\nu}, -\nu) \\ & + U(x + \hat{\mu}, -\nu) U(x + \hat{\mu} - \hat{\nu}, -\mu) U(x - \hat{\nu}, \nu) \end{aligned} \quad (37)$$

are referred to as *staples*. A 3D projection is represented in the image below:



This makes the Gibbs sampling algorithm more efficient than the Metropolis because it is possible to change a single link  $U(x, \mu) = U^{\text{old}} \rightarrow U(x, \mu) = U^{\text{new}}$  at one time without having to recompute the entire action. In fact, the accept–reject step only depends on the variation of the action given by the first term in Eq. (36):

$$\delta S_E^{\text{gauge}} = \frac{-\beta}{2n} \text{Re Tr}[(U^{\text{new}} - U^{\text{old}})R(x, \mu)]. \quad (38)$$

There are many algorithms that are similar to the Gibbs sampling but more efficient. One of the most common is the heatbath.<sup>20</sup>

In order to make a MCMC step, whether global in the Metropolis or local in the Gibbs sampling and derived algorithms, one must be able to generate a new link at random with uniform probability in the gauge group  $\mathcal{G}$ .

For  $\mathcal{G} = \text{U}(1)$  it is sufficient to generate a uniform random number  $u \in [0, 1]$  and update

$$U(x, \mu) \rightarrow U'(x, \mu) = e^{2\pi i u}. \quad (39)$$

For  $\mathcal{G} = \text{SU}(2)$  one can use the map between  $\text{SU}(2)$  and  $\text{SO}(3)$  (the symmetry group of a 3-sphere), generate a uniform point on a 3-sphere  $(u^0, u^1, u^2, u^3)$  and map it back into  $\text{SU}(2)$  via

$$U(x, \mu) \rightarrow U'(x, \mu) = u^0 + u^1 \sigma^1 + u^2 \sigma^2 + u^3 \sigma^3. \quad (40)$$

where  $\sigma_i$  are the Pauli matrices.

For  $\mathcal{G} = \text{SU}(n)$  and  $n > 2$  there is no exact technique but a common recursive technique<sup>21</sup> consists of

$$U(x, \mu) \rightarrow U'(x, \mu) = \prod_{i < j} G_{ij}, \quad (41)$$

where  $G_{ij}$  are random  $\text{SU}(2)$  matrices that acts on the  $ij$  subgroup of  $\text{SU}(n)$ .

The general algorithm is reported in Fig. 6.

```

1 Algorithm: Generate a random  $SU(n)$  matrix
2 Input:  $n$ 
3 Output:  $A$ 
4
5 if  $n == 1$  return  $\exp(2\pi i \text{uniform}())$ 
6  $A = n \times n$  identity matrix
7 for  $i = 0$  to  $n - 2$ 
8   for  $j = i + 1$  to  $n - 1$ 
9      $\alpha = \pi \text{uniform}()$ 
10     $\phi = 2 \text{uniform}()$ 
11     $\cos(\theta) = 2 \text{uniform}() - 1$ 
12     $\sin(\theta) = \sqrt{1 - \cos^2(\theta)}$ 
13     $u^0 = \cos(\alpha)$ 
14     $u^1 = \sin(\alpha) \sin(\theta) \cos(\phi)$ 
15     $u^2 = \sin(\alpha) \sin(\theta) \sin(\phi)$ 
16     $u^3 = \sin(\alpha) \cos(\theta)$ 
17     $G = u^0 + u^1 \sigma^1 + u^2 \sigma^2 + u^3 \sigma^3$ 
18     $A' = A$ 
19    for  $k = 0$  to  $n - 1$ 
20       $A'^{ik} = G^{00} A^{ik} + G^{01} A^{jk}$ 
21       $A'^{jk} = G^{10} A^{ik} + G^{11} A^{jk}$ 
22     $A = A'$ 
23 return  $A$ .

```

Fig. 6. General algorithm for generating a random element in a U(1) and/or  $SU(n)$  gauge group with uniform distribution within the group. For  $SU(2)$  it uses the invertible map in  $SO(3)$  and for  $SU(n > 2)$  it generates the matrix as product of  $SU(2)$  subgroups. The function “uniform()” is assumed to return a uniform random number in  $(0, 1)$ .

### 3.3. Example: Quark–antiquark potential

According to Quantum Mechanics, a state consisting of one static quark and one static antiquark separated by a distance  $r$  evolves in time by acquiring a phase  $e^{iHt}$  where  $H$ , the Hamiltonian, is equal to the static potential between the quarks  $H = V(r)$ . In fact, kinetic contribution to  $H$  is zero because the quarks are static.

On a Euclidean lattice the same state evolves according to  $e^{-V(r)t}$ , therefore the potential  $V(r)$  can be determined by computing the lattice expectation value of the operator that corresponds to this system.

The quark and the antiquark have to be created at the same point, separated at distance  $r$ , evolve for a time  $t$  and then reunite and annihilate. Due to the fact that an antiquark is nothing other than a quark propagating backwards in time, the operator that corresponds to this system is the product of links around a square path of size  $r \times t$ :

$$\begin{aligned}
P^{r \times t}(x) &\stackrel{\text{def}}{=} U(x, \mu)U(x + \hat{\mu}, \mu) \cdots U(x + (r - 1)\hat{\mu}, \mu)U(x + r\hat{\mu}, \nu) \\
&\quad \times U(x + r\hat{\mu} + \hat{\nu}, \nu) \cdots U(x + r\hat{\mu} + (t - 1)\hat{\nu}, \nu)U(x + r\hat{\mu} + t\hat{\nu}, -\mu) \\
&\quad \times U(x + (r - 1)\hat{\mu} + t\hat{\nu}, -\mu) \cdots U(x + \hat{\mu} + (t - 1)\hat{\nu}, -\mu) \\
&\quad \times U(x + t\hat{\nu}, -\nu)U(x + (t - 1)\hat{\nu}, -\nu) \cdots U(x + \hat{\nu}, -\nu).
\end{aligned} \tag{42}$$

```

1 Algorithm: Compute the static quark-antiquark potential
2 Input:  $\beta$ , size of gauge group  $n$ , number of MCMC steps  $N$ 
3 Output:  $V(r)$ 
4
5 Create local array  $V(r)$  and initialize it to zero
6
7 for each lattice site  $x$ 
8   for each direction  $\mu$ 
9     set  $U(x, \mu)$  to a random element of the gauge group  $SU(n)$ 
10
11 for  $c = 1$  to  $N$ 
12   replace  $U$  with the next configuration in the MCMC (use  $\beta$ )
13   for each lattice site  $x$ 
14     for each rectangular path  $r \times t$  starting in  $x$ 
15       compute  $P^{r \times t}(x)$ , the product of links along the path
16       compute  $v = -\log(P^{r \times t}(x))/(t \times L^4)$ 
17       add  $v$  to  $V(r)$ 
18 return  $V(r)$ 

```

Fig. 7. Example of algorithm to compute the static quark–antiquark potential. Notice the role of steps 7–9 is to create the initial configuration, step 11 loops over the Markov chain, step 12 creates the next configuration in the chain (using Metropolis, Gibbs sampling or other algorithm), steps 13–16 measure the operator, and step 17 averages the results.

The static potential can therefore be determined by measuring the left-hand side of

$$\langle 0 | \sum_x P^{r \times t}(x) | 0 \rangle \propto e^{-V(r)t} \quad (43)$$

that is measuring

$$V(r) = -\frac{1}{t} \log \left( \frac{1}{N} \sum_{U,x} \text{Re Tr } P^{r \times t}(x) \right). \quad (44)$$

Figure 8 shows the result of this computation.<sup>22</sup> The open squares represent the static potential for a pure SU(3) gauge field theory in 4D ( $s = 4$ ) referred to as quenched QCD. Notice that, for long distance,  $V(r) \simeq \sigma r$  where  $\sigma$  is the string tension. The solid points in figure represent the same static potential with a gauge field coupled to two dynamical light quarks for two different values of the quark mass,  $m$ . We expect a change of regime when  $V(r) > 2m$ . In fact, according to current models of confinement,<sup>23</sup> when the static potential exceeds the energy required to create a new couple of quark and antiquark, these new particles pop up from the vacuum and screen the interaction between the original static quarks. The plot shows an indication of this phenomenon.

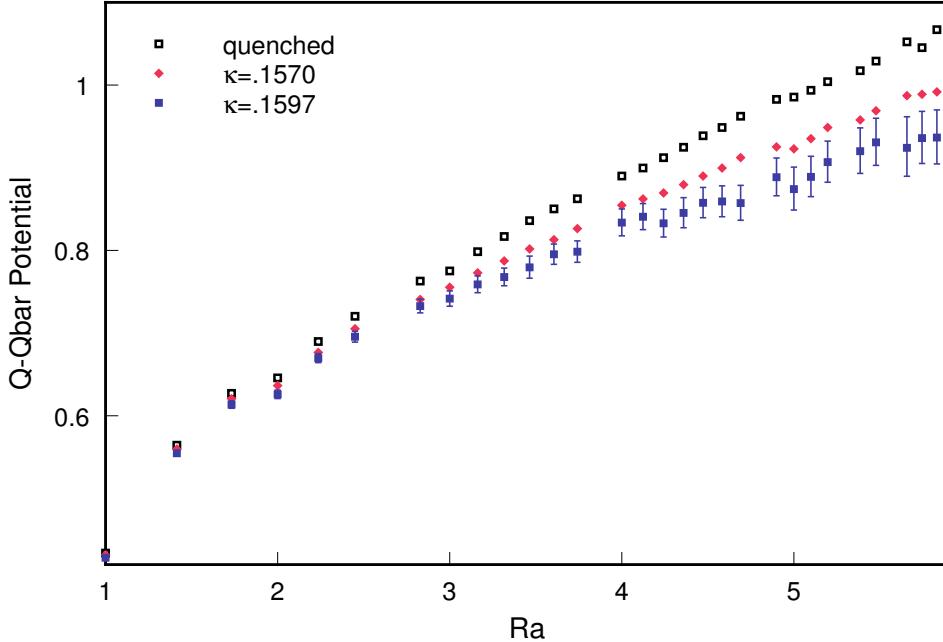


Fig. 8. The static quark potential for quenched and two flavor dynamical QCD for different masses of the sea quarks.<sup>22</sup> Results are in units of the lattice spacing.

#### 4. Fermions

Fermions, here identified by  $\psi$ , are solutions of the Dirac equation. The latter can be derived from the continuum Euclidean action

$$\mathcal{S}_E^{\text{Dirac}} = \int d^4x \bar{\psi}_\alpha^i(x) (\gamma_{\alpha\beta}^\mu D_\mu^{ij} + m\delta_{\alpha\beta}) \psi_\beta^j(x), \quad (45)$$

where  $D_\mu = \partial_\mu + igA_\mu(x)$  is the covariant derivative,  $\alpha\beta$  are spin indices and  $ij$  are gauge indices.

From now on the gauge indices  $ij$  will often be omitted and we will restrict to 4D,  $s = 4$ , since fermions are only defined for even number of dimensions.

Notice that because of the Wick rotation, gamma matrices have to be rotated too. The Euclidean gamma matrices are Hermitian ( $\gamma_\mu^\dagger = \gamma_\mu$ ) and satisfy  $\{\gamma_\mu, \gamma_\nu\} = 2\delta_{\mu\nu}$ . Two possible choices for Euclidean gamma matrices in four dimensions are listed in Appendix.

In the absence of gauge interaction,  $D_\mu = \partial_\mu$  and the simplest symmetrical discretized derivative looks like

$$\partial_\mu \psi(x) \stackrel{\text{def}}{=} \frac{1}{2} [\psi(x + \hat{\mu}) - \psi(x - \hat{\mu})]. \quad (46)$$

The most local generalization of Eq. (46) that preserves the gauge invariance of the action is

$$D_\mu \psi(x) \stackrel{\text{def}}{=} \frac{1}{2} [U(x, \mu)\psi(x + \hat{\mu}) - U^\dagger(x - \mu, \mu)\psi(x - \hat{\mu})]. \quad (47)$$

With this definition for the derivative, Eq. (45) becomes

$$\mathcal{S}_E^{\text{Dirac}} = \sum_{x,x'} \bar{\psi}_\alpha(x) Q_{\alpha\beta}(x, x') \psi_\beta(x) \quad (48)$$

and  $Q$  is called fermionic matrix

$$Q_{\alpha\beta}^{\text{naive}}(x, x') = m \delta_{x,x'} \delta_{\alpha\beta} + \sum_\mu \gamma_\alpha^\mu \frac{1}{2} [U(x, \mu) \delta_{x+\hat{\mu}, x'} - U(x, -\mu) \delta_{x-\hat{\mu}, x'}]. \quad (49)$$

There is still a problem with this naive action. The quark propagator  $S$  is defined as the inverse of the fermionic matrix  $Q$ , i.e.

$$S_{\alpha\beta}^{ij}(x, x') \stackrel{\text{def}}{=} \langle 0 | T\{\psi_\beta^j(x'), \bar{\psi}_\alpha^i(x)\} | 0 \rangle = (Q^{-1})_{\alpha\beta}^{ij}(x, x'). \quad (50)$$

In the absence of gauge field ( $U(x, \mu) = 1$ ) for a massless quark ( $m = 0$ ), for a single Fourier component of the fermionic field ( $\psi(x) = e^{ip_\mu x^\mu}$ ) the inverse propagator reads

$$S^{-1} = Q = m + i \sum_\mu \gamma^\mu \sin(p_\mu) \quad (51)$$

( $p_\mu$  is in units of  $1/a$  here). This propagator has 16 poles as opposed to the single pole at  $p = 0$  for the continuum propagator. The additional poles arise when the spatial components of  $p$  are equal to  $\pi$ .

The physical interpretation of the additional poles is that this naive discretization of the action describes 16 degenerate fermions as opposed to a single one. Different solutions to this problem lead to different implementations of lattice fermions. We consider here Wilson, clover, staggered, overlap, and domain wall fermions.

#### 4.1. Wilson fermions

Wilson proposed to remove the additional poles by giving mass to the corresponding modes.<sup>19</sup> This is done by adding a new term to the Lagrangian proportional to

$$r \bar{\psi}_\alpha(x) D_\mu D^\mu \psi_\alpha(x). \quad (52)$$

This corresponds to replacing Eq. (49) with

$$\begin{aligned} Q_{\alpha\beta}(x, x') &= (m + 4r) \delta_{x,x'} \delta_{\alpha\beta} - \frac{1}{2} \sum_\mu [(r - \gamma^\mu)_{\alpha\beta} U(x, \mu) \delta_{x+\hat{\mu}, x'} \\ &\quad - (r + \gamma^\mu)_{\alpha\beta} U(x, -\mu) \delta_{x-\hat{\mu}, x'}]. \end{aligned} \quad (53)$$

Introducing the definition

$$\kappa = \frac{1}{2m + 8r} \quad (54)$$

and scaling the field  $\psi$ , Eq. (53) can be rewritten as (omitting spin indices)

$$Q^W(x, x') = \delta_{x,x'} - \kappa \sum_\mu [(r - \gamma^\mu) U(x, \mu) \delta_{x+\hat{\mu}, x} - (r + \gamma^\mu) U(x, -\mu) \delta_{x-\hat{\mu}, x}] \quad (55)$$

which is the standard form for the Wilson fermionic matrix. Notice that any value of  $r > 0$  will do the job and one usually chooses  $r \equiv 1$ . The choice  $r = 0$  corresponds to the naive action Eq. (49).

In the Wilson fermionic action the fermion mass is traded in for the adimensional parameter  $\kappa$  defined in the asymptotically free limit by Eq. (54). In the presence of gauge interaction,  $\kappa$  is renormalized and Eq. (54) holds only approximatively. This renormalization shifts the value of  $\kappa$  that corresponds to chiral fermions ( $m = 0$ ). From now on we will identify with  $\kappa^*$  that value of  $\kappa$  that corresponds to a chiral fermion.  $\kappa^*$  is not known *a priori* but it can be determined numerically as it corresponds to a pole of  $Q^{-1}$ .

For Wilson fermions on a cold configurations,  $U \equiv 1$ , the fermion propagator reads

$$S^{-1} = Q = m + i \sum_{\mu} \gamma^{\mu} \sin(p_{\mu}) + 2r \sum_{\mu} \sin^2(p_{\mu}/2). \quad (56)$$

The main problem with Wilson fermions is that the fermionic matrix for  $m = 0$  does not anticommute with  $\gamma_5$  and therefore the action is not invariant under the global chiral symmetry

$$\psi(x) \rightarrow e^{i\gamma_5\theta} \psi(x), \quad \bar{\psi}(x) \rightarrow \bar{\psi}(x) e^{i\gamma_5\theta} \quad (57)$$

which is a symmetry of the continuum Dirac action.

The breaking of chiral symmetry invalidates chiral perturbation theory which is used to guide the extrapolation of the spectrum to the limit  $m \rightarrow 0$ . This extrapolation is crucial in order to compute correlations involving fermions with mass smaller than  $1/a$ .

## 4.2. Clover fermions

The simplest way to Symanzik improve Wilson fermions is by shifting fermions according to

$$\psi(x) \rightarrow (1 + c\gamma^{\mu}D_{\mu})\psi(x) \quad (58)$$

and tune  $c$  in order to cancel any  $O(a)$  dependence in the correlations. The effect on the action is the same as replacing<sup>24,25</sup>

$$Q^{\text{SW}}(x, x') = Q^W(x, x') - \frac{irc_{\text{SW}}}{4} \sum_{\mu \neq \nu} \gamma_{\mu} \gamma_{\nu} F_{\mu\nu}(x), \quad (59)$$

where  $F_{\mu\nu} = [D_{\mu}, D_{\nu}]$  is a lattice version of the electromagnetic tensor which, in terms of the links, can be expressed as

$$\begin{aligned} F_{\mu\nu} &= (B - B^{\dagger})/8, \\ B &= P(x, \mu, \nu) + P(x, \mu, -\nu) + P(x, -\mu, \nu) + P(x, -\mu, -\nu). \end{aligned} \quad (60)$$

Equation (48) with (59) is referred to as Sheikholeslami–Wohlert (SW) action or simply *clover* action because of the expression for  $F$ , Eq. (60).

The value of the coefficient  $c_{\text{SW}}$  does not affect the results in the continuum limit  $a \rightarrow 0$  but does affect the rate of convergence in this limit and the symmetry restoration for those symmetries broken by the lattice.

There are three standard techniques to choose the parameter  $c_{\text{SW}}$

- 1-loop improvement<sup>26</sup>

$$c_{\text{SW}} = 1 + 1.5954/\beta \quad (61)$$

- Tadpole improvement<sup>27</sup> (i.e. resumming the contribution of all tadpole graphs to the renormalization of the tree-level  $c_{\text{SW}}$ ).

$$c_{\text{SW}} = \frac{1}{u_0^3} \quad (62)$$

where  $u_0$  is the average of  $\frac{1}{3}\text{Re Tr } P(x, \mu, \nu)$  and it can be extracted from numerical simulations.

- Nonperturbative improvement.<sup>26</sup> This is the most sophisticated technique. The idea behind it is that of determining the improvement coefficients for the different operators (including  $c_{\text{SW}}$ ) by measuring independently on lattice the left and right-hand side of Ward identities and imposing the constraint that they coincide. The results for  $c_{\text{SW}}$  can be summarized by the following fitting function (valid only for  $\beta > 5.7$ )

$$c_{\text{SW}} = \frac{\beta^3 - 3.648\beta^2 - 7.254\beta + 6.642}{\beta^3 - 5.2458\beta^2}. \quad (63)$$

Even if different techniques give different results for  $c_{\text{SW}}$  they are consistent with each other provided the operators are improved by adopting the same procedure. Whichever improvement technique is used, the SW action generates correlation amplitudes that converge to the continuum limit up to correction of the second order in  $a$  and order 1 in  $g^2$  (for 1-loop) or exactly (for nonperturbative improvement).

### 4.3. Heavy fermions and Fermilab action

As mentioned previously the lattice regularization acts as infrared cutoff and prevent particles with mass  $m_Q$  higher than  $1/a$  to propagate properly. This presents a problem for the simulation of heavy quarks since in typical LQCD computations the lattice spacing is of the order of  $(2 \text{ GeV})^{-1}$ . There are four ways to implement heavy fermions on the lattice.

**Extrapolation:** Perform simulations with fermion masses lighter than the cut-off,  $m < 1/a$ , and extrapolate at the physical heavy fermion mass  $m \rightarrow \bar{m}_Q$ .

**Static fermions:** Perform simulations in the static limit,  $m_Q \rightarrow \infty$ . In this limit the Wilson action becomes the HQET<sup>30</sup> action with fermionic matrix given by

$$Q^{\text{HQET}}(x, x') = \frac{1 - \gamma^0}{2} U(x, 0) \delta_{x+\hat{0}, x'} + \frac{1 + \gamma^0}{2} U(x, -0) \delta_{x-\hat{0}, x'}. \quad (64)$$

In this limit a fermion propagator is known exactly and it is the product of links in the time direction

$$\begin{aligned} S(x, x') = & \frac{1 + \gamma^0}{2} U(x, 0) U(x + \hat{0}, 0) \cdots U(x' - \hat{0}, 0) \delta_{\mathbf{x}, \mathbf{x}'} \theta(x_0 - x'_0) \\ & + \frac{1 - \gamma^0}{2} (U(x, 0) U(x + \hat{0}, 0) \cdots U(x' - \hat{0}, 0))^{\dagger} \delta_{\mathbf{x}, \mathbf{x}'} \theta(x'_0 - x_0). \end{aligned} \quad (65)$$

Notice how static fermions require only one spin component because no term in the action mixes different spin components. Computations in the static limit are also useful to guide the extrapolation in the approach previously mentioned.

**Nonrelativistic limit:** Adopt a nonrelativistic approach (NRQCD) and perform an expansion of the fermionic action around the on-shell momentum<sup>31</sup> of propagating fermions  $p_\mu = p_\mu^{\text{on-shell}} + k_\mu$ . Nonrelativistically  $p_\mu^{\text{on-shell}} = m_Q v_\mu$  where  $v_\mu$  is the on-shell velocity of the heavy fermion and  $k_\mu$  is the off-shell momentum. In QCD,  $k_\mu$  is of the order  $\Lambda_{\text{QCD}}$ . This expansion results in the NRQCD action described by the fermionic matrix

$$Q^{\text{NRQCD}}(x, x') = i\gamma^0(D_0 + \mathbf{v} \cdot \mathbf{D} + O(1/m_Q)). \quad (66)$$

NRQCD fermions require two spin components which are mixed by  $O(q/m_Q^2)$  terms. Corrections can be taken into account systematically.

**Fermilab action:** The Fermilab action is an effective action that interpolates smoothly between the regular fermions and the static limit, and eliminates errors proportional to  $(am_Q)^n$ . This is achieved by taking Wilson fermions with the clover action and introducing different couplings for spacelike and timelike interaction terms in the Lagrangian. Moreover, in the spirit of Symanzik, higher order operators are added to the Lagrangian to cancel discretization terms that become sizable for large masses. A discussion of these operators up to dimension 4 in  $\Lambda_{\text{QCD}}/m_Q$  (HQET) and dimension 8 in  $v^\mu \gamma_\mu$  (NRQCD) can be found in Refs. 28 and 29.

#### 4.4. Staggered fermions

The Kogut–Susskind fermions, also known as staggered fermions provide an alternative to Wilson fermions.

The idea of Kogut and Susskind is that of interpreting the 16 poles of the naive fermion propagator, Eq. (51), as due to the four spin components of four different types of fermions, here referred as *flavors*, which live on a blocked lattice<sup>32–36</sup> as shown in Fig. 9.

In order to introduce staggered fermions, we will adopt the following notation.

- $x, x'$  will indicate points on the full lattice.
- $y, y'$  will indicate points on the blocked lattice.
- $z$  will label the vertices of a hypercube of side  $a$  so that each  $x$  has a unique representation as  $y + z$ .  $z_\mu \in 0, 1$  are the four-space–time components of  $z$ .

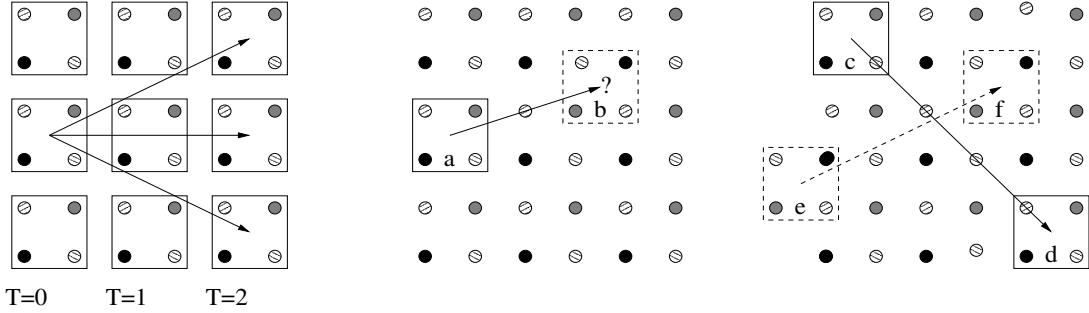


Fig. 9. Graphical representation of a 2D lattice and its blocked lattice (left). The center figure represents an example of an incorrect two-point correlation amplitude between staggered fermions. The right figures represent two correct two-point correlation amplitudes corresponding to different blockings of the same lattice.

- $\psi_{a\alpha}(y)$  will represent the four fermion flavors where  $a$  is the flavor index and  $\alpha$  is the spin index. Note that  $\psi$  is now defined on the blocked lattice.
- $\chi(x)$  will indicate the proper staggered field which corresponds to  $\psi$  but is defined on the full lattice.
- We will omit the color index since the formalism is completely transparent to it.

The map between naive lattice fermions and staggered fermions is realized by

$$\psi_{a\alpha}(y) = \frac{1}{2} \sum_z \Omega(y+z)_{a\alpha} \mathcal{P}(y, y+z) \chi(y+z), \quad (67)$$

where

$$\Omega(x) \stackrel{\text{def}}{=} \gamma_0^{x_0} \gamma_1^{x_1} \gamma_2^{x_2} \gamma_3^{x_3} \quad (68)$$

and  $\mathcal{P}(y, y+z)$  is a product of links connecting  $y$  to  $y+z$  that makes  $\psi(y)$  transform correctly under local gauge transformations. The map in Eq. (67) between  $\psi \leftrightarrow \chi$  is invertible.

The naive action of Eqs. (48) and (49) after substituting in Eq. (67) becomes

$$\begin{aligned} S_E &= \sum_y \bar{\psi}_{a\alpha}(y) (\gamma_\mu^{\alpha\beta} D_\mu + m) \psi_{a\beta}(y) \\ &= \sum_{y,z,z'} \chi^\dagger(y+z) \Omega^\dagger(y+z) \mathcal{P}(\gamma_\mu D_\mu + m) \mathcal{P} \Omega(y+z') \chi(y+z') \\ &= \sum_{x,\mu} \frac{\eta(x,\mu)}{2} [\chi^\dagger(x) U(x,\mu) \chi(x+\mu) - \chi^\dagger(x) U(x,-\mu) \chi(x-\mu)] \\ &\quad + \sum_x m |\chi(x)|^2 \\ &= \sum_{x,x'} \chi^\dagger(x) Q^{KS}(x, x') \chi(x'), \end{aligned} \quad (69)$$

where

$$\eta(x,\mu) = \frac{1}{4} \text{tr}\{\Omega^\dagger(x) \gamma_\mu \Omega(x \pm \mu)\} = (-1)^{\sum_{\nu < \mu} x_\nu} \quad (70)$$

and

$$Q^{\text{KS}}(x, x') = m\delta_{x,x'} + \frac{1}{2} \sum_{\mu} \eta(x, \mu) [U(x, \mu)\delta_{x+1,x'} - U(x, -\mu)\delta_{x-1,x'}]. \quad (71)$$

Notice that in deriving the staggered action one uses a derivative term  $D_{\mu}$  defined on the full lattice and not on the blocked lattice. This has the effect of coupling the different fermions in nontrivial ways and breaks the continuum  $SU(4)_{\text{flavor}} \times SU(4)_{\text{spin}}$  symmetry down to the discrete subgroup  $\text{SW}_4 \times \Gamma_4$  where  $\text{SW}_4$  is the hypercubic subgroup of Euclidean rotations,  $\text{SO}(4)$ , and  $\Gamma_4$  is the Clifford algebra generated by the  $\gamma$  matrices. This symmetry breaking allows us to identify the different flavors.

In fact, in the naive discretization of the action (45) the doubling problem is related to the lattice symmetry<sup>37</sup>

$$\psi(x) \rightarrow \tilde{\psi}(x) = \prod_{\mu} (i\gamma_5 \gamma_{\mu})^{z_{\mu}} e^{ix \cdot z \pi} \psi(x), \quad (72)$$

where  $z$  is in the hypercube. Equation (72) suggests that the naively discretized Eq. (45) contains interaction terms that couple one fermion mode into another by emission of a hard gluon. For staggered fermions these modes correspond to different flavors and, therefore, can be distinguished.

While Wilson fermions completely break the axial symmetry, staggered fermions break  $SU(4)_V \times SU(4)_A$  only partially and the subgroup  $U(1)_V \times U(1)_A$  remains unbroken thus preserving some form of the PCAC relations<sup>38</sup> which are important to guide the extrapolation to the chiral limit of quantities of phenomenological interest involving light fermions.

Another advantage of staggered fermions over Wilson fermions is that the inversion of the corresponding fermionic matrix  $Q$ , i.e. the computation of the fermion propagator, is about eight times faster for the same lattice size.

One disadvantage of staggered fermions is that they entangle space-time and flavor symmetries thus making it difficult to engineer physical states of definite quantum numbers.

In terms of the spinors the most general pseudoscalar meson can be written as

$$\pi_{\xi}(y) = \xi^{ab} \bar{\psi}_{\alpha,a}(y) \Gamma_{\alpha\beta} \psi_{\beta,b}(y), \quad (73)$$

where  $\Gamma$  is  $\gamma^5$  or  $\gamma^0\gamma^5$ .

For this operator to have the quantum numbers of a pion,  $\xi^{ab}$  must be an element of the **15** representation of  $SU(4)_{\text{flavor}}$ . The choice  $\xi^{ab} = \delta_{ab}$  would correspond to the singlet, the  $\eta_1$ .

Different choices of basis for the  $\xi^{ab}$  matrices are present in the literature and they are equivalent to each other up to an unitary transformation since there is only one irreducible representation of the Clifford algebra in dimension 4, the algebra of ordinary gamma matrices (associated to the group  $\Gamma_4$ ). Therefore, according with usual conventions, we choose the following basis for the  $\xi$  matrices

$$\xi_{(5)} = (\gamma^5)^*, \quad \xi_{(\mu)} = (\gamma^{\mu})^*, \quad \xi_{(\mu 5)} = (\gamma^{\mu}\gamma^5)^*, \quad \xi_{(\mu\nu)} = (\gamma^{\mu}\gamma^{\nu})^*. \quad (74)$$

Figure 9 represent a lattice, a blocked lattice, and how staggered mesons may propagate from one hypercube to another hypercube. Notice that source and destination hypercube has to be consistent with the same blocking of the lattice or the meson propagator will correspond to a nontrivial mix of the different mesons and will exhibit an oscillating behavior.

Staggered fermions with the action Eq. (69) converge quadratically to the continuum. One way to Symanzik improve this action and achieve  $O(a^2)$  improvement is by replacing each link that appears in Eq. (71) with a weighted sum of links and paths connecting the same end points as the link. The choice of the weight factors that cancels all  $O(a^2)$  effects is called ASQTAD action.<sup>39</sup>

#### 4.5. Chiral fermions: Overlap

In continuum space the massless fermionic matrix is  $Q = \sum_\mu \gamma^\mu (\partial_\mu + igA_\mu)$  and it satisfies the chirality condition

$$Q\gamma^5 + \gamma^5 Q = 0. \quad (75)$$

Therefore the eigenstates of  $P_L = (1 - \gamma^5)/2$  and  $P_R = (1 + \gamma^5)/2$  are not mixed by the action. The only term in the action that couples  $L$  and  $R$  chirality states is the mass term.

Equation (75) does not hold for  $Q^W$ ,  $Q^{SW}$  and  $Q^{KS}$  and, in fact, the Nielsen–Ninomiya theorem<sup>40</sup> states that it is not possible to define a local, translationally invariant, Hermitian  $Q$  that preserves chiral symmetry and does not cause doublers. Nevertheless, one may look for a fermion formulation on the lattice that exhibit chiral symmetry and no doublers by requiring that chiral symmetry holds for on-shell states only. For on-shell states chiral transformation can be written as

$$\psi \rightarrow e^{i\theta\gamma^5(1-aQ/2)}\psi, \quad \bar{\psi} \rightarrow \bar{\psi}e^{i\theta\gamma^5(1-aQ/2)}. \quad (76)$$

In fact on shell  $Q\psi = 0$  and Eq. (76) becomes the ordinary chiral symmetry transformation. Requiring that the fermionic matrix  $Q$  be invariant under infinitesimal transformation leads to the Ginsparg–Wilson relation

$$Q\gamma^5 + \gamma^5 Q = aQ\gamma^5 Q. \quad (77)$$

Neuberger<sup>41,42</sup> proposed the first fermionic matrix  $Q$  that satisfies the Ginsparg–Wilson relation and thus provides exact chiral symmetry on the lattice

$$Q^{\text{Overlap}} = 1 + \gamma^5 \text{sign}[\gamma^5(Q^W - 1)]. \quad (78)$$

This formulation is referred to as overlap fermions.

Since any Hermitian matrix  $\Sigma$  can be written as  $\Sigma = \Lambda D \Lambda^\dagger$  where  $D$  is a diagonal matrix, one can define any function  $f$  of a Hermitian matrix via

$$f(\Sigma) = \Lambda f(D) \Lambda^\dagger \quad (79)$$

and  $f(D)$  is a diagonal matrix with diagonal elements given by  $f(D)_{ii} = f(D_{ii})$ . The argument of sign function in the definition of the overlap fermionic matrix is Hermitian therefore Eq. (78) is well defined.

The sign function can be approximated by polynomials and, in general, it is more difficult to deal with than Wilson, clover or even staggered fermions. All numerical techniques<sup>47</sup> for calculating a  $\det Q^{\text{Overlap}}$  as required for dynamical fermions and for calculating  $(Q^{\text{Overlap}})^{-1}$  can be viewed as different ways of approximating Eq. (78).

#### 4.6. Chiral fermions: Domain wall

An alternative way to implement lattice chiral fermions was developed by Kaplan<sup>43</sup> in 1992. It is referred to as using Domain Wall (DW) fermions.<sup>44–47</sup> This formulation consists of replacing each fermion  $\psi_\alpha(x)$  with multiple fermions labeled by a new index  $k$ ,

$$\psi_\alpha(x) \rightarrow \Psi_{\alpha,k}(x). \quad (80)$$

The DW action is

$$\mathcal{S}_E^{\text{DW}} = \sum_{x,k} \bar{\Psi}_{\alpha,k}(x) Q_{\alpha\beta,kk'}^{\text{DW}}(x, x') \Psi_{\beta,k'}(x'), \quad (81)$$

where

$$\begin{aligned} Q_{\alpha\beta,kk'}^{\text{DW}}(x, x') &= Q_{\alpha\beta}^{\text{W}}(x, x') \delta_{kk'} \\ &+ (P_L)_{\alpha\beta} \theta(k < N_5 - 1) \delta_{k+1,k'} + (P_R)_{\alpha\beta} \theta(k > 0) \delta_{k-1,k'} \\ &+ (P_L)_{\alpha\beta} \delta_{k,N_5-1} \delta_{k',0} + (P_R)_{\alpha\beta} \delta_{k,0} \delta_{k',N_5-1}. \end{aligned} \quad (82)$$

$k_5$  and  $N_5$  are parameters of the model. The index  $k$  runs from 0 to  $N_5 - 1$  and it is usually interpreted as a fifth dimension of the system. Notice that the gauge field that appears in  $Q^{\text{W}}$  is independent on this fifth dimension.

The computation of  $\psi' = Q^{-1}\psi$  for a regular four-dimensional input fermionic field  $\psi$  is performed in three steps.

- The input field  $\psi$  is mapped into the five-dimensional DW field  $\Psi$ .  $L$  components are mapped into the  $k = 0$  slice and  $R$  components are mapped into the  $k = N_5 - 1$  slice:

$$\Psi_{\alpha,0}(x) = (P_L)_{\alpha\beta} \psi_\beta(x), \quad (83)$$

$$\Psi_{\alpha,k}(x) = 0 \quad \text{for } 0 < k < N_5 - 1, \quad (84)$$

$$\Psi_{\alpha,N_5-1} = (P_R)_{\alpha\beta} \psi_\beta(x). \quad (85)$$

- The DW fermionic matrix, Eq. (82), is inverted numerically using one of the algorithms explained later:

$$\Psi' = (Q^{\text{DW}})^{-1} \Psi. \quad (86)$$

- The output DW field  $\Psi'$  is mapped back into the output four-dimensional field  $\psi'$  by

$$\psi'_\alpha(x) = (P_L)_{\alpha\beta} \Psi'_{\beta,0}(x) + (P_R)_{\alpha\beta} \Psi'_{\beta,N_5-1}(x). \quad (87)$$

The effect of the DW action is that of projecting the  $L$  modes of the 4D fermion to one wall and the  $R$  modes to the other wall, thus different chiralities are mixed only by the explicit mass term in  $Q^{\text{DW}}$ , Eq. (82). The use of the Wilson action in each slice of the extra dimension  $k$  guarantees that no doublers are present.

It is possible to translate the DW fermions into an equivalent four-dimensional approximation for the Neuberger operator,<sup>48</sup> Eq. (78) and therefore  $Q^{\text{Overlap}}$  is a local operator. In fact, any rational polynomial approximation of the overlap operator is equivalent to a domain wall formulation with a finite fifth dimension.<sup>49</sup>

#### 4.7. Quenched and dynamical fermions

Including fermionic field variables in the MCMC configurations is not practical. The usual technique to deal with fermions is integrate them out analytically so that fermions do not appear in the PI measure.

Let us consider a system consisting of a gauge field coupled to  $n_f$  degenerate fermions

$$\begin{aligned} \mathcal{S}_E &= \mathcal{S}_E^{\text{gauge}} + n_f \mathcal{S}_E^{\text{fermi}} \\ &= \frac{-\beta}{2n} \sum_{x,\mu \neq \nu} \text{Re Tr } P(x, \mu, \nu) + n_f \sum_{x,y} \psi_\alpha(x) Q_{\alpha\beta}(x, y) \psi_\beta(y). \end{aligned} \quad (88)$$

For a typical correlation amplitude, fermions can be integrated out as follows:

$$\langle 0 | \cdots \psi_\alpha(x) \bar{\psi}_\beta(x') \cdots | 0 \rangle = \int [dU] [d\psi] \cdots \psi_\alpha(x) \bar{\psi}_\beta(x') \cdots e^{-\mathcal{S}_E^{\text{gauge}} - n_f \mathcal{S}_E^{\text{fermi}}} \quad (89)$$

$$= \int [dU] \cdots Q_{\alpha\beta}^{-1}(x, x') \cdots e^{-\mathcal{S}_E^{\text{gauge}} + n_f \log \det Q} \quad (90)$$

$$= \frac{1}{N} \sum_U \cdots Q_{\alpha\beta}^{-1}(x, x') \cdots \quad (91)$$

where the field configurations are now generated at random by sampling from a probability distribution proportional to  $\exp(-\mathcal{S}_E^{\text{full}})$  with

$$\mathcal{S}_E^{\text{full}} = \mathcal{S}_E^{\text{gauge}} - n_f \log \det Q. \quad (92)$$

In case the correlation amplitudes involve multiple possible Wick contractions, one has to sum over all possible Wick contractions:

$$\begin{aligned} &\langle 0 | \cdots \psi_\alpha(x) \bar{\psi}_\beta(x') \cdots \psi_\gamma(x'') \bar{\psi}_\delta(x''') | 0 \rangle \\ &= \frac{1}{N} \sum_U \cdots Q_{\alpha\beta}^{-1}(x, x') \cdots Q_{\gamma\delta}^{-1}(x'', x''') \cdots + \cdots Q_{\alpha\delta}^{-1}(x, x''') \cdots Q_{\gamma\beta}^{-1}(x'', x') \cdots. \end{aligned} \quad (93)$$

For staggered fermions, since  $Q^{\text{KS}}$  represents four degenerate flavors as opposed to a single one, therefore Eq. (92) must be replaced by

$$\mathcal{S}_E^{\text{full KS}} = \mathcal{S}_E^{\text{gauge}} - \frac{n_f}{4} \log \det Q^{\text{KS}}. \quad (94)$$

It is still debated whether the above procedure is correct, since there is no known local operator that corresponds to the fourth root of  $Q^{\text{KS}}$ . Anyway, there are indications from numerical studies<sup>50</sup> in 2D that the following relation may hold in 4D:

$$(\det Q^{\text{KS}})^{1/4} = \det Q^{\text{Overlap}} + O(a^2) \quad (95)$$

which would provide a solid theoretical justification for Eq. (94).

From the definition it is evident that  $Q$  is a sparse matrix. For Wilson fermions its dimensions are  $M \times M$  and  $M = 8n(L/a)^4$  (real and imaginary part  $\times$  4 spin components  $\times n$  color components  $\times$  number of lattice sites);  $Q$  has diagonal elements set to 1 and has only 4 other elements different from zero for each row and column (corresponding to  $+\mu$  and  $-\mu$ ). For staggered fermions  $Q^{\text{KS}}$  is 16 times smaller because it has no spin indices and this makes its inversion much faster. For domain wall fermions  $Q^{\text{DW}}$  is  $N_5^2$  larger than  $Q^{\text{W}}$  thus making the inversion of domain wall fermions much slower than for Wilson fermions.

One approximation that has been used and abused consists of setting  $n_f = 0$  in the full action. This approximation is called *quenching*. It has the effect of ignoring second quantization for fermions. This is equivalent to, in a perturbative language, ignoring fermion loops. Quenching introduces unknown systematic errors in the computation and its only justification is that the computation of  $\det Q$  is nonlocal therefore generating the Markov Chain with  $n_f \neq 0$  is very computing intensive.

For some quantities such as the static quark potential, Fig. 8, the effect of quenching is very small. For other quantities such as the light spectrum of QCD, its effect is sizable, Fig. 16. Quenching also affects the behavior of the spectrum in the chiral limit as explained in Ref. 51.

The contributions  $\det Q$  to the action is referred to as *dynamical fermions* or *sea quarks* in the context of LQCD.

#### 4.8. *CPTH theorem on the lattice*

Lattice correlation amplitudes computed using the action in Eq. (88) are invariant under charge conjugation,  $C$ , parity,  $P$ , time reversal,  $T$ , and a new symmetry,  $H$ . These symmetries apply to the measurement of an operator on each gauge configuration  $U$ .

It is useful to write how these symmetries affect a fermion propagator  $S = Q^{-1}$  and make explicit the dependence on all indices and on the gauge configuration  $U$ .

- **Charge conjugation,  $C$ :**

$$S_{\alpha\beta}^{ij}(x, y, U) = (\gamma^0 \gamma^2)_{\alpha\alpha'} S_{\alpha'\beta'}^{ji}(y, x, U^C) (\gamma^2 \gamma^0)_{\beta'\beta} \quad (96)$$

- **Parity,  $P$ :**

$$S_{\alpha\beta}^{ij}(x, y, U) = \gamma_{\alpha\alpha'}^0 S_{\alpha'\beta'}^{ij}(x^P, y^P, U^P) \gamma_{\beta'\beta}^0 \quad (97)$$

- **Time reversal,  $T$ :**

$$S_{\alpha\beta}^{ij}(x, y, U) = (\gamma^0 \gamma^5)_{\alpha\alpha'} S_{\alpha'\beta'}^{ij}(x^T, y^T, U^T) (\gamma^5 \gamma^0)_{\beta'\beta} \quad (98)$$

- **$H$  symmetry:**

$$S_{\alpha\beta}^{ij}(x, y, U) = \gamma_{\beta\alpha'}^5 S_{\alpha'\beta'}^{ji}(y, x, U) \gamma_{\beta'\alpha}^5 \quad (99)$$

$U^P, U^C, U^T$  are the parity reversed, charge conjugate, and time reversed gauge configurations respectively.

#### 4.9. Inversion algorithms

One way to invert the matrix  $Q$  is by using a stochastic technique.

In fact for any Hermitian positive definite matrix  $\Sigma$  the following exact relation holds:

$$\Sigma_{ij}^{-1} = \frac{1}{Z} \int d\phi_0 \cdots d\phi_{n-1} \phi_j^* \phi_i e^{-\phi_n^* \Sigma_{nm} \phi_m} \quad (100)$$

And substituting in the above equation  $\Sigma = Q^\dagger Q$  and multiplying both terms by  $Q^\dagger$  one obtains

$$(Q^{-1})_{\alpha\beta}^{ij}(x, x') = \frac{1}{Z} \int [d\phi] (Q\phi)_\beta^{j*}(x') \phi_\alpha^i(x) e^{-\phi^\dagger(Q^\dagger Q)\phi}. \quad (101)$$

This multidimensional integral can be computed via Monte Carlo. The field  $\phi$  in this context is usually referred to as pseudofermionic field. The stochastic computation of the full inverse propagator must be done for each gauge configuration and therefore it is computationally expensive. Various techniques for reducing the variance in the integration and reduce the statistical noise have been proposed by many authors.<sup>52,53</sup>

In most cases one does not need the full inverse  $Q^{-1}$  and it suffices to solve in  $\psi'$  the equation

$$Q\psi' = \psi \leftrightarrow \psi' = Q^{-1}\psi \quad (102)$$

for a small set of given input vectors  $\psi$ s. This inversion can be performed by minimizing the norm of the residual vector

$$r = \psi - Q\psi'. \quad (103)$$

The two algorithms commonly used for performing this numerical minimization are the Minimal Residual, Fig. 10 and Stabilized Biconjugate Gradient Fig. 11.<sup>c</sup>

<sup>c</sup>In writing fermionic algorithms we adopted the following notation:

$$\begin{aligned} \psi' \cdot \psi &\rightarrow \sum_{x,\alpha,i} \psi'^{i*}_\alpha(x) = \psi_\alpha^i(x), \\ |\psi|^2 &\rightarrow \sum_{x,\alpha,i} \psi_\alpha^i(x) = \psi_\alpha^i(x), \\ \psi' = Q\psi &\rightarrow \psi'^{i*}_\alpha(x) = \sum_{y,\beta,j} Q_{\alpha\beta}^{ij}(x, y) \psi_\beta^j(y). \end{aligned} \quad (104)$$

```

1 Algorithm: Minimal Residual Inverter
2 Input:  $\psi, Q$ 
3 Output:  $\psi'$ 
4 Temporary fields:  $q, r$ 
5
6  $r = \psi - Q\psi$ 
7  $\psi' = \psi$ 
8 do
9    $q = Qr$ 
10   $\alpha = (q \cdot r) / (q \cdot q)$ 
11   $\psi' = \psi' + \alpha r$ 
12   $r = r - \alpha q$ 
13  residue =  $r \cdot r$ 
14 while residue > precision

```

Fig. 10. Minimal Residual, a numerical algorithm to compute  $\psi' = Q^{-1}\psi$  by minimizing  $Q\psi' = \psi$ .

```

1 Algorithm: Stabilized Biconjugate Gradient
2 Input:  $\psi, Q$ 
3 Output:  $\psi'$ 
4 Temporary fields:  $p, q, r, s, t$ 
5
6  $\psi' = \psi$ 
7  $r = \psi - Q\psi$ 
8  $q = r$ 
9  $p = 0$  (zero field)
10  $s = 0$  (zero field)
11  $\rho = \rho' = \alpha = \omega = 1$ 
12 do
13    $\rho = q \cdot r$ 
14    $\beta = (\rho/\rho')(\alpha/\omega)$ 
15    $\rho' = \rho$ 
16    $p = r + \beta p - \beta\omega s$ 
17    $s = Qp$ 
18    $\alpha = \rho/(q \cdot s)$ 
19    $r = r - \alpha s$ 
20    $t = Qr$ 
21    $\omega = (t \cdot r)/(t \cdot t)$ 
22    $\psi' = \psi' + \omega r + \alpha p$ 
23   residue =  $r \cdot r$ 
24 while residue > precision

```

Fig. 11. Stabilized Biconjugate Gradient, another numerical algorithm to compute  $\psi' = Q^{-1}\psi$  by minimizing  $Q\psi' = \psi$ .

#### 4.10. Dynamical fermions algorithms

For any Hermitian matrix  $\Sigma$ ,

$$\det \Sigma = \frac{1}{Z} \int d\phi_0 \cdots d\phi_{n-1} e^{-\phi_n^*(\Sigma^{-1})_{nm}\phi_m}. \quad (105)$$

Thus for two degenerate flavors the contribution to the action due to fermions is

$$(\det Q)^2 = \det Q^\dagger Q = \frac{1}{Z} \int [d\phi] e^{-\phi^\dagger(Q^\dagger Q)^{-1}\phi}, \quad (106)$$

where  $\phi$  are pseudobosonic fields. Equation (106) can be evaluated numerically using Monte Carlo.

Various techniques based on the above equation have been developed to speed up the MCMC and to take into account odd numbers of dynamical quarks.

The most common MCMC algorithm for dynamical fermions is the Hybrid Monte Carlo algorithm discussed in Refs. 54 and 55.

The technique used in Ref. 22 consists of observing that  $\det Q^\dagger Q = (\det \gamma^5 Q)^2$  where  $\gamma^5 Q$  is Hermitian, and approximating the determinant with a truncated determinant, defined as the product of the eigenvalues of  $\gamma^5 Q$  below some cutoff. The eigenvalues are computed by diagonalizing  $\gamma^5 Q$  using the Lanczos algorithm.

One of the most promising techniques in terms of efficiency for light fermion mass is a version of the Hybrid Monte Carlo based on the Schwartz Alternating Procedure discussed in Refs. 56 and 57.

Theoretical work on algorithms for dynamical overlap fermions are proposed in Refs. 58 and 59. Some preliminary phenomenological results can be found in Ref. 60. Algorithms for dynamical domain wall fermions are discussed in Refs. 61 and 62.

#### 4.11. Example: Pion mass and decay constant

We consider here, as an example, the computation of the mass of the pion,  $m_\pi$ , and the pion decay constant,  $f_\pi$ , for a  $SU(n)$  gauge theory (for  $n = 3$  this is QCD) as defined by the action in Eq. (92).

In order to be able to put a pion on the lattice one needs to have a definition of the former, i.e. one needs to have an operator function of the gauge and quark fields that has the pion as lowest eigenstate.

The pion is the lightest pseudoscalar in the theory and the simplest operator that transforms as a pseudoscalar under rotations is

$$\Psi_\pi(x) \stackrel{\text{def}}{=} \sum_{\alpha, \beta} \bar{\psi}(x)_\alpha \gamma_5^{\alpha\beta} \psi_\beta(x). \quad (107)$$

```

1 Algorithm: Compute the static quark-antiquark potential
2 Input:  $\beta$ ,  $\kappa$ , size of gauge group  $n$ , number of MCMC steps  $N$ 
3 Output:  $C^\pi(t)$ 
4
5 Create local array  $C^\pi(t)$  and initialize it to zero
6
7 for each lattice site  $x$ 
8   for each direction  $\mu$ 
9     set  $U(x, \mu)$  to a random element of the gauge group  $SU(n)$ 
10
11 for  $c = 1$  to  $N$ 
12   replace  $U$  with the next configuration in the MCMC
13   for each spin component  $a$ 
14     for each color component  $i$ 
15       make a field  $\psi(x) = 0$ 
16       for each lattice site  $x$  on timeslice  $x^0 = 0$ 
17         set  $\psi^{\alpha,i}(x = 0) = 1$ 
18         compute  $\psi' = Q^{-1}\psi$ 
19         for each lattice site  $x$ 
20           add  $|\psi'(x)|^2$  to  $C^\pi(t = x^0)$ 
21 return  $C^\pi(t)$ 

```

Fig. 12. Example of algorithm to compute a pion propagator. Notice the role of steps 7–9 is to create the initial configuration, step 11 loops over the Markov chain, step 12 creates the next configuration in the chain (using Metropolis, Gibbs sampling or other algorithm), steps 14–18 measure the operator, and steps 19–20 average the results separately for each lattice time-slice.

We identify with  $|E_k\rangle$  its eigenstates and  $E_k$  the corresponding ordered eigenvalues so that, by definition,  $E_0$  is  $m_\pi$  and  $|E_0\rangle$  is  $|\pi\rangle$ . A quantum mechanical computation shows that

$$C_\pi(y_0 - x_0) \stackrel{\text{def}}{=} FT_{\mathbf{x}}^0 FT_{\mathbf{y}}^0 \langle 0 | \Psi(x) \Psi^\dagger(y) | 0 \rangle \quad (108)$$

$$\begin{aligned} &= \sum_k FT_{\mathbf{x}}^0 FT_{\mathbf{y}}^0 \langle 0 | \Psi(x) | E_k \rangle \frac{1}{2E_k} \langle E_k | \Psi^\dagger(y) | 0 \rangle \\ &= \sum_k \frac{|\langle 0 | \Psi(0) | E_k \rangle|^2}{2E_k} e^{iE_k x_0} e^{-iE_k y_0} \\ &= \frac{|\langle 0 | \Psi(0) | \pi \rangle|^2}{2m_\pi} e^{-im_\pi(y_0 - x_0)} + \dots \\ &= \frac{f_\pi^2 m_\pi}{2} e^{-im_\pi(y_0 - x_0)} + \dots \end{aligned} \quad (109)$$

Here  $FT_{\mathbf{x}}^0 = \sum_{\mathbf{x}}$  is the Fourier transform in the spatial components of  $x$  at zero momentum. The dots indicate exponential terms that oscillate faster than the leading term. The last step follows from the definition of  $f_\pi$ , the pion decay constant.

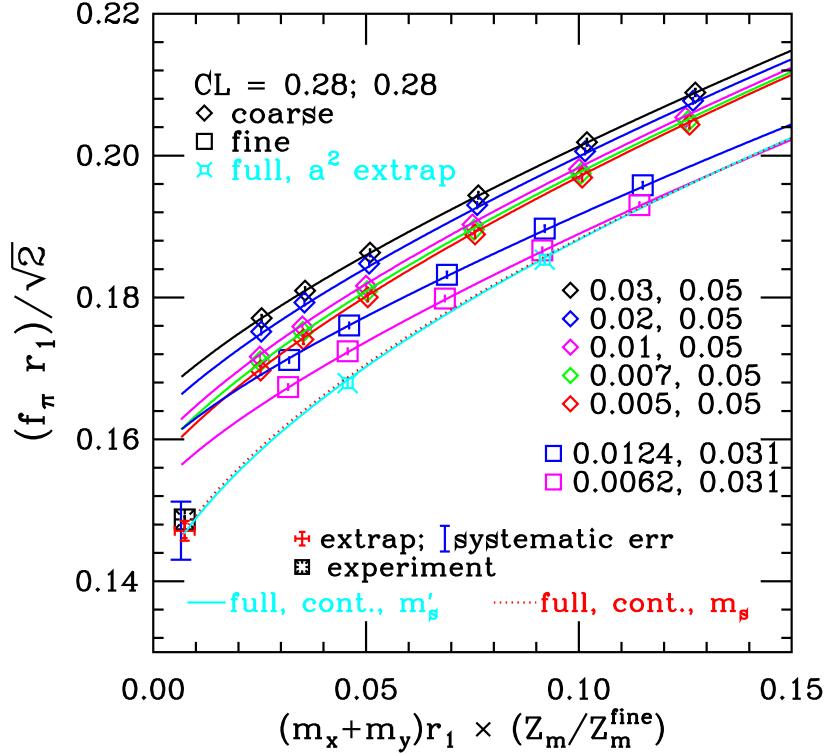


Fig. 13. Global fit of partially quenched data from the MILC collaboration<sup>63</sup> and chiral extrapolation of  $f_\pi$  for various input values of the propagating and dynamical quarks' masses. Here  $m_x$  and  $m_y$  represents the bare  $u$  and  $d$  quark masses respectively.

After a Wick rotation

$$C_\pi(y_0 - x_0) = \frac{f_\pi^2 m_\pi}{2} e^{-m_\pi(y_0 - x_0)} + \dots \quad (110)$$

and the fast oscillating terms, represented by the dots, are replaced by fast decaying exponentials. For  $y_0 - x_0 = t$  and considering periodic boundary conditions

$$C_\pi(t) = \frac{f_\pi^2 m_\pi}{2} [e^{-m_\pi(t)} + e^{-m_\pi(L-t)}] + \dots \quad (111)$$

The lattice formulation of QCD provides the numerical technique to evaluate the left-hand side of Eq. (109):

$$\begin{aligned} C_\pi(t) &= FT_{\mathbf{x}}^0 FT_{\mathbf{y}}^0 \int [d\phi][dA] \Psi^\dagger(x) \Psi(y) e^{-S_E} \\ &\simeq \frac{1}{N} \sum_U FT_{\mathbf{x}}^0 FT_{\mathbf{y}}^0 \text{Re Tr}(\gamma^5 Q^{-1}(x, y) \gamma^5 Q^{-1}(y, x)) \\ &= \frac{1}{N} \sum_U \sum_x \sum_y |Q^{-1}(x, y)|^2 \delta(t - |y_0 - x_0|). \end{aligned} \quad (112)$$

In the last step we used  $H$  symmetry on the second propagator.

State	$I^G$	$J^{PC}$	Operator $\Psi$
scalar	$1^-$	$0^{++}$	$\bar{\psi}\psi'$
	$1^-$	$0^{++}$	$\bar{\psi}\gamma^0\psi'$
pseudoscalar	$1^-$	$0^{-+}$	$\bar{\psi}\gamma^5\psi'$
	$1^-$	$0^{-+}$	$\bar{\psi}\gamma^0\gamma^5\psi'$
vector	$1^+$	$1^{--}$	$\bar{\psi}\gamma^\mu\psi'$
	$1^+$	$1^{--}$	$\bar{\psi}\gamma^\mu\gamma^0\psi'$
axial	$1^-$	$1^{++}$	$\bar{\psi}\gamma^\mu\gamma^5\psi'$
tensor	$1^+$	$1^{+-}$	$\bar{\psi}\gamma^\mu\gamma^j\psi'$
octet	$\frac{1}{2}$	$\frac{1}{2}^-$	$(\psi^{Ti}\gamma^2\gamma^0\psi'^j)(\gamma^5\psi''^k)\varepsilon_{ijk}$
	$\frac{1}{2}$	$\frac{1}{2}^-$	$(\psi^{Ti}\gamma^2\gamma^0\gamma^5\psi'^j)(\psi''^k)\varepsilon_{ijk}$
decuplet	$\frac{3}{2}$	$\frac{3}{2}^+$	$(\psi^{Ti}\gamma^2\gamma^0\gamma^i\psi'^j)(\psi''^k)\varepsilon_{ijk}$

Fig. 14. Example of currents used on lattice and their relative quantum numbers.  $\psi$ ,  $\psi'$  and  $\psi''$  are different flavors. The superscripts  $i$ ,  $j$  and  $k$  are gauge indices.

By computing Eq. (112) for different values of  $t$  and fitting the results with Eq. (111), one can extract both  $m_\pi$  and  $f_\pi$ . Some numerical results<sup>64</sup> for  $f_\pi$  computed from Eq. (112) for different values of the quark masses are shown in Fig. 15. The extrapolated  $f_\pi$  is compared with the experimental results.

In practice, because of dimensional transmutation, one always computes pure numbers such as  $m_\pi$  in units of  $1/a$  and one can eliminate such dependence on  $a$  by computing ratios of masses or other dimensionless ratios.

Similarly one can compute masses and decay constants of other particles by choosing the right operator. A list of operators for various interesting states is listed in Fig. 14. For a deeper analysis on how to built this type of operators for baryons can be found in Ref. 65.

Figure 15 shows the computation of the neutron mass for different fermion formulations at different lattice spacing.<sup>66</sup> As expected, within error, they all agree with each other in the continuum limit.

## 5. Error Analysis

There are two types of errors in any LQCD computation. Statistical errors and systematic errors. Statistical errors are under control today. The main source of systematic error are discussed below, they are being addressed by recent computations, and will be removed in the near future.

- Discretization. The typical lattice spacing is today of the order of  $(2 \text{ GeV})^{-1}$ . This introduces discretization errors that are sometime difficult to quantify. One effect, for example, is the breaking of continuous rotational symmetry. One way to reduce discretization effects is by means of Symazik improved actions.

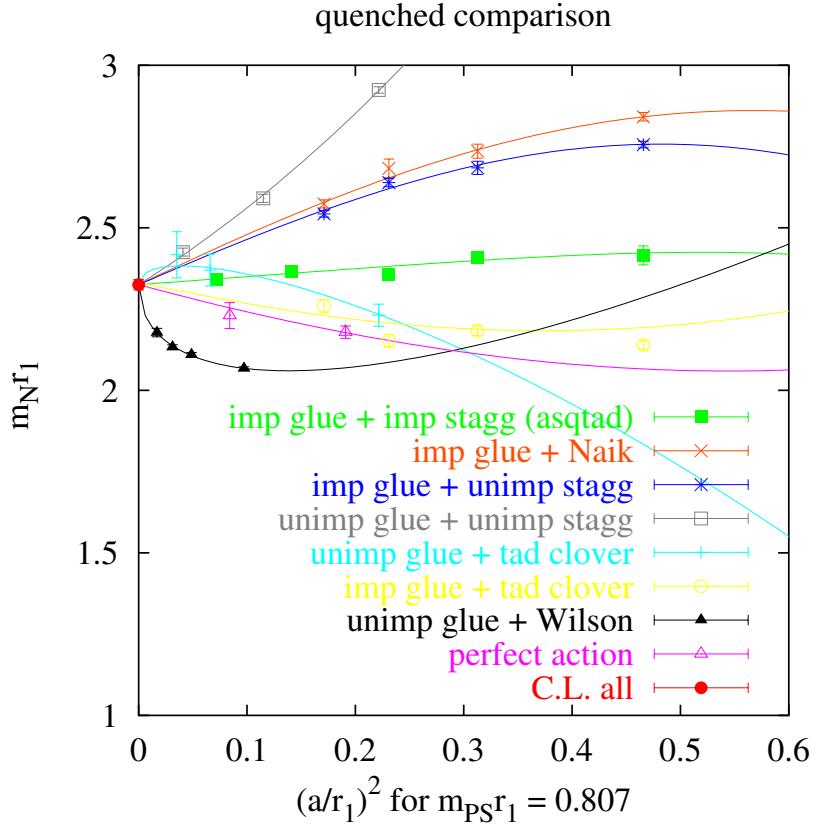


Fig. 15. The plot shows the extrapolation to the continuum limit of the neutron mass computed in LQCD using various types of fermions.<sup>63</sup> Improved glue refers to an  $O(a^2)$  improved gauge action. Different formulations agree with each other within the error but clover and staggered converge faster than Wilson as expected.  $m_{PS}$  in the plot refers to the pion mass which is used to set the unit scale  $r_1$ .

- Quenching. This error is the most difficult to quantify and it is now being eliminated thanks to new algorithms and cheaper computing power. A great deal of effort has been put in this direction in the recent years. Recent computations with dynamical quarks have been a success but the dynamical quark masses are still larger than the physical  $u$  and  $d$  quark mass.
- Chirality. Both Wilson and staggered fermions break chiral symmetry and do not allow computations at zero quark mass. Moreover, the more the chiral limit is approached, the more expensive it is to invert the fermionic matrix  $Q$ . Alternative fermionic discretizations such as domain wall fermions and overlap fermions promise restoration of the chiral symmetry in the continuum limit and provide a better approximation of continuum physics.
- Matching. Most experimental quantities are usually expressed in the  $\overline{MS}$  scheme, while the lattice computations are performed in a different regularization/renormalization scheme. The procedure to relate one to the other is called matching and it mainly perturbative in nature. When lattice results are converted

in the  $\overline{MS}$  scheme using one-loop matching they become affected by matching errors as big as 20%. The computation of matching beyond one-loop is a very challenging task. Some attempts to automate this process via a numerical approach to lattice perturbation theory can be found in Refs. 67 and 68. It is important to stress that the use of the  $\overline{MS}$  scheme is a convention and this matching would not be necessary if the lattice regularization were used everywhere.

In LQCD, errors due to continuum extrapolation, extrapolation to physical masses (both for heavy quarks and light quarks) and finite lattice size are today very much under control and below 2% for most quantities of phenomenological interest.

Figures 16 and 17 show the effect of quenching on the light quark spectrum and how the latest dynamical LQCD computations<sup>69,70</sup> agree with experiment.

## 6. Conclusion

The lattice formulation provides a way to regularize, define and compute the Path Integral in a Quantum Field Theory. This formulation and the associated numerical techniques have been of crucial importance in deepening our knowledge of quantum field theories in physical regimes where perturbation theory fails, as in the case of QCD at strong coupling. For example, LQCD computations have played and are still playing an important role in understanding the process of quark confinement,<sup>71,72</sup> determining the phase structure of gauge theories,<sup>73</sup> confronting QCD predictions

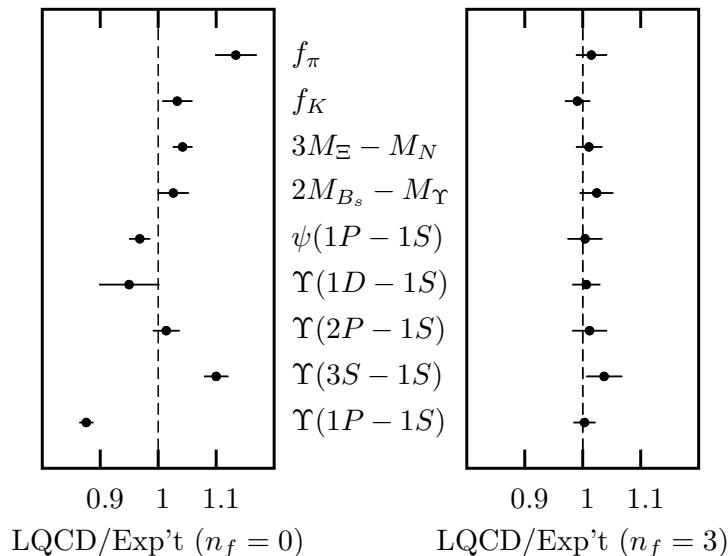


Fig. 16. Recent LQCD results from the MILC collaboration.<sup>68</sup> The two images show ratios of LQCD results over experimental results for various QCD quantities of phenomenological interest without and with dynamical fermions, respectively. Dynamical fermions are implemented using  $O(a^2)$  improved staggered fermions.

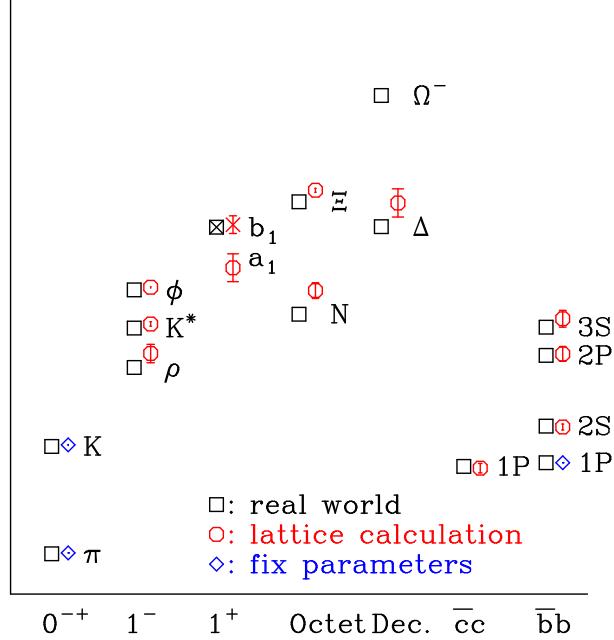


Fig. 17. Unquenched hadron masses and splittings compared with experimental values.<sup>69</sup> The  $\Upsilon$  and  $c\bar{c}$  columns are differences from the ground state masses as computed by HPQCD and Fermilab groups. The  $\pi$ ,  $K$  and  $\Upsilon(1P-1S)$  masses are used to fix the quark masses and the lattice spacing.

with experiments,<sup>69</sup> and extracting fundamental physical parameters such as quark masses<sup>74,75</sup> and CKM matrix elements<sup>76,77</sup> from experimental results.

Three advantages of LQCD over most analytical techniques is that it is easier to understand, the effect of its approximations can be controlled numerically, and the precision of any computation can be reduced arbitrarily just by increasing the dedicated computing time.

Today most LQCD computations are limited by available computing power. In order to overcome this limitation many groups have been working on designing dedicated machines for LQCD such as APEnext,<sup>78</sup> QCDOC<sup>79</sup> and the Earth Simulator.<sup>80</sup> Other groups have focused on the development of software and algorithms optimized for commodity hardware such as PC clusters<sup>81</sup> and building infrastructures for the exchange of field configurations.<sup>82</sup>

Most of the algorithms and the examples discussed here and many more are implemented in a free software library called FermiQCD.<sup>83–85</sup> Despite its name, FermiQCD is not LQCD specific but it is suitable for generic LQFT computations. It has a modular design and an easy to use syntax very similar to the one we have used in this paper (in fact, algorithms such as the MinRes and the BiCGStab map line by line). Moreover, all of the FermiQCD algorithms are parallel and they can run on distributed memory architectures such as PC clusters. FermiQCD has been used for production grade computations at Fermilab and other institutions.

FermiQCD can be downloaded from [www.fermiqcd.net](http://www.fermiqcd.net).

### Acknowledgments

I wish to acknowledge the Fermilab theory group for many years of fruitful collaboration and specifically thank Paul Mackenzie for his comments about this manuscript. I thank Anthony Duncan, Estia Eichten and the MILC collaboration for giving me permission to reproduce some plots from their papers, and David Kaplan for spotting a typo in my first draft. I am also very grateful to K. K. Phua, editor-in-chief of this journal, for inviting me to write this review.

## Appendix A. Useful Formulas in 4D Euclidean Space–Time

### A.1. Wick rotation

The Euclidean action is obtained from the Minkowskian one by performing a Wick rotation. Under this rotation the basic vectors of the theory transform according with the following table (E for Euclidean, M for Minkowski)

E	M	E	M
$x^0$	$ix^0$	$x^i$	$x^i$
$\partial^0$	$-i\partial_0$	$\partial^i$	$\partial_i$
$A^4$	$-iA_0$	$A^i$	$A_i$
$F^0 i$	$-iF_{0i}$	$F^{ij}$	$F_{ij}$
$\gamma^0$	$\gamma^0$	$\gamma^i$	$-i\gamma^i$
$\gamma^5$	$\gamma^5$		

(A.1)

and the integration measure transforms as follows:

$$\exp(-\mathcal{S}_E) = \exp(i\mathcal{S}_M), \quad (\text{A.2})$$

where

$$\mathcal{S}_E = \int d^4x_E \mathcal{L}_E[\dots] = -i \int d^4x_M \mathcal{L}_M[\dots]. \quad (\text{A.3})$$

The choice  $d^4x_E = id^4x_M$  can be made, hence  $\mathcal{L}_E[\dots] = -\mathcal{L}_M[\dots]$

The Euclidean metric tensor is defined as

$$g_E^{\mu\nu} = -\delta^{\mu\nu} = \text{diag}(-1, -1, -1, -1). \quad (\text{A.4})$$

### A.2. Spin matrices

- Dirac matrices (Dirac representation)

$$\gamma^0 = \begin{pmatrix} \mathbf{1} & 0 \\ 0 & -\mathbf{1} \end{pmatrix}, \quad \gamma^i = \begin{pmatrix} 0 & -i\sigma_i \\ i\sigma_i & 0 \end{pmatrix}, \quad \gamma^5 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \quad (\text{A.5})$$

- Dirac matrices (Chiral representation)

$$\gamma^0 = \begin{pmatrix} 0 & \mathbf{1} \\ \mathbf{1} & 0 \end{pmatrix}, \quad \gamma^i = \begin{pmatrix} 0 & -i\sigma_i \\ i\sigma_i & 0 \end{pmatrix}, \quad \gamma^5 = \begin{pmatrix} -\mathbf{1} & 0 \\ 0 & \mathbf{1} \end{pmatrix}. \quad (\text{A.6})$$

All the Euclidean Dirac matrices are Hermitian. The following relations hold:

$$g^{\mu\nu} = \frac{1}{2}\{\gamma^\mu, \gamma^\nu\} = \delta^{\mu\nu}, \quad (\text{A.7})$$

$$\sigma^{\mu\nu} = \frac{i}{2}[\gamma^\mu, \gamma^\nu], \quad (\text{A.8})$$

$$\gamma^5 = \gamma^0\gamma^1\gamma^2\gamma^3 \quad (\text{A.9})$$

and all the  $\sigma^{\mu\nu}$  are Hermitian.

- Projectors

$$L = \frac{1 - \gamma^5}{2}, \quad R = \frac{1 + \gamma^5}{2}. \quad (\text{A.10})$$

- Traces

$$\text{tr}(\gamma^\mu\gamma^\nu) = 4\delta^{\mu\nu}, \quad (\text{A.11})$$

$$\text{tr}(\gamma^\mu\gamma^\nu\gamma^\rho) = 0, \quad (\text{A.12})$$

$$\text{tr}(\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma) = 4(\delta^{\mu\nu}\delta^{\rho\sigma} - \delta^{\mu\rho}\delta^{\nu\sigma} + \delta^{\mu\sigma}\delta^{\rho\nu}), \quad (\text{A.13})$$

$$\text{tr}(\gamma^5\gamma^\mu\gamma^\nu\gamma^\rho\gamma^\sigma) = 4\epsilon_E^{\mu\nu\rho\sigma}, \quad (\text{A.14})$$

where  $\epsilon_E^{0123} = -1$ .

## References

1. M. Peskin and D. V. Shroeder, *Quantum Field Theory* (Addison Wesley, 1994).
2. C. Itzykson and J.-Z. Zuber, *Quantum Field Theory* (McGraw-Hill, 1985).
3. M. Creutz, *Quarks, Gluons and Lattices* (Cambridge University Press, 1983).
4. H. J. Rothe, *Lattice Gauge Theories*, 3rd edn. (World Scientific, Singapore, 2005).
5. I. Montvay and G. Münster, *Quantum Fields on a Lattice* (Cambridge University Press, 1994).
6. R. Gupta, hep-lat/9807028.
7. T. Muta, *Foundations of Quantum Chromodynamics* (World Scientific, Singapore, 1987).
8. W. Marciano and H. Pagels, *Phys. Rep.* **36**, 135 (1978).
9. L. Maiani and M. Testa, *Phys. Lett. B* **245**, 585 (1990).
10. S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability* (Springer-Verlag, London, 1993).
11. G. Banhot, *Rep. Prog. Phys.* **51**, 429 (1988).
12. J. Shao and D. Tu, *The Jackknife and Bootstrap* (Springer-Verlag, 1995).
13. K. Symanzik, *Nucl. Phys. B* **226**, 187, 205 (1983).
14. D. J. Gross and F. Wilczek, *Phys. Rev. Lett.* **30**, 1343 (1973); D. J. Gross and F. Wilczek, *Phys. Rev. D* **8**, 3633 (1973); H. Politzer, *Phys. Rev. Lett.* **30**, 1346 (1973); H. Politzer, *Phys. Rep. C* **14**, 129 (1974).
15. J. M. Rabin, Introduction to quantum field theory for mathematicians, in *Geometry and Quantum Field Theory*, IAS/Park City Mathematics Series, Vol. 1, eds. D. S. Freed and K. K. Uhlenbeck (American Mathematical Society, Providence, RI, 1995).
16. J. Glimm and R. Jaffe, *Quantum Physics*, 2nd edn. (Springer-Verlag, 1987).

17. M. Gockeler, R. Horsley, A. C. Irving, D. Pleiter, P. E. L. Rakow, G. Schierholz and H. Stuben, hep-ph/0502212.
18. Y. Aharonov and D. Bohm, *Phys. Rev. (Ser. 2)* **115**, 485 (1959).
19. K. G. Wilson, *Phys. Rev. D* **10**, 2445 (1974).
20. M. Creutz, *Phys. Rev. D* **21**, 2308 (1980).
21. N. Cabibbo and E. Marinari, *Phys. Lett. B* **119**, 387 (1982).
22. A. Duncan, E. Eichten and H. Thacker, *Phys. Rev. D* **59**, 014505 (1999).
23. G. 't Hooft, *Nucl. Phys. B* **138**, 1 (1978).
24. B. Sheikholeslami and R. Wolhert, *Nucl. Phys. B* **259**, 572 (1985).
25. M. Lüscher, Advanced Lattice QCD, Talk given at Les Houches Summer School in Theoretical Physics, Session 68 (1998), hep-lat/9802029.
26. M. Lüscher, S. Sint, R. Sommer and H. Wittig, *Nucl. Phys. B* **491**, 344 (1997).
27. G. P. Lepage and P. B. Mackenzie, *Phys. Rev. D* **48**, 225 (1993).
28. A. X. El-Khadra, A. S. Kronfeld, P. B. Mackenzie, S. M. Ryan and J. N. Simone, *Phys. Rev. D* **58**, 014506 (1998).
29. M. B. Oktay, A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, *Nucl. Phys. Proc. Suppl.* **129**, 349 (2004).
30. M. B. Wise, hep-ph/9311212.
31. A. Manohar, *Phys. Rev. D* **56**, 230 (1997).
32. J. Kogut and L. Susskind, *Phys. Rev. D* **11**, 395 (1975).
33. T. Banks, J. Kogut and L. Susskind, *Phys. Rev. D* **13**, 1043 (1996).
34. L. Susskind, *Phys. Rev. D* **16**, 3031 (1977).
35. G. W. Kilcup and S. R. Sharpe, *Nucl. Phys. B* **283**, 493 (1987).
36. M. F. Golterman, *Nucl. Phys. B* **273**, 663 (1986).
37. G. P. Lepage, *Phys. Rev. D* **59**, 074502 (1999).
38. W. Lee and S. R. Sharpe, *Phys. Rev. D* **60**, 114503 (1999).
39. MILC Collab. (C. Bernard *et al.*), *Phys. Rev. D* **58**, 014503 (1998).
40. H. B. Nielsen and M. Ninomiya, *Nucl. Phys. B* **185**, 20 (1981), [Erratum, *ibid.* **195**, 541 (1982)].
41. H. Neuberger, *Phys. Rev. Lett.* **81**, 4060 (1998).
42. R. Narayanan and H. Neuberger, *Phys. Rev. Lett.* **71**, 3251 (1993); *Nucl. Phys. B* **412**, 574 (1994); *ibid.* **443**, 305 (1995).
43. D. B. Kaplan, *Phys. Lett. B* **288**, 342 (1992).
44. Y. Shamir, *Nucl. Phys. B* **406**, 90 (1993).
45. V. Furman and Y. Shamir, *Nucl. Phys. B* **439**, 54 (1995).
46. V. Furman and Y. Shamir, *Nucl. Phys. B* **439**, 54 (1995).
47. T. Blum *et al.*, *Phys. Rev. D* **69**, 074502 (2004).
48. A. Borici, *Nucl. Phys. Proc. Suppl.* **83**, 771 (2000).
49. A. D. Kennedy, *Nucl. Phys. Proc. Suppl.* **140**, 190 (2005).
50. S. Durr and C. Hoelbling, *Phys. Rev. D* **71**, 054501 (2005).
51. S. R. Sharpe and Y. Zhang, *Phys. Rev. D* **53**, 5125 (1996).
52. UKQCD Collab. (C. Michael and J. Peisa), *Phys. Rev. D* **58**, 034506 (1998).
53. TrinLat Collab. (A. O'Cais, K. J. Juge, M. J. Peardon, S. M. Ryan and J. I. Skullerud), hep-lat/0409069.
54. R. T. Scalettar, D. J. Scalapino and R. L. Sugar, *Phys. Rev. B* **34**, 7911 (1986).
55. S. Duane, A. D. Kennedy, B. J. Pendleton and D. Roweth, *Phys. Lett. B* **195**, 216 (1987).
56. M. Lüscher, *Comp. Phys. Commun.* **156**, 209 (2004).
57. M. Lüscher, *J. High Energy Phys.* **05**, 052 (2003).
58. H. Neuberger, *Phys. Rev. D* **60**, 065006 (1999).

59. T. DeGrand and S. Schaefer, *Phys. Rev. D* **71**, 034507 (2005).
60. Z. Fodor, S. D. Katz and K. K. Szabo, hep-lat/0409070.
61. P. Chen *et al.*, hep-lat/9812011.
62. Y. Aoki *et al.*, hep-lat/0411006.
63. C. T. Sachrajda, Lattice simulations and effective theories, Lectures presented at the Advanced School on Effective Theories, Almunecar (Spain) June 1995, hep-lat/960527.
64. MILC Collab. (C. Aubin *et al.*), *Phys. Rev. D* **70**, 114501 (2004).
65. S. Basak *et al.*, hep-lat/0508018.
66. C. T. H. Davies, G. P. Lepage, F. Niedermayer and D. Toussaint, *Nucl. Phys. Proc. Suppl.* **140**, 261 (2005).
67. F. Di Renzo and L. Scorzato, *J. High Energy Phys.* **0410**, 073 (2004).
68. H. D. Trottier, *Nucl. Phys. Proc. Suppl.* **129**, 142 (2004).
69. HPQCD Collab. (C. T. H. Davies *et al.*), *Phys. Rev. Lett.* **92**, 022001 (2004).
70. C. Aubin *et al.*, *Phys. Rev. D* **70**, 094505 (2004).
71. C. W. Bernard *et al.*, *Phys. Rev. D* **62**, 034503 (2000).
72. M. Engelhardt, *Nucl. Phys. B Proc. Suppl.* **140**, 92 (2005).
73. J. B. Kogut and M. A. Stephanov, *Camb. Monogr. Part. Phys. Nucl. Phys. Cosmol.* **21**, 1 (2004).
74. QCDSF Collab. (M. Gockeler, R. Horsley, A. C. Irving, D. Pleiter, P. E. L. Rakow, G. Schierholz and H. Stüber), hep-ph/0409312.
75. UKQCD Collab. (C. McNeile, C. Michael and G. Thompson), *Phys. Lett. B* **600**, 77 (2004).
76. D. Becirevic, hep-ph/0211340.
77. M. Okamoto, hep-ph/0505190.
78. ApeNEXT Collab. (F. Bodin *et al.*), *Nucl. Phys. Proc. Suppl.* **140**, 176 (2005).
79. P. Boyle *et al.*, *Nucl. Phys. Proc. Suppl.* **140**, 169 (2005).
80. S. Aoki *et al.*, *Nucl. Phys. Proc. Suppl.* **129**, 859 (2004).
81. D. Holmgren, P. B. Mackenzie, D. Petratwick, R. Rechenmacher and J. Simone, Prepared for CHEP'01: Computing in High-Energy Physics and Nuclear, Beijing, China, 3–7 Sep. 2001.
82. B. Joo and W. Watson, *Nucl. Phys. Proc. Suppl.* **140**, 209 (2005).
83. M. Di Pierro, hep-lat/0011083.
84. M. Di Pierro, *Nucl. Phys. Proc. Suppl.* **106**, 1034 (2002).
85. FermiQCD Colab. (M. Di Pierro *et al.*), *Nucl. Phys. Proc. Suppl.* **129**, 832 (2004).

**Copyright of International Journal of Modern Physics A: Particles & Fields; Gravitation; Cosmology; Nuclear Physics is the property of World Scientific Publishing Company and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.**

# B and D meson semileptonic decays in three-flavor lattice QCD

P.B. Mackenzie<sup>\*a</sup>, C. Aubin<sup>c,d</sup>, C. Bernard<sup>d</sup>, C. DeTar<sup>e</sup>, M. Di Pierro<sup>f</sup>,  
Steven Gottlieb<sup>h</sup>, E. Gregory<sup>j</sup>, U.M. Heller<sup>k</sup>, J.E. Hetrick<sup>l</sup>, A.X. El-Khadra<sup>m</sup>,  
A.S. Kronfeld<sup>a</sup>, L. Levkova<sup>h</sup>, F. Maresca<sup>e</sup>, D. Menscher<sup>m</sup>, M. Nobes<sup>n,o</sup>, M. Okamoto<sup>a</sup>,  
M.B. Oktay<sup>q</sup>, J. Osborn<sup>e</sup>, D. Renner<sup>j</sup>, J.N. Simone<sup>a</sup>, R. Sugar<sup>p</sup>, D. Toussaint<sup>j</sup>,  
H.D. Trottier<sup>o</sup>

E-mail: [mackenzie@fnal.gov](mailto:mackenzie@fnal.gov)

<sup>a</sup>Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

<sup>c</sup>Physics Department, Columbia University, New York, New York, USA

<sup>d</sup>Department of Physics, Washington University, St. Louis, Missouri 63130, USA

<sup>e</sup>Physics Department, University of Utah, Salt Lake City, Utah 84112, USA

<sup>f</sup>School of Computer Science, Telecommunications and Information Systems, DePaul University,  
Chicago, Illinois 60604, USA

<sup>h</sup>Department of Physics, Indiana University, Bloomington, Indiana 47405, USA

<sup>j</sup>Department of Physics, University of Arizona, Tucson, Arizona 85721, USA

<sup>k</sup>American Physical Society, Ridge, New York 11961, USA

<sup>l</sup>Physics Department, University of the Pacific, Stockton, California 95211, USA

<sup>m</sup>Physics Department, University of Illinois, Urbana, Illinois 61801, USA

<sup>n</sup>Laboratory of Elementary-Particle Physics, Cornell University, Ithaca, New York 14853, USA

<sup>o</sup>Physics Department, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada

<sup>p</sup>Department of Physics, University of California, Santa Barbara, California 93106, USA

<sup>q</sup>School of Mathematics, Trinity College, Dublin, Ireland

## Fermilab Lattice, MILC, and HPQCD Collaborations

We have calculated the semileptonic form factors of  $B$  and  $D$  mesons using unquenched, improved staggered light quarks, and improved clover heavy quarks. We discuss the use of unitarity constraints to bound the form factors in the high recoil momentum region.

XXIIIrd International Symposium on Lattice Field Theory  
25-30 July 2005  
Trinity College, Dublin, Ireland

---

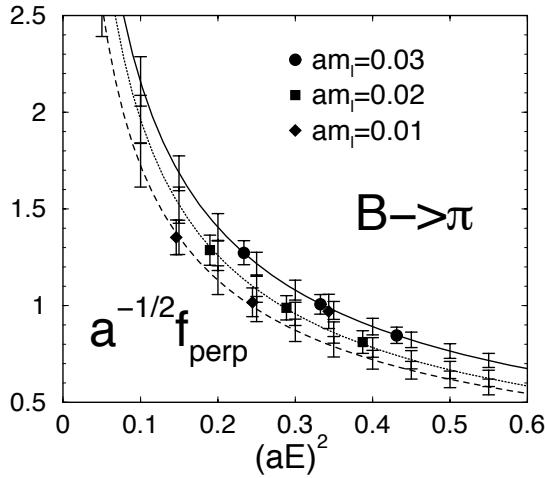
\*Speaker.

## 1. Introduction

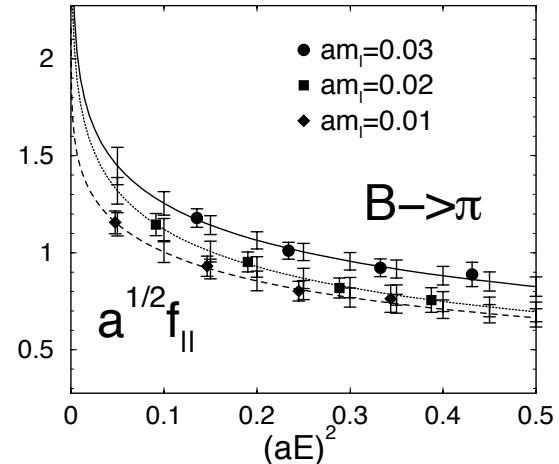
Semileptonic decays of  $B$  and  $D$  mesons have a central place in lattice phenomenology of the Standard Model. They are processes with single, stable mesons, and so are among the most tractable of current lattice calculations. They are of crucial importance in determining Standard Model parameters: they can pin down four elements of the CKM matrix.  $D$  semileptonic decays are of particular interest now because CLEO is measuring them more accurately than ever before (a few per cent).  $B$  semileptonic decays are of interest because they feed directly into the analysis of the unitarity triangle. We are making a study of these decays with unquenched improved staggered fermions, following the general approach begun in Ref. [1]. Results for  $D$  mesons were reported in Ref. [2]. Earlier results were reported in Ref. [3]. Note also this year's plenary review at Lattice 2005 by Okamoto.

## 2. Methods

We use  $\mathcal{O}(a)$  improved clover heavy quarks with the Fermilab interpretation [4]. We use  $\mathcal{O}(a^2)$  improved, “asqtad” light quarks [5] and  $\mathcal{O}(a^2)$  improved glue. Our main results employ the asqtad “coarse” data set, with lattice spacing  $a = 0.121$  fm., volume= $20^3 \times 64$ , and light quark masses  $m_l = 0.12\text{--}0.3 m_s$ . The data sets for each light quark mass contained around 500 configurations each. We used four separate time sources on each gauge configuration. No correlations were visible between the time sources. Staggered chiral perturbation theory [6] was used in the chiral extrapolations. One-loop lattice perturbation theory [7] was used, with a nonperturbative estimate of the current renormalization following Ref. [8].

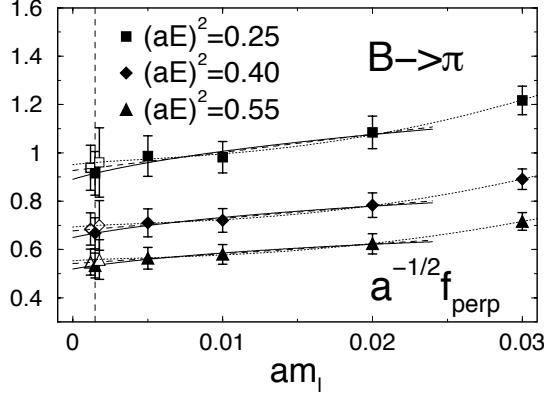


**Figure 1:** The form factor  $f_{\perp}$  for  $B$  semileptonic decay. The Bećerivić-Kaidalov function was used to interpolate to fiducial values of the energy. The analogous graphs for  $D$  decay are given in Refs. [2] and [3].

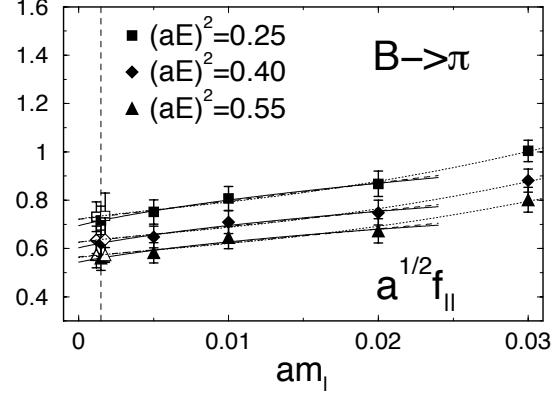


**Figure 2:** The same as Fig. 1, but for  $f_{\parallel}$ .

In the main analysis, for a given light quark mass, the Bećirević-Kaidalov parameterization of the form factors [9] was used to interpolate the form factors to fiducial values of the energy, as illustrated in Figs. 1 and 2. These interpolated form factors were then extrapolated to the physical quark mass, as shown in Figs. 3 and 4.

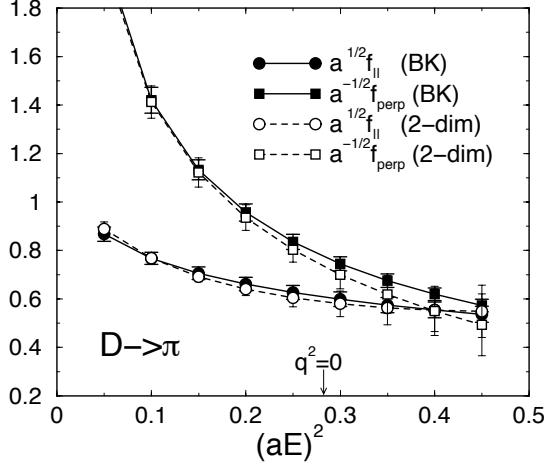


**Figure 3:** Chiral extrapolation of  $f_{\text{perp}}$ .

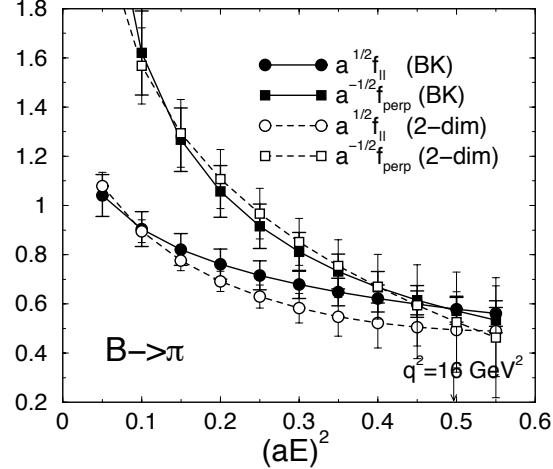


**Figure 4:** Chiral extrapolation of  $f_{||}$ .

To check whether the use of the BK parameterization introduced any model-dependence into the results, we redid some fits, replacing the two-stage fit (energy interpolation followed by chiral extrapolation) with a two-dimensional fit in energy and quark mass simultaneously. Results were consistent, but with larger statistical errors in the two-dimensional fit, especially in the high pion momentum region, as shown in Figs. 5 and 6. The difference is used as an estimate of uncertainty



**Figure 5:** The form factors for  $D$  semileptonic decay obtained with a two-dimensional chiral and  $E^2$  fit compared with those obtained with Bećirević-Kaidalov based energy interpolation, followed by chiral extrapolation.



**Figure 6:** The same as Fig. 5, but for  $B$  decay.

due to the BK parameterization. The BK-based fitting produces reduced statistical errors in the high-momentum region, at the cost of introducing possible model dependence.

As a preliminary check for discretization errors, the form factors for decay into relatively heavy  $u$  and  $d$  quarks have been compared on the asqtad coarse and fine ( $a = 0.086$  fm) lattices. Results agreed within statistical errors. Discretization uncertainties were estimated with HQET power counting, and are our largest current uncertainty. Analyzing form factor data on a larger range of lattice spacings is likely to give the greatest future improvement in our uncertainties.

### 3. Preliminary results

Preliminary estimates for the uncertainties in CKM matrix element determinations are shown in Table 1. Comparing predictions with experiment, we obtain for the CKM matrix elements:

$$|V_{ub}| \times 10^3 = 3.48(29)(38)(47), \quad (3.1)$$

$$|V_{cd}| = 0.239(10)(24)(20), \quad (3.2)$$

$$|V_{cs}| = 0.969(39)(94)(24), \quad (3.3)$$

where the errors are statistical, systematic, and experimental.

If we instead use the accepted values of the CKM matrix elements as inputs, we obtain for the  $D$  meson decay rates

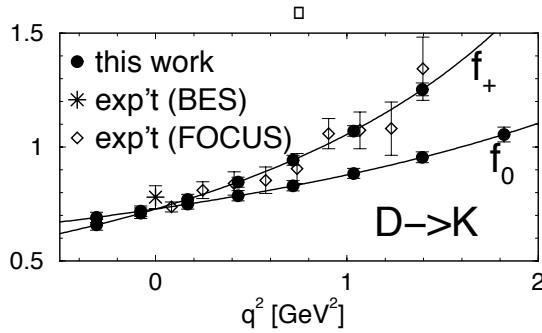
$$\begin{aligned} \Gamma(D^0 \rightarrow \pi^- l^+ \nu) &= (7.7 \pm 0.6 \pm 1.5 \pm 0.8) \times 10^{-3} \text{ps}^{-1}, \\ \Gamma(D^0 \rightarrow K^- l^+ \nu) &= (9.2 \pm 0.7 \pm 1.8 \pm 0.2) \times 10^{-2} \text{ps}^{-1}, \\ \frac{\Gamma(D^0 \rightarrow \pi^- l^+ \nu)}{\Gamma(D^0 \rightarrow K^- l^+ \nu)} &= 0.084 \pm 0.007 \pm 0.017 \pm 0.009, \end{aligned} \quad (3.4)$$

where the errors are statistical, systematic, and from CKM matrix element.

After the publication of our results for the  $q^2$  dependence of the  $D \rightarrow Kl\nu$  form factor, a high-statistics measurement of the shape appeared from the FOCUS Collaboration [10]. As can be seen in Fig. 7, the results agree well.

**Table 1:** Systematic errors for CKM matrix elements from the semileptonic decays. Errors for  $V_{ub}$  decay are obtained from the integration with  $q_{\min}^2 = 16$  GeV $^2$ .

decay	$D \rightarrow \pi(K)l\nu$	$B \rightarrow \pi l\nu$
CKM matrix element	$ V_{cd(s)} $	$ V_{ub} $
discretization effect	9%	9%
fitting 3- and 2-point functions	3%	3%
chiral extrapolation	3%(2%)	4%
$q^2$ dependence (BK parameterization)	2%	4%
current renormalization	0%	1%
$a$ uncertainty	1%	1%
total systematic	10%	11%



**Figure 7:** The form factors in  $D \rightarrow Kl\nu$  decay, compared with experimental data for  $f_+$ .

#### 4. Unitarity constraints

Unitarity constraints can be used without introducing model dependence to constrain the shape of form factors in the high recoil momentum region (where our statistical uncertainties range from poor to infinite). Arnesen, Grinstein, Rothstein, and Stewart [11] have recently given a lucid explanation. The function

$$z(t, t_0) = \frac{\sqrt{t_+ - t} - \sqrt{t_+ - t_0}}{\sqrt{t_+ - t} + \sqrt{t_+ - t_0}} \quad (4.1)$$

maps  $q^2 = t > t_+$  onto  $|z| = 1$ , and  $t < t_+$  onto  $[-1, 1]$  in the complex plane. Here,  $t = (p_H - p_L)^2$ ,  $t_+ = (m_H + m_L)^2$ , and  $t_- = (m_H - m_L)^2$ .  $t_0$ , taken as  $0.65 t_-$  here, is a free parameter adjusted to center the physical region on  $z \sim 0$ .

For the various semileptonic decays, the physical region,  $0 < t < t_-$ , is mapped into

$B \rightarrow \pi l \nu$ :  $-0.34 < z < 0.22$ ,

$D \rightarrow \pi l \nu$ :  $-0.17 < z < 0.16$ ,

$D \rightarrow Kl\nu$ :  $-0.04 < z < 0.06$ ,

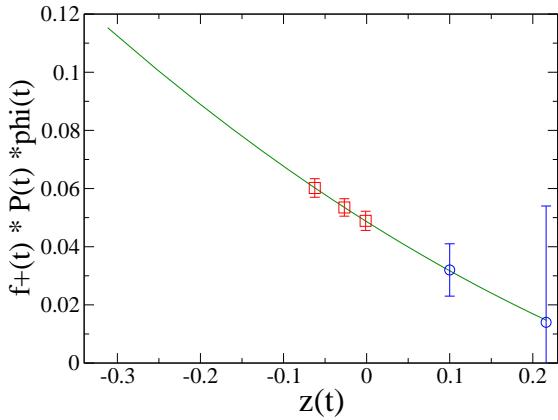
$B \rightarrow D l \nu$ :  $-0.02 < z < 0.04$ .

When unitarity is taken into account, the smallness of the physical range of  $z$  will constitute a small parameter that limits the number of parameters required to describe the form factors to a given accuracy.

In this parameterization, the form factors have the form:

$$f(t) = \frac{1}{P(t)\phi(t, t_0)} \sum_{k=0}^{\infty} a_k(t_0) z(t, t_0)^k. \quad (4.2)$$

$P(t)$  and  $\phi(t, t_0)$  contain most of the complexity of the form factors. Unitarity requires simply that  $\sum a_k^2 < 1$ . In  $B$  semileptonic decay, just five terms in the series are necessary for 1% accuracy. Fitting three points of our raw data for  $f_+$  at  $m_l = 0.02$  (red squares in Fig. 8), we obtain from the fit (shown without error bars as a green line) good constraints on the form factor beyond the region where we have data. (Two sample points are shown with error bars as blue circles. The leftmost blue point corresponds to a recoil momentum of around 1.7 GeV.) (Our data end at recoil momenta of around 1 GeV.) We are currently examining a simultaneous fit of  $f_+$  and  $f_0$ , and the use of unitarity-based interpolation parameters to supplement or replace the BK-based fitting.



**Figure 8:** A fit (green line) of our data (red squares) for  $f_+$  normalized by  $P(t)$  and  $\phi(t, t_0)$ , as a function of  $z(t, t_0)$ . Unitarity-based extrapolation of our data into the high recoil-momentum region (blue circles) gives useful constraints beyond the range of the lattice data.

## References

- [1] C. T. H. Davies *et al.* [HPQCD, MILC, and Fermilab Lattice Collaborations], Phys. Rev. Lett. **92**, 022001 (2004) [arXiv:hep-lat/0304004].
- [2] C. Aubin *et al.* [HPQCD, MILC, and Fermilab Lattice Collaborations], Phys. Rev. Lett. **94**, 011601 (2005) [arXiv:hep-ph/0408306].
- [3] M. Okamoto *et al.*, Nucl. Phys. Proc. Suppl. **140**, 461 (2005) [arXiv:hep-lat/0409116]; M. Okamoto *et al.*, Nucl. Phys. Proc. Suppl. **129**, 334 (2004) [arXiv:hep-lat/0309107].
- [4] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997) [arXiv:hep-lat/9604004].
- [5] T. Blum *et al.*, Phys. Rev. D **55**, 1133 (1997); K. Orginos and D. Toussaint, *ibid.* **59**, 014501 (1999); J. Lagae and D. Sinclair, *ibid.* **59**, 014511 (1999); G. P. Lepage, *ibid.* **59**, 074502 (1999); K. Orginos, D. Toussaint and R. L. Sugar, *ibid.* **60**, 054503 (1999); C. Bernard *et al.*, *ibid.* **61**, 111502 (2000).
- [6] C. Aubin and C. Bernard, arXiv:hep-lat/0409027, and work in progress.
- [7] M. Nobes *et al.*, work in progress.
- [8] A. X. El-Khadra *et al.* Phys. Rev. D **64**, 014502 (2001).
- [9] D. Bećirević and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000) [arXiv:hep-ph/9904490].
- [10] J. M. Link *et al.* [FOCUS Collaboration], Phys. Lett. B **607**, 233 (2005) [arXiv:hep-ex/0410037].
- [11] M. C. Arnesen, B. Grinstein, I. Z. Rothstein and I. W. Stewart, Phys. Rev. Lett. **95**, 071802 (2005) [arXiv:hep-ph/0504209].

# Charmed-Meson Decay Constants in Three-Flavor Lattice QCD

C. Aubin,<sup>1</sup> C. Bernard,<sup>2</sup> C. DeTar,<sup>3</sup> M. Di Pierro,<sup>4</sup> E. D. Freeland,<sup>5</sup> Steven Gottlieb,<sup>6</sup> U. M. Heller,<sup>7</sup> J. E. Hetrick,<sup>8</sup> A. X. El-Khadra,<sup>9</sup> A. S. Kronfeld,<sup>10</sup> L. Levkova,<sup>6</sup> P. B. Mackenzie,<sup>10</sup> D. Menscher,<sup>9</sup> F. Maresca,<sup>3</sup> M. Nobes,<sup>11</sup> M. Okamoto,<sup>10</sup> D. Renner,<sup>12</sup> J. Simone,<sup>10</sup> R. Sugar,<sup>13</sup> D. Toussaint,<sup>12</sup> and H. D. Trottier<sup>14</sup>

(Fermilab Lattice, MILC, and HPQCD Collaborations)

<sup>1</sup>*Physics Department, Columbia University, New York, New York 10027, USA*

<sup>2</sup>*Department of Physics, Washington University, St. Louis, Missouri 63130, USA*

<sup>3</sup>*Physics Department, University of Utah, Salt Lake City, Utah 84112, USA*

<sup>4</sup>*School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois 60604, USA*

<sup>5</sup>*Liberal Arts Department, The School of the Art Institute of Chicago, Chicago, Illinois 60603, USA*

<sup>6</sup>*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

<sup>7</sup>*American Physical Society, Ridge, New York 11961, USA*

<sup>8</sup>*Physics Department, University of the Pacific, Stockton, California 95211, USA*

<sup>9</sup>*Physics Department, University of Illinois, Urbana, Illinois 61801, USA*

<sup>10</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

<sup>11</sup>*Laboratory of Elementary-Particle Physics, Cornell University, Ithaca, New York 14853, USA*

<sup>12</sup>*Department of Physics, University of Arizona, Tucson, Arizona 85721, USA*

<sup>13</sup>*Department of Physics, University of California, Santa Barbara, California 93106, USA*

<sup>14</sup>*Physics Department, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6*

(Received 28 June 2005; published 14 September 2005)

We present the first lattice QCD calculation with realistic sea quark content of the  $D^+$ -meson decay constant  $f_{D^+}$ . We use the MILC Collaboration's publicly available ensembles of lattice gauge fields, which have a quark sea with two flavors (up and down) much lighter than a third (strange). We obtain  $f_{D^+} = 201 \pm 3 \pm 17$  MeV, where the errors are statistical and a combination of systematic errors. We also obtain  $f_{D_s} = 249 \pm 3 \pm 16$  MeV for the  $D_s$  meson.

DOI: 10.1103/PhysRevLett.95.122002

PACS numbers: 12.38.Gc, 13.20.Fc

Flavor physics currently plays a central role in elementary particle physics [1]. To aid the experimental search for physics beyond the standard model, several hadronic matrix elements must be calculated nonperturbatively from quantum chromodynamics (QCD). One of the most important of these is the decay constant of the  $B$  meson  $f_B$  [2]. Any framework for calculating  $f_B$  should, therefore, be subjected to stringent tests, and such a test is a key aim of this Letter.

The most promising method for these nonperturbative calculations is numerical lattice QCD. For many years the results suffered from an unrealistic treatment of the effects of sea quarks. In the last few years, however, this obstacle seems to have been removed: with three flavors of sea quarks lattice QCD now agrees with experiment for a wide variety of hadronic quantities [3]. This validation of lattice QCD has been realized, so far, only for so-called “gold-plated” quantities: masses and matrix elements of the simplest hadronic states. Note, however, that many of the hadronic matrix elements relevant to flavor physics are in this class, including  $f_B$ .

The challenges in computing  $f_B$  are essentially the same for the  $D^+$ -meson decay constant  $f_{D^+}$ . Experiments have observed the leptonic decay  $D^+ \rightarrow l^+ \nu_l$ , but not  $B^+ \rightarrow$

$l^+ \nu_l$ . One can, thus, determine  $|V_{cd}|f_{D^+}$ , where  $V_{cd}$  is an element of the Cabibbo-Kobayashi-Maskawa (CKM) matrix. Taking  $|V_{cd}|$  from elsewhere, one gets  $f_{D^+}$ . In 2004 the CLEO-c Collaboration measured  $f_{D^+}$  with a 20% error [4], and a more precise measurement is expected soon.

This Letter reports the first lattice-QCD calculation of  $f_{D^+}$  with three flavors of sea quarks [5]. We find

$$f_{D^+} = 201 \pm 3 \pm 6 \pm 9 \pm 13 \text{ MeV}, \quad (1)$$

where the uncertainties are statistical, and a sequence of systematic effects, discussed below. We also obtain the decay constant of the  $D_s$  meson,

$$f_{D_s} = 249 \pm 3 \pm 7 \pm 11 \pm 10 \text{ MeV}. \quad (2)$$

The second result is more precise than a recent lattice-QCD calculation with the same sea quark content but nonrelativistic heavy quarks, which found  $f_{D_s} = 290 \pm 20 \pm 41$  MeV [6]. These results are more reliable than older calculations [7] because we now incorporate (three) sea quarks and, for  $f_{D^+}$ , also because the light valence quark masses are smaller than before.

These results test the methods of Ref. [3] because they are predictions. The input parameters have been fixed previously [3,8–11], and, once comparably precise experi-

mental measurements become available, one can see how Eqs. (1) and (2) fare. Indeed, this work is part of a program to calculate matrix elements for leptonic and semileptonic decays [10,12,13], neutral-meson mixing, and quarkonium [11,14]. So far, these lattice-QCD calculations agree with experiment for the normalization of  $D$ -meson semileptonic form factors [12,15,16]. They also have predicted correctly the form-factor shape [12,17], as well as the mass of the  $B_c$  meson [14,18].

In this set of calculations we use ensembles of unquenched lattice gauge fields generated by the MILC Collaboration [9,19], with lattice spacing  $a = 0.175$ ,  $0.121$ , and  $0.086$  fm. The key feature of these ensembles is that they incorporate three flavors of sea quarks, one whose mass is close to that of the strange quark and two with a common mass taken as light as possible.

For the sea quark and light valence quark we use the “Asqtad” staggered-fermion action [20]. Several different quark masses appear in this calculation; for convenience, they are defined in Table I. At  $a = 0.175$ ,  $0.121$ , and  $0.086$  fm there are, respectively, 4, 5, and 2 ensembles with various sea quark masses ( $m_l, m_h$ ) [9,19]. The larger simulation mass  $m_h$  is close to the physical strange quark mass  $m_s$ . The light pair’s mass  $m_l$  is not as small as those of the up and down quarks in nature, but the range  $0.1m_s \leq m_l \leq 0.8m_s$  suffices to control the extrapolation in quark mass with chiral perturbation theory ( $\chi$ PT). For carrying out the chiral extrapolation, it is useful to allow the valence mass  $m_q$  to vary separately from the sea mass [21]. At  $a = 0.175$ ,  $0.121$ , and  $0.086$  fm we have, respectively, 6, 12, and 8 or 5 values of the valence mass, in the range  $0.1m_s \leq m_q \leq m_s$ .

A drawback of staggered fermions is that they come in four species, called tastes. The steps taken to eliminate three extra tastes per flavor are not (yet) proven, although there are several signs that they are valid. Calculations of  $f_{D^+}$  and  $f_{D_s}$  are sensitive to these steps: if Eqs. (1) and (2) agree with precise measurements, it should be more plausible that the techniques used to reduce four tastes to one are correct.

For the charmed quark we use the Fermilab action for heavy quarks [22]. Discretization effects are entangled with the heavy-quark expansion, so we use heavy-quark

effective theory (HQET) as a theory of cutoff effects [23]. This provides good control, as discussed in Ref. [24], and the framework has been tested with the (successful) prediction of the  $B_c$  meson mass [14]. Nevertheless, heavy-quark discretization effects are the largest source of systematic error in  $f_{D_s}$ , and the second-largest in  $f_{D^+}$ .

The decay constant  $f_{D_q}$ , for a  $D_q$  meson with light valence quark  $q$  and momentum  $p_\mu$ , is defined by [25]

$$\langle 0 | A_\mu | D_q \rangle = i f_{D_q} p_\mu, \quad (3)$$

where  $A_\mu = \bar{q} \gamma_\mu \gamma_5 c$  is an electroweak axial vector current. The combination  $\phi_q = f_{D_q} \sqrt{m_{D_q}}$  emerges directly from the lattice Monte Carlo calculations. As usual in lattice gauge theory, we compute two-point correlation functions  $C_2(t) = \langle O_{D_q}^\dagger(t) O_{D_q}(0) \rangle$ ,  $C_A(t) = \langle A_4(t) O_{D_q}(0) \rangle$ , where  $O_{D_q}$  is an operator with the quantum numbers of the charmed pseudoscalar meson, and  $A_4$  is the (lattice) axial vector current. The operators are built from the heavy-quark and staggered-quark fields as in Ref. [26]. We extract the  $D_q$  mass and the amplitudes  $\langle D | O_{D_q} | 0 \rangle$  and  $\langle 0 | A_4 | D \rangle$  from fits to the known  $t$  dependence. Statistical errors are determined with the bootstrap method, which allows us to keep track of correlations.

The lattice axial vector current must be multiplied by a renormalization factor  $Z_{A_4^{cq}}$ . We write [27]  $Z_{A_4^{cq}} = \rho_{A_4^{cq}} (Z_{V_4^{cc}} Z_{V_4^{qq}})^{1/2}$ , because the flavor-conserving renormalization factors  $Z_{V_4^{cc}}$  and  $Z_{V_4^{qq}}$  are easy to compute non-perturbatively. The remaining factor  $\rho_{A_4^{cq}}$  should be close to unity because the radiative corrections mostly cancel [28]. A one-loop calculation gives [29]  $\rho_{A_4^{cq}} = 1.052$ ,  $1.044$ , and  $1.032$  at  $a = 0.175$ ,  $0.121$ , and  $0.086$  fm. We estimate the uncertainty of higher-order corrections to be  $2\alpha_s(\rho_{A_4^{cq}} - 1) \approx 1.3\%$ ;  $\alpha_s$  is the strong coupling.

The heart of our analysis is the chiral extrapolation, from the simulated to the physical quark masses. It is necessary, and nontrivial, because the cloud of “pions” surrounding the simulated  $D_q$  mesons is not the same as for real pions. With staggered quarks the (squared) pseudoscalar meson masses are

$$M_{ab,\xi}^2 = (m_a + m_b)\mu + a^2 \Delta_\xi, \quad (4)$$

where  $m_a$  and  $m_b$  are quark masses,  $\mu$  is a parameter of  $\chi$ PT, and the representation of the meson under the taste symmetry group is labeled by  $\xi = P, A, T, V, I$  [30]. A symmetry as  $m_a, m_b \rightarrow 0$  ensures that  $\Delta_P = 0$ . The “pion” cloud in the simulation includes all these pseudoscalars.

According to next-to-leading order  $\chi$ PT, the decay constant takes the form

$$\phi_q = \Phi [1 + \Delta f_q(m_q, m_l, m_h) + p_q(m_q, m_l, m_h)], \quad (5)$$

where  $\Phi$  is a quark-mass-independent parameter.  $\Delta f_q$

arises from loop processes involving light pseudoscalar mesons, and  $p_q$  is an analytic function. To obtain them, one must take into account the flavor-taste symmetry of the simulation [30] and the inequality (in general) of the valence and sea quark masses [21]. One finds [31]

$$\Delta f_q = -\frac{1+3g^2}{2(4\pi f_\pi)^2} [\bar{h}_q + h_q^I + a^2(\delta'_A h_q^A + \delta'_V h_q^V)], \quad (6)$$

where  $f_\pi \approx 131$  MeV is the pion decay constant,  $g$  is the  $D$ - $D^*$ - $\pi$  coupling [32], and  $\delta'_A$  and  $\delta'_V$  parametrize effects that arise only at nonzero lattice spacing [30]. The terms  $\bar{h}_q$ ,  $h_q^I$ ,  $h_q^A$ , and  $h_q^V$  are functions of the pseudoscalar meson masses. The last two,  $h_q^A$  and  $h_q^V$ , are too cumbersome to write out here. It is instructive to show the other two,  $\bar{h}_q$  and  $h_q^I$ , when  $m_q = m_l$  or  $m_h$ :

$$\bar{h}_q = \frac{1}{16} \sum_{\xi} n_{\xi} [2I(M_{ql,\xi}^2) + I(M_{qh,\xi}^2)], \quad (7)$$

$$h_l^I = -\frac{1}{2}I(M_{ll,I}^2) + \frac{1}{6}I(M_{\eta,I}^2), \quad (8)$$

$$h_h^I = -I(M_{hh,I}^2) + \frac{2}{3}I(M_{\eta,I}^2), \quad (9)$$

where  $I(M^2) = M^2 \ln M^2 / \Lambda_\chi^2$  (with  $\Lambda_\chi$  the chiral scale), and  $M_{\eta,I}^2 = (M_{ll,I}^2 + 2M_{hh,I}^2)/3$ . The term  $h_l^I$  receives contributions only from taste-singlet mesons (representation  $I$ ). The term  $\bar{h}_q$  receives contributions from all representations, with multiplicity  $n_{\xi} = 1, 4, 6, 4, 1$  for  $\xi = P, A, T, V, I$ , respectively. The analytic function is

$$p_q = (2m_l + m_h)f_1(\Lambda_\chi) + m_qf_2(\Lambda_\chi) + O(a^2), \quad (10)$$

where  $f_1$  and  $f_2$  are quark-mass-independent parameters. They are essentially couplings of the chiral Lagrangian, and their  $\Lambda_\chi$  dependence must cancel that of  $\Delta f_q$ . This specifies  $O(a^2)$  terms proportional to  $f_1$  and  $f_2$ , which can be removed after our fit. We estimate the remaining  $O(a^2)$  effects of light quarks to be small: around 4% at  $a = 0.121$  fm and 1.4% at  $a = 0.086$  fm.

The salient feature [33] of the chiral extrapolation of  $\phi_q$  is that  $\Delta f_q$  contains a “chiral log”  $I(2m_q \mu) \sim m_q \ln m_q$ , which has a characteristic curvature as  $m_q \rightarrow 0$ . Equations (4)–(8) show that the chiral log is diluted by discretization effects, because  $a^2 \Delta \xi \neq 0$  for  $\xi \neq P$ .

We can now discuss how we carry out the chiral extrapolation. Recall that we compute  $\phi_q$  for many combinations of the valence and light sea quark masses. At each lattice spacing, we fit all results for  $\phi_q$  to the mass dependence prescribed by Eqs. (4)–(10). Of the 12 parameters, eight— $\mu$ , the four nonzero  $\Delta_{\xi}$ ,  $f_\pi$ ,  $\delta'_A$ , and  $\delta'_V$ —appear in the  $\chi$ PT for light pseudoscalar mesons. We constrain them with prior distributions whose central value and width are taken from the  $\chi$ PT analysis of pseudoscalar meson masses and decay constants on the same ensembles of lattice gauge fields [9]. The rest— $\Phi$ ,  $g^2$ ,  $f_1$ , and  $f_2$ —

appear only for charmed mesons. We constrain  $g^2$  to its experimentally measured value, within its measured uncertainty [34]. Thus, only three parameters— $\Phi$ ,  $f_1$ , and  $f_2$ —are determined solely by the  $\phi_q$  fit. To obtain physical results we reconstitute the fit setting the light sea quark mass  $m_l \rightarrow (m_u + m_d)/2$ , and  $\Delta_{\xi} = \delta'_{A,V} = 0$ . For  $\phi_d$  ( $\phi_s$ ) we set the light valence mass  $m_q \rightarrow m_d$  ( $m_s$ ).

To isolate the uncertainties of the chiral extrapolation from other sources of uncertainty, we consider the ratio  $R_{q/s} = \phi_q/\phi_s$ . Figure 1 shows  $R_{q/s}$  at  $a = 0.121$  fm as a function of  $m_q/m_s$ , projected onto  $m_q = m_l$ . The gray (red) curve is the result of the full fit of  $\phi_q$  to the separate sea- and valence-mass dependence. The black curve, and the extrapolated value at  $m_q/m_s = 0.05$ , results from setting  $\Delta_{\xi} = \delta'_{A,V} = 0$  when reconstituting the fit. At the other lattice spacings we obtain similar results.

The precision after the chiral extrapolation is, however, a bit illusory. We tried several variations in the fit procedure: fitting the ratio directly, adding terms quadratic in the quark masses to Eq. (10), and varying the widths of the prior constraints of the parameters. When these possibilities are taken into account, the extrapolated value of  $R_{d/s}$  varies by 5%, which we take as a systematic uncertainty. This variation could be reduced with higher statistics at the lightest sea quark masses.

The lattice spacing dependence of  $\phi_s = f_{D_s} \sqrt{m_{D_s}}$  is shown in Fig. 2. The (blue) circles are the main results. In a preliminary report of this work [5] the  $O(a^2)$  terms in  $\phi_s$  were not removed. The (red) squares illustrate the effect of omitting this step. As one can see, the effect is small at  $a = 0.086$  fm, but it is the main reason why the results in Eqs. (1) and (2) are smaller than in Ref. [5].

The  $\chi$ PT expressions for  $\phi_q$  assume that the  $D_q$  meson is static. Since its mass is about 1900 MeV and the pseudoscalars are a few hundred MeV, this is a good starting point. Some corrections to this approximation can be absorbed into the fit parameters, with no real change in the analysis. A more interesting change arises in the one-loop self-energy diagrams, for which the function  $I(M^2)$  is modified, and depends on  $m_{D^*} - m_D$  as well as  $M$ . By replacing our standard extrapolation by one using the

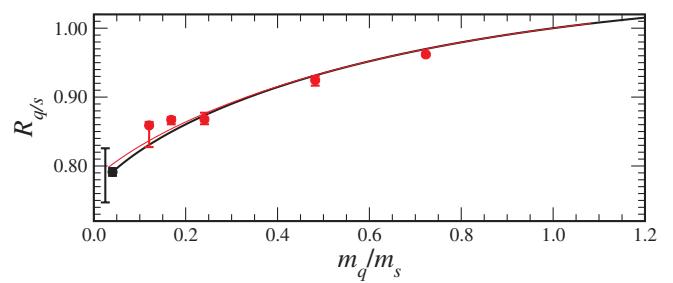


FIG. 1 (color online). Chiral extrapolation of  $R_{q/s}$  at  $a = 0.121$  fm. Data points show only statistical errors, but the systematic error of fitting is shown at left.

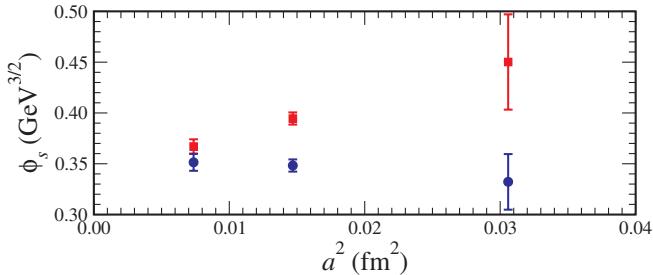


FIG. 2 (color online). Dependence of  $\phi_s$  on  $a^2$ . Circles result from removing the  $O(a^2)$  pieces in Eq. (10); squares omit this step.

modified function, we estimate the associated error to be 1.5% or less. Finite-volume effects also modify  $I(M^2)$ ; based on our experience with  $f_\pi$  and  $f_K$  [9] and on continuum  $\chi$ PT [35], we estimate a further error of 1.5% or less.

Although  $\chi$ PT is able to remove (most of) the light-quark discretization errors, heavy-quark discretization effects remain. We estimate this uncertainty using HQET as a theory of cutoff effects [23,24]. To arrive at a numerical estimate, one must choose a typical scale  $\bar{\Lambda}$  for the soft interactions; we choose  $\bar{\Lambda} \approx 500\text{--}700$  MeV. We then estimate a discretization uncertainty of 2.7%–4.2% at  $a = 0.086$  fm. Similarly, the results at  $a = 0.121$  fm are expected to lie within 1%–2% of those at  $a = 0.086$  fm.

Because we cannot disentangle heavy- and light-quark discretization effects, to quote final results we average the results at  $a = 0.086$  and 0.121 fm. We then find

$$R_{d/s} = 0.786(04)(05)(04)(42), \quad (11)$$

$$\phi_s = 0.349(05)(10)(15)(14) \text{ GeV}^{3/2}, \quad (12)$$

which are the principal results of this work. The uncertainties (in parentheses) are, respectively, from statistics, input parameters  $a$  and  $m_c$ , heavy-quark discretization effects, and chiral extrapolation. A full error budget is in Table II; all uncertainties are reducible in future work. The results for  $f_{D^+}$  and  $f_{D_s}$  in Eqs. (1) and (2) are obtained via  $f_{D_s} = \phi_s / \sqrt{m_{D_s}}$ ,  $f_{D^+} = R_{d/s} \phi_s / \sqrt{m_{D^+}}$ , by inserting the physical meson masses.

Present experimental measurements,  $f_{D^+} = 202 \pm 41 \pm 17$  MeV [4] and  $f_{D_s} = 267 \pm 33$  MeV [25], are not yet precise enough to put our results in Eqs. (1) and (2) to a stringent test. The anticipated measurements of  $f_{D^+}$  and, later,  $f_{D_s}$  from CLEO-c are therefore of great interest. If validated, our calculation of  $f_{D^+}$  has important implications for flavor physics. For  $B$  physics it is crucial to compute the decay constant  $f_B$ . To do so, we must simply change the heavy-quark mass. In fact, heavy-quark discretization effects, with the Fermilab method, are expected to be smaller, about half as big.

TABLE II. Error budget (in percent) for  $R_{d/s}$ ,  $\phi_s$ ,  $\phi_d$ .

Source	$R_{d/s}$	$\phi_s$	$\phi_d$
Statistics	0.5	1.4	1.5
Input parameters $a$ and $m_c$	0.6	2.8	2.9
Higher-order $\rho_{A_4^{Cq}}$	0	1.3	1.3
Heavy-quark discretization	0.5	4.2	4.2
Light-quark discretization and $\chi$ PT fits	5.0	3.9	6.3
Static $\chi$ PT	1.4	0.5	1.5
Finite volume	1.4	0.5	1.5
Total systematic	5.4	6.5	8.5

We thank the U.S. National Science Foundation, the Office of Science of the U.S. Department of Energy, Fermilab, and Indiana University for support, particularly for the computing needed for the project. Fermilab is operated by Universities Research Association Inc., under contract with the U.S. Department of Energy.

Recently, the CLEO-c Collaboration announced a new measurement,  $f_{D^+} = 223 \pm 17 \pm 3$  MeV [36].

- [1] See, for example, the CKM Unitarity Triangle Workshop, <http://ckm2005.ucsd.edu/>.
- [2] V. Lubicz, Nucl. Phys. B Proc. Suppl. **140**, 48 (2005); M. Wingate, *ibid.* **140**, 68 (2005).
- [3] C. T. H. Davies *et al.*, Phys. Rev. Lett. **92**, 022001 (2004).
- [4] G. Bonvicini *et al.*, Phys. Rev. D **70**, 112004 (2004).
- [5] For a preliminary report of this work, see J. N. Simone *et al.*, Nucl. Phys. B Proc. Suppl. **140**, 443 (2005).
- [6] M. Wingate *et al.*, Phys. Rev. Lett. **92**, 162001 (2004).
- [7] For example, A. X. El-Khadra *et al.*, Phys. Rev. D **58**, 014506 (1998); C. Bernard *et al.*, *ibid.* **66**, 094501 (2002).
- [8] C. Aubin *et al.*, Phys. Rev. D **70**, 031504 (2004).
- [9] C. Aubin *et al.*, Phys. Rev. D **70**, 114501 (2004).
- [10] M. Di Pierro *et al.*, Nucl. Phys. B Proc. Suppl. **129**, 328 (2004).
- [11] M. Di Pierro *et al.*, Nucl. Phys. B Proc. Suppl. **129**, 340 (2004).
- [12] C. Aubin *et al.*, Phys. Rev. Lett. **94**, 011601 (2005).
- [13] M. Okamoto *et al.*, Nucl. Phys. B Proc. Suppl. **140**, 461 (2005); C. Aubin *et al.* (to be published).
- [14] I. F. Allison *et al.*, Phys. Rev. Lett. **94**, 172001 (2005).
- [15] M. Ablikim *et al.*, Phys. Lett. B **597**, 39 (2004).
- [16] G. S. Huang *et al.*, Phys. Rev. Lett. **94**, 011802 (2005).
- [17] J. M. Link *et al.*, Phys. Lett. B **607**, 233 (2005).
- [18] D. Acosta *et al.*, hep-ex/0505076.
- [19] C. Bernard *et al.*, Phys. Rev. D **64**, 054506 (2001); C. Aubin *et al.*, *ibid.* **70**, 094505 (2004).
- [20] T. Blum *et al.*, Phys. Rev. D **55**, R1133 (1997); K. Orginos and D. Toussaint, *ibid.* **59**, 014501 (1999); J. Lagae and D. Sinclair, *ibid.* **59**, 014511 (1999); G. P. Lepage, *ibid.* **59**, 074502 (1999); K. Orginos, D. Toussaint, and R. L. Sugar, *ibid.* **60**, 054503 (1999); C. Bernard *et al.*, *ibid.* **61**, 111502 (2000).

- [21] C. Bernard and M. Golterman, Phys. Rev. D **49**, 486 (1994); S.R. Sharpe and N. Shores, *ibid.* **62**, 094503 (2000).
- [22] A.X. El-Khadra, A.S. Kronfeld, and P.B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
- [23] A.S. Kronfeld, Phys. Rev. D **62**, 014505 (2000).
- [24] A.S. Kronfeld, Nucl. Phys. B Proc. Suppl. **129**, 46 (2004).
- [25] S. Eidelman *et al.*, Phys. Lett. B **592**, 1 (2004).
- [26] M. Wingate *et al.*, Phys. Rev. D **67**, 054505 (2003).
- [27] A.X. El-Khadra *et al.*, Phys. Rev. D **64**, 014502 (2001).
- [28] J. Harada *et al.*, Phys. Rev. D **65**, 094513 (2002).
- [29] M. Nobes *et al.* (private communication).
- [30] W. Lee and S. Sharpe, Phys. Rev. D **60**, 114503 (1999); C. Bernard, *ibid.* **65**, 054031 (2002); C. Aubin and C. Bernard, *ibid.* **68**, 034014 (2003); **68**, 074011 (2003).
- [31] C. Aubin and C. Bernard, Nucl. Phys. B Proc. Suppl. **140**, 491 (2005).
- [32] B. Grinstein *et al.*, Nucl. Phys. **B380**, 369 (1992); J.L. Goity, Phys. Rev. D **46**, 3929 (1992).
- [33] A. Kronfeld and S. Ryan, Phys. Lett. B **543**, 59 (2002).
- [34] A. Anastassov *et al.*, Phys. Rev. D **65**, 032003 (2002).
- [35] D. Arndt and C.-J. Lin, Phys. Rev. D **70**, 014503 (2004).
- [36] M. Artuso *et al.*, hep-ex/0508057.

## Lattice QFT with FermiQCD

---

PoS(LAT2005)104

**M. Di Pierro\***

*School of Computer Science, Telecommunications and Information Systems  
DePaul University, Chicago, IL 60604, USA*

**J. M. Flynn**

*School of Physics and Astronomy,  
University of Southampton, Southampton, SO17 1BJ, UK*

FermiQCD is a C++ library for fast development of parallel Lattice Quantum Field Theory computations. It has been developed following a top-down fully Object Oriented design approach with focus on simplicity of use. FermiQCD includes: a heatbath algorithm for Wilson and  $O(a^2)$  improved  $SU(n)$  gauge actions; inversion algorithms for Wilson, Clover, Kogut-Susskind, Asqtad, and Domain Wall fermionic actions; example programs for various types of meson propagators; and converters for the most common gauge file formats.

*XXIIIrd International Symposium on Lattice Field Theory  
25-30 July 2005  
Trinity College, Dublin, Ireland*

---

\*Speaker.

## 1. INTRODUCTION

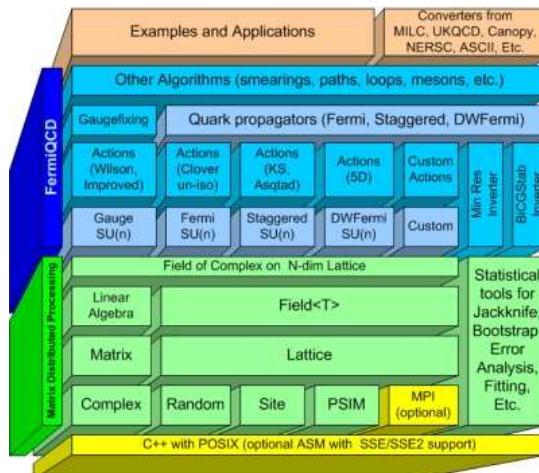
FermiQCD [1, 2, 3] is a library for fast development of parallel applications for Lattice Quantum Field Theories and Lattice Quantum Chromodynamics [4]. It was designed both to be easy to use, with a syntax very similar to common mathematical notation, and, at the same time, optimized for PC clusters.

FermiQCD takes a top-down approach: the top level functions were designed first, followed by optimized implementations of those functions. The most critical parts are optimized in assembler using SSE/SSE2 instructions. All FermiQCD algorithms are parallel but parallelization is hidden from the high level programmer. At the lowest level, parallelization is implemented in MPI and/or in PSIM. PSIM is a parallel emulator that allows running, testing and debugging of parallel code on a single processor PC without requiring MPI.

All components are implemented as separate, although related, classes. For example, in FermiQCD lattices and fields are objects while actions and inverters are static methods of the corresponding classes. FermiQCD components range from low level linear algebra, fitting and statistical functions (including the Bootstrap and a Bayesian fitter based on Levenberg-Marquardt minimisation) to high level parallel algorithms specifically designed for lattice quantum field theories such as the Wilson [5] and  $O(a^2)$ -improved gauge actions, the Clover fermionic action, the Asqtad [6] action for KS fermions, and the Domain Wall action [7].

One can create a new action by creating a new class and plugging it into the rest the library. All other components, such as the inverters, will work with it. For example FermiQCD provides three inverters, MinRes, BiCGStab and UML. The first two are general and work with any action and any type of field, the third (UML) is highly optimized for KS and ASQTAD actions.

Figure 1 shows a schematic representation of FermiQCD's components. The lower components are referred to as Matrix Distributed Processing and they define the language used in FermiQCD. The upper components are the algorithms. The top components represent examples, ap-



**Figure 1:** Components of FermiQCD.

plications and other tools. These tools include converters for the most gauge field formats: MILC, NERSC, UKQCD, CANOPY, and some binary formats.

## 2. SYNTAX OVERVIEW AND PROGRAM EXAMPLE

All FermiQCD algorithms are implemented on top of an Object Oriented Linear Algebra package with a Maple-like syntax. For example

$$A = \text{trace}(\gamma^2(\gamma^0 - 1)/2) \quad (2.1)$$

$$B = e^{i\theta\lambda_3} \quad (2.2)$$

where  $\gamma^\mu$  are the Dirac Gamma matrices in Euclidean space and  $\lambda_3$  is one of the generators of  $SU(3)$ , are implemented in FermiQCD as

```
Complex A = trace(Gamma[i]*(Gamma[0]-1)/2);
Matrix B = exp(I*theta*Lambda[3]);
```

A 4D  $16 \times 8^3$  lattice is declared (with obvious generalization to arbitrary dimensions and sizes) as

```
int L[]={16,8,8,8};
mdp_field lattice(4,L);
```

An  $SU(n)$  gauge field  $U_\mu(x)$  is declared and initialized by

```
gauge_field U(lattice,n); set_cold(U);
```

The following sets  $\beta = 6.0$  and performs 10 heatbath [8] steps with the Wilson gauge action

```
coefficients gauge; gauge["beta"]=6.0;
WilsonGaugeAction::heatbath(U,gauge,10);
```

Any field can be saved: `U.save("filename")`; loaded: `U.load("filename")`; and translated: `U.shift(mu)`. A field can also be transformed locally. Here is how to implement a global gauge transformation  $G$  of  $U$

```
Matrix G=exp(2*I*Lambda[2]);
mdp_site x(lattice);
forallsites(x)
  for(int mu=0; mu<U.ndim; mu++)
    U(x,mu)=G*U(x,mu)*inv(G);
```

`U(x,mu)` is an  $n \times n$  matrix and `x` is an object that represents a lattice site. `forallsites(x)` is a parallel loop. Each processing node loops over the lattice sites stored by the node.

A Wilson fermionic field is declared as

```
fermi_field psi(lattice,n);
```

and a gauge invariant shift can be implemented as

```
psi_up(x)=U(x,mu)*psi(x+mu);
psi_dw(x)=hermitian(U(x-mu,mu))*psi(x-mu);
```

Notice that  $x+\hat{\mu}$  reads as  $x + \hat{\mu}$  and  $x-\hat{\mu}$  reads as  $x - \hat{\mu}$  where  $\mu = 0, 1, 2, 3$  is one of the possible lattice directions.

Multiplication by the fermionic matrix is invoked as follows

```
coefficients quark; quark["kappa"]=0.1245; quark["c_{SW}"]=0.0;
if(quark["c_{SW}"]!=0) compute_em_field(U);
default_fermi_action=FermiCloverActionFast::mul_Q;
mul_Q(psi_out,psi_in,U,quark);
```

The chromo-electromagnetic field is required by the clover term and computed only if required. It is stored inside a gauge field object. FermiQCD includes three different equivalent implementations of the above algorithm, declared in the following classes: FermiCloverActionSlow, FermiCloverActionFast, FermiCloverActionSSE2. The second is optimized in C++, the third is optimized in assembler.

The inverse multiplication  $\psi_{\text{out}} = Q^{-1}[U]\psi_{\text{in}}$  is invoked with the following call

```
mul_invQ(psi_out,psi_in,U,quark,1e-20,1e-12);
```

where  $1e-20$  is the target absolute precision for the numerical inversion and  $1e-12$  is the target relative precision.

An ordinary quark propagator is declared and generated by

```
fermi_propagator S(lattice,n);
generate(S,U,quark,1e-20,1e-12);
```

and it can be used, for example, to build a meson propagator  $C_\pi(t)$  by summing the following expression over  $x$  and over the spin components  $a, b$

```
Cpi[x(TIME)]+=real(trace(S(x,a,b)*hermitian(S(x,a,b))));
```

Everything works similarly for the other actions and other types of fields. Figure 2 shows a complete parallel program for generating `nconfig` gauge configurations in  $SU(5)$  on a  $16 \times 8^4$  lattice, saving them and computing the average plaquette and the pion propagator on each configuration.

### 3. BENCHMARKS

Table 1 shows typical running times for the FermiQCD inverters applied to different actions. Times are in microsecond per lattice site per step. Notice that MinRes involves one `mul_Q` per step, BiCGStab involves two, and the UML inverter also involves two but only applied to sites of even parity. These times were computed on one 3.2GHz Pentium 4 (typical computations in parallel with a Myrinet network show a drop in efficiency of 20-30% when scaling up to 16-32 processors).

```

1 #include "fermiqcd.h"
2 int main(int argc, char **argv) {
3   mdp.open_wormholes(argc, argv); // START
4   define_base_matrices("FERMIQCD"); // set Gamma convention
5   int n=5, nconfig=100;
6   int L[] = {16,8,8,8};
7   mdp_lattice lattice(4,L);           // declare lattice
8   gauge_field U(lattice, n);         // declare fields
9   fermi_propagator S(lattice,n);     // declare propagator
10  mdp_site x(lattice);              // declare a site var
11  coefficients gauge; gauge["beta"]=6.0; // set parameters
12  coefficients quark; quark["kappa"]=0.1234; quark["c_{SW}"]=0.0;
13  default_fermi_action=FermiCloverActionFast::mul_Q;
14  mdp_array<float,1> Cpi(L[TIME]); // declare and zero Cpi
15  for(int t=0; t<L[TIME]; t++) Cpi(t)=0;
16  set_hot(U);
17  for(int k=0; k < nconfig; k++) {
18    WilsonGaugeAction::heatbath(U,gauge,10); // do heatbath
19    mdp << average_plaquette(U) << endl;      // print plaquette
20    U.save(string("gauge")+tostring(k));        // save config
21    if(quark["c_{SW}"]!=0) compute_em_field(U);
22    generate(S,U,quark,1e-20,1e-12);            // make propagator
23    forallsites(x)                            // contract pion
24      for(int a=0; a<4; a++)                  // source spin
25        for(int b=0; b<4; b++)                // sink spin
26          Cpi[x(TIME)]+=real(trace(S(x,a,b)*hermitian(S(x,a,b))));
27    mpi.add(Cpi.address(),Cpi.size());        // parallel add
28    for(int t=0; t<L[TIME]; t++)
29      mdp << t << " " << Cpi(t) << endl; // print output
30  }
31  mdp.close_wormholes();                  // STOP
32  return 0;
33 }

```

**Figure 2:** A complete parallel program in FermiQCD.

Action	Inverter	float	float (SSE)	double	double (SSE2)
Wilson	Min Res	8.83	1.79	6.84	2.07
Wilson	BiCGStab	17.8	3.16	13.8	4.42
Clover	Min Res	9.76	1.98	12.08	2.82
Clover	BiCGStab	19.63	4.71	24.95	6.08
KS	Min Res	1.42	0.78	1.71	1.01
KS	BiCGStab	2.95	1.63	3.56	2.11
KS	UML	1.89	1.14	2.08	1.34
Asqtad	Min Res	3.73	2.47	4.29	5.24
Asqtad	BiCGStab	7.65	5.02	8.79	6.61
Asqtad	UML	1.14	3.14	5.24	3.81

**Table 1:** Running times for FermiQCD inverters using different actions.

## 4. CONCLUSIONS

FermiQCD is now a stable and mature product and the project mailing list currently numbers more than 30 members. The Wilson and Asqtad inverters are as fast if not faster than any other software package available for PC clusters. FermiQCD is an Open Source project and users can contribute to its improvement by creating new classes and adding functionality. Some of our objectives include the addition of an optimized gauge action, optimized Domain Wall fermions, HMC for dynamical fermions, compatibility with the ILDG format [9], support for the SciDAC QMP API, and a GUI for visual development.

FermiQCD and additional documentation can be downloaded from: [www.fermiqcd.net](http://www.fermiqcd.net)

## Acknowledgements

We wish to acknowledge the Fermilab theory group, the University of Southampton, and the University of Iowa for their contribution to the development of FermiQCD.

## References

- [1] M. Di Pierro, arXiv:hep-lat/0011083.
- [2] M. Di Pierro, Nucl. Phys. Proc. Suppl. **106**, 1034 (2002) [arXiv:hep-lat/0110116].
- [3] M. Di Pierro *et al.* [FermiQCD Colaboration], Nucl. Phys. Proc. Suppl. **129**, 832 (2004) [arXiv:hep-lat/0311027].
- [4] arXiv:hep-lat/0509013
- [5] K. G. Wilson, Phys. rev. **D10** (1974) 2445
- [6] C. Bernard *et al.* [MILC Collaboration], Phys. Rev. D **58** (1998) 014503 [hep-lat/9712010].
- [7] D. B. Kaplan, Phys. Lett. **B288** (1992) 342
- [8] M. Creutz, Phys. Rev. **D21** 2308 (1980)
- [9] B. Joo and W. Watson, Nucl. Phys. Proc. Suppl. **140**, 209 (2005) [arXiv:hep-lat/0409165].



Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT<sup>®</sup>

Nuclear Physics B (Proc. Suppl.) 140 (2005) 443–445

NUCLEAR PHYSICS B  
PROCEEDINGS  
SUPPLEMENTS

[www.elsevierphysics.com](http://www.elsevierphysics.com)

## Leptonic decay constants $f_{D_s}$ and $f_D$ in three flavor lattice QCD

J.N. Simone<sup>a\*</sup>, C. Aubin<sup>b</sup>, C. Bernard<sup>b</sup>, C. DeTar<sup>c</sup>, M. di Pierro<sup>d</sup>, A.X. El-Khadra<sup>e</sup>, Steven Gottlieb<sup>f</sup>, E.B. Gregory<sup>g</sup>, U.M. Heller<sup>h</sup>, J.E. Hetrick<sup>i</sup>, A.S. Kronfeld<sup>a</sup>, P.B. Mackenzie<sup>a</sup>, D.P. Menscher<sup>e</sup>, M. Nobes<sup>j</sup>, M. Okamoto<sup>a</sup>, M.B. Oktay<sup>e</sup>, J. Osborn<sup>c</sup>, R. Sugar<sup>k</sup>, D. Toussaint<sup>g</sup>, and H.D. Trottier<sup>j</sup>

<sup>a</sup>Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510, USA

<sup>b</sup>Department of Physics, Washington University, St. Louis, MO 63130, USA

<sup>c</sup>Physics Department, University of Utah, Salt Lake City, UT 84112, USA

<sup>d</sup>School of Computer Science, DePaul University, Chicago, IL 60604, USA

<sup>e</sup>Physics Department, University of Illinois, Urbana, IL 61801, USA

<sup>f</sup>Department of Physics, Indiana University, Bloomington, IN 47405, USA

<sup>g</sup>Department of Physics, University of Arizona, Tucson, AZ 85721, USA

<sup>h</sup>American Physical Society, One Research Road, Box 9000, Ridge, NY 11961, USA

<sup>i</sup>University of the Pacific, Stockton, CA 95211, USA

<sup>j</sup>Physics Department, Simon Fraser University, Burnaby, BC, Canada

<sup>k</sup>Department of Physics, University of California, Santa Barbara, CA, 93106, USA

We determine the leptonic decay constants  $f_{D_s}$  and  $f_D$  in three flavor unquenched lattice QCD. We use  $O(a^2)$ -improved staggered light quarks and  $O(a)$ -improved charm quarks in the Fermilab heavy quark formalism. Our preliminary results, based upon an analysis at a single lattice spacing, are  $f_{D_s} = 263^{+5}_{-9} \pm 24$  MeV and  $f_D = 225^{+11}_{-13} \pm 21$  MeV. In each case, the first reported error is statistical while the second is the combined systematic uncertainty.

### 1. INTRODUCTION

The leptonic decay constants  $f_{B_s}$  and  $f_B$  are critical in testing the flavor sector of the Standard Model. Reliable lattice calculations are of fundamental importance since the determination of these decay constants remains beyond the reach of experiment.

Precise experimental determinations of the decay constants  $f_{D_s}$  and  $f_D$  and the semileptonic decays  $D \rightarrow \pi \ell \nu$  and  $D \rightarrow K \ell \nu$  will result from the high-statistics charm program of CLEO-c. Comparing these experimental results with lattice results will serve both as a critical check of

lattice methods for charm and as a means of assessing the reliability of lattice calculations for the bottom quark.

This work calculates the leptonic decay constants  $f_{D_s}$  and  $f_D$  using  $O(a^2)$ -improved staggered light quarks and  $O(a)$ -improved charm quarks in the Fermilab heavy quark formalism [1]. It was done with three flavors of light sea quarks. The staggered fermion action for the light quarks makes possible calculations at lighter quarks masses than previously used [2], allowing a better controlled chiral extrapolation. We use the results of staggered, partially quenched chiral perturbation theory ( $S\chi PT$ ) in performing the chiral extrapolations.

\*talk presented by J. Simone

## 2. $f_D \sqrt{m_D} / f_{D_s} \sqrt{m_{D_s}}$ RATIO

We use the MILC collaboration Asqtad gauge ensembles with lattice spacing  $a \approx 1/8$  fm [3]. Each ensemble has one sea quark flavor of mass,  $m_s$ , approximating the strange quark and two flavors, degenerate in mass,  $m_u$ . The five ensembles we use have light flavors in the range  $0.1m_s \leq m_u \leq 0.6m_s$ . For each ensemble, decay constants were computed at twelve logarithmically spaced values of the valence quark mass in the range  $0.1m_s \leq m_q \leq m_s$ . In all, decay constants were computed for sixty partially quenched ( $m_q, m_u$ ) combinations.

The chiral expansion for the decay constants is known at leading order in both the heavy quark expansion and staggered chiral perturbation theory [4]. We take the parameterization

$$R_{q/s} = 1 + a_0(\Delta f_q - \Delta f_s) + (m_q - m_s) \times (a_1 + a_2\tilde{m} + a_3m_q + a_4m_s + \dots) \quad (1)$$

for the ratio  $R_{q/s} = f_{D_q} \sqrt{m_{D_q}} / f_{D_s} \sqrt{m_{D_s}}$  in our chiral extrapolation. For staggered quarks, the chiral log terms  $\Delta f_x$  contain discretization effects from taste violations in the pseudoscalar masses as well as explicit taste violation terms. We include quark mass terms up to  $O(m^2)$  with the constraint  $a_4 = a_3 - a_1^2$ . The sum of sea quark masses is  $\tilde{m} \equiv 2m_u + m_s$ .

The parameters  $a_j$  in Eq. (1) are determined in a fully covariant  $\chi^2$  minimization using all sixty partially quenched decay constant results. Pseudoscalar masses and coefficients of the explicit taste violation terms were fixed to the values determined in an analysis of the light mesons [5]. Bayesian priors were input for each  $a_j$ . With  $\xi = 1/(4\pi f)^2$ , the prior for  $a_0$  is  $-0.5\xi(1 + 3g^2)(1 \pm 0.30)$  with  $g^2 \approx 0.35$ . Other priors are  $0 \pm 1$  in units of  $2\xi s$  where  $s$  is the slope relating the quark mass and the pion mass squared.

The fit, shown in Fig. 1, has  $\chi^2/\text{dof} = 0.2$ . The extrapolation according to S $\chi$ PT is shown along the “full QCD” direction where the valence quark mass equals the sea quark mass. The parameters in this extrapolation are determined using all sixty partially-quenched points. We obtain  $R_{d/s} = 0.833^{+0.042}_{-0.036}$  in the chiral limit. The error is the combined uncertainty from statistics and

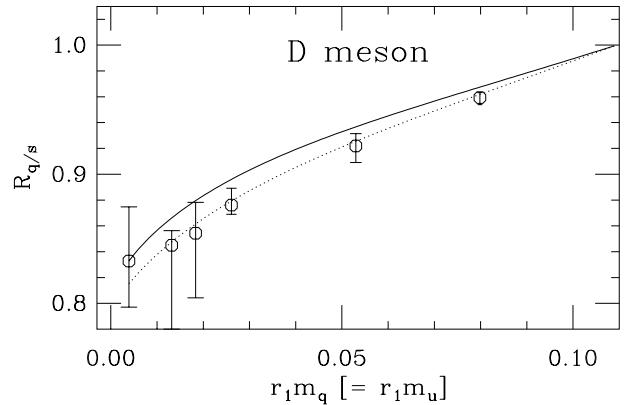


Figure 1. The chiral extrapolation of the ratio  $f_D \sqrt{m_D} / f_{D_s} \sqrt{m_{D_s}}$ . The ratio is determined by the solid curve where staggering effects have been removed. The dashed curve includes these discretization effects.

parameter prior estimates for the extrapolation function. Systematic effects from matching the lattice theory to QCD, the lattice spacing and the tuning of charm quark mass mostly cancel in the ratio. Residual discretization effects from light quarks, not removed by the extrapolation procedure, are estimated to be 4%. Our estimate is based on the size of known taste-breaking effects shown in Fig. 1 and is similar to taste-breaking effects found in  $f_K$  and  $f_\pi$  [5]. The ratio was determined using the nominal strange quark mass rather than the tuned value which leads to an uncertainty of 2%.

## 3. $f_{D_s} \sqrt{m_{D_s}}$ DETERMINATION

We obtain  $f_{D_s} \sqrt{m_{D_s}}$  by first interpolating to the tuned valence  $m_s$  value obtained from the light mesons on the same gauge ensembles. Then, a mild sea quark extrapolation (see Fig. 2) is needed to obtain  $f_{D_s}$ . We extrapolate linearly to the tuned  $\hat{m} = (m_u + m_d)/2$  value obtained from the light mesons [5]. The combined statistical and extrapolation error is 3.3%. The uncertainty from the tuning of  $m_s$  and  $\hat{m}$  is 1%. The tuning of the charm mass leads to a 4% error. The lattice spacing uncertainty is 2% [6]. The

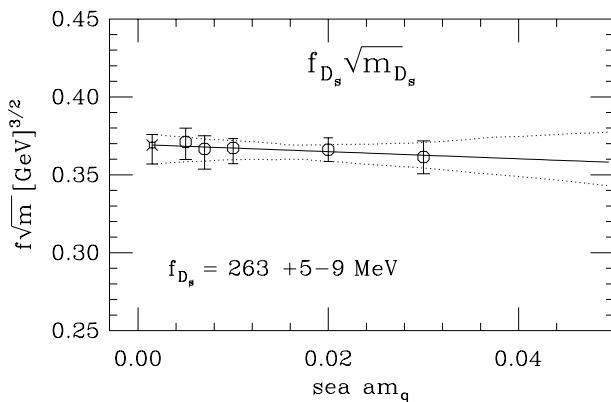


Figure 2. Extrapolation in the light sea quark mass for  $f_{D_s}\sqrt{m_{D_s}}$ . The curves show a linear fit (solid) and the 68% confidence level statistical error bounds (dotted).

dominant systematic uncertainty, 7%, is from the mismatch between the lattice theory and QCD, as discussed in Ref. [7]. Our final results will include an improved estimate of this uncertainty incorporating results from finer and coarser lattice spacings, which are now in progress.

#### 4. RESULTS

Statistical and systematic uncertainties are summarized in Table 1. Our estimates of heavy quark matching effects and light quark discretization effects are based on results from a single lattice spacing. We will refine our error estimates and update our results once decay constants from additional lattice spacings are known. The heavy quark matching uncertainty can be reduced by including the higher order matchings for the action and the currents [8,9].

Combining in quadrature the systematic uncertainties shown in Table 1, we find our preliminary results:

$$\begin{aligned} \frac{f_{D_s}\sqrt{m_{D_s}}}{f_D\sqrt{m_D}} &= 1.20 \pm 0.06 \pm 0.06, \\ f_{D_s} &= 263^{+5}_{-9} \text{ MeV}, \\ f_D &= 225^{+11}_{-13} \text{ MeV}. \end{aligned}$$

Table 1  
Error budget as percentage of each quantity.

source	$R_{d/s}$	$f_{D_s}\sqrt{m_{D_s}}$	$f_D\sqrt{m_D}$
stat.+extrap.	4.7	3.3	6.2
HQ matching	<1	7	7
LQ discret.	4	4	4
$m_c$ det.	<1	4	4
val. $m_s$ , $m_d$	2	1	2.2
$a$ & sea quark	<1	2	2

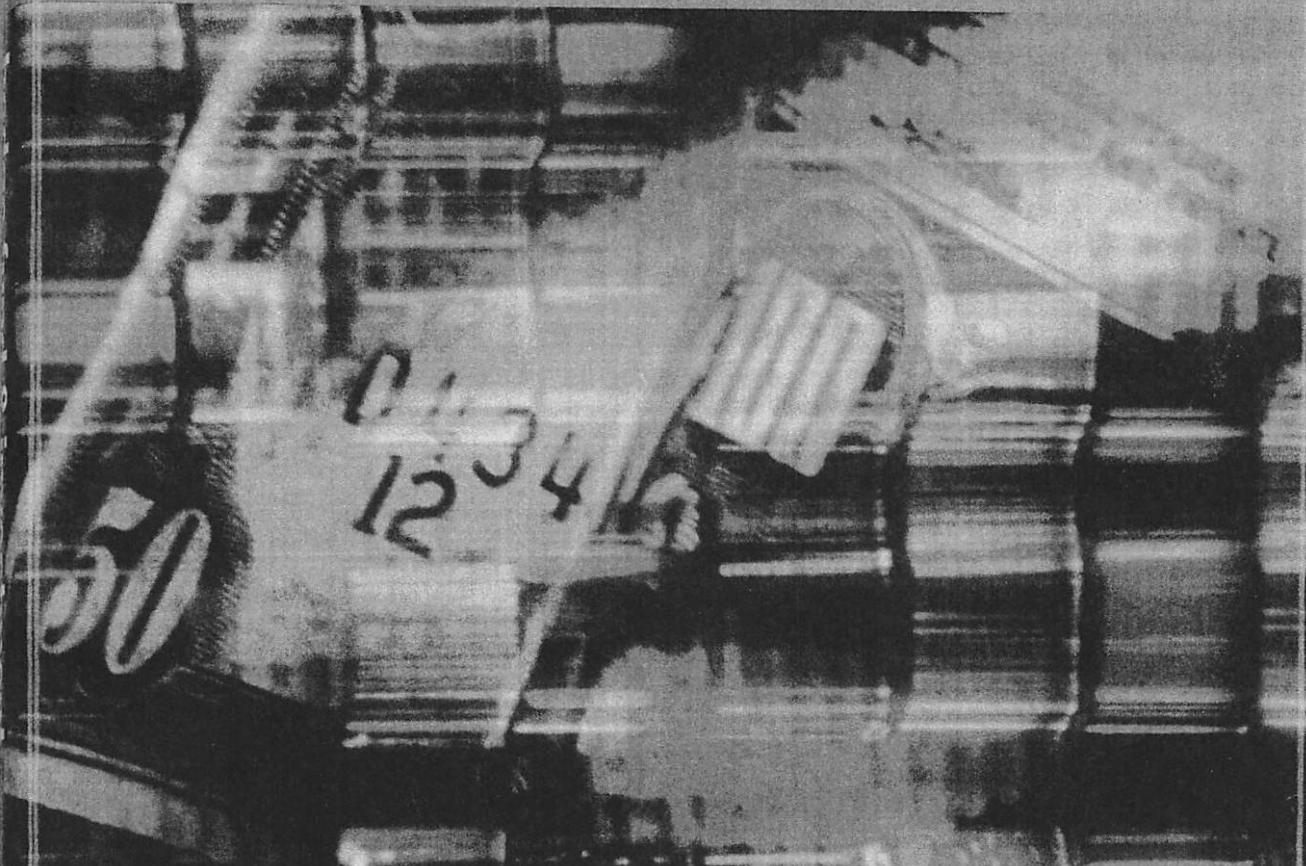
#### ACKNOWLEDGMENTS

We acknowledge support through the DOE and NSF high energy physics programs. We thank the DOE SciDAC program and Fermilab Computing Division for support. Fermilab is operated by Universities Research Association Inc. under contract with the United States Department of Energy.

#### REFERENCES

1. A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997) [arXiv:hep-lat/9604004].
2. C. Bernard *et al.* [MILC Collaboration], arXiv:hep-lat/0410014.
3. C. Aubin *et al.*, arXiv:hep-lat/0402030.
4. C. Aubin and C. Bernard, arXiv:hep-lat/0409027.
5. C. Aubin *et al.* [MILC Collaboration], arXiv:hep-lat/0407028.
6. C. T. H. Davies *et al.* [HPQCD, MILC and Fermilab Lattice Collaborations], Phys. Rev. Lett. **92**, 022001 (2004) [arXiv:hep-lat/0304004].
7. C. Aubin *et al.* [Fermilab Lattice, MILC and HPQCD Collaborations], arXiv:hep-ph/0408306.
8. M. A. Nobes and H. D. Trottier, Nucl. Phys. Proc. Suppl. **129**, 355 (2004) [arXiv:hep-lat/0309086].
9. M. B. Oktay, A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Nucl. Phys. Proc. Suppl. **129**, 349 (2004) [arXiv:hep-lat/0310016].

ELSEVIER FINANCE



# **hedge fund & investment management**

EDITED BY

Izzy Nelken



**Hari P. Krishnan** is an executive director and co-director of alternative asset allocation at Morgan Stanley. He runs over \$1 billion of advisory capital for high net worth individuals, family offices and institutions. He was previously an options strategist at a market making firm at the CBOE and a senior economist at the Chicago Board of Trade. Hari has a BA in math from Columbia, an MSc and PhD in applied math from Brown and did postdoctoral work at the Columbia Earth Institute.

**Jack W. Mosevich** joined the financial industry in 1986 at Merrill Lynch after several years as a professor of Mathematics and Computer Science. His main areas of expertise are quantitative finance, risk management, derivatives analytics and portfolio construction.

Jack's experience has been equally divided between the buy-side and sell-side. He has worked in both large corporations such as UBS, Merrill Lynch and Burns Fry, as well as smaller firms such as Stafford Capital Management and Contego Capital Management. Jack is currently a Clinical Professor of Finance in the College of Commerce at DePaul University. His recent research is concentrated on risk management and portfolio construction in both traditional and especially in the alternative asset management areas. Jack is also a consultant with MetaCryption Quantitative Finance.

In addition to his core employment Jack has been a part-time instructor at the University of Chicago Program on Financial Mathematics since its inception in 1997.

Jack possesses a Ph.D. degree in Mathematics from the University of British Columbia.

**Izzy Nelken** is president of Super Computer Consulting, Inc. in Northbrook, Illinois. Super Computer Consulting Inc. specializes in complex derivatives, structured products, risk management and hedge funds. Izzy holds a Ph.D. in Computer Science from Rutgers University and was on the faculty at the University of Toronto. Izzy's firm has many consulting clients including several regulatory bodies, major broker-dealers, large and medium sized banks as well as hedge funds. Izzy is a lecturer at the prestigious mathematics department at the University of Chicago. He teaches numerous courses and seminars around the world on a variety of topics. Izzy's seminars are known for being non mathematical. Instead they combine cutting edge analytics with real world applications and intuitive examples.

**Massimo Di Pierro** is an expert in numerical and quantitative methods applied to scientific and financial modeling. He is one of the founders and owners of MetaCryption LLC.

Dr. Di Pierro is currently full-time Assistant Professor at the School of Computer Science, Telecommunications and Information Systems of DePaul University in

Chicago. He teaches graduate students regularly, and topics include Monte Carlo Simulations, Parallel Algorithms, Network Programming, and Computer Security. Dr. Di Pierro is one of the leading developers of the Master of Science in Computation Finance at DePaul.

He has published more than 20 papers in different fields and a number of software products including MCQF (a software library for financial analysis) [www.fermiqcd.net](http://www.fermiqcd.net) (a toolkit for parallel large scale grid-like computations), Spider (a web content manager used by the United Nations).

Dr. Di Pierro earned a Ph.D in Physics from the University of Southampton in UK and has worked for three years as Associate Researcher at Fermilab.

**Ms. Rachlin** is a Managing Director, and is a member of the Asset Allocation and Risk Management team and the Investment Committee at Mariner Investment Group, Inc. Ms. Rachlin was formerly a Director and founding member of Deerfield International Administrative Services, Ltd. Her responsibilities included overseeing sales, marketing and product development. Prior to Deerfield, Ms. Rachlin was Co-Head of the IBL Agent Department at both New Japan Securities International and Aubrey G. Lanston & Co., Inc. overseeing sales and trading of international fixed income products into the Americas. Prior thereto, she was a Managing Director and a Director at S.G. Warburg and Co., Inc. and S.G. Warburg, plc. in the fixed income department. Ms. Rachlin also traded fixed-income arbitrage for 5 years at Citibank, N.A. and Government Arbitrage Co. She has written several chapters for financial textbooks edited by Frank J. Fabozzi on economics and investment management, as well as other articles for finance journals. She holds an AB economics degree cum laude from Cornell University, an MBA. specializing in finance from the University of Chicago and an MA – creative writing from Antioch University McGregor. She serves as Board Member and Treasurer of the Poetry Society of America.

**Robert Sherak** is the founder, portfolio manager, and CEO of The Midway Group, and a hedge fund manager founded in 2000, based in New York City. Since 1976, Bob has been involved with fixed income securities, particularly mortgage backed securities, as a portfolio manager, trader, research analyst, and a programmer. His academic training was in cognitive psychology (memory, linguistics, decision making, and artificial intelligence) and computer science.

**Hilary Till** co-founded Premia Capital Management, LLC (<http://www.premiacap.com>) in 1998 with Joseph Eagleeye. Chicago-based Premia Capital specializes in detecting pockets of predictability in derivatives markets using statistical techniques.

She is also a principal of Premia Risk Consultancy, Inc., which advises investment firms on derivatives strategies and risk management policy.

# Chapter 9

## On ranking schemes and portfolio selection

MASSIMO DI PIERRO AND JACK W. MOSEVICH

### ABSTRACT

There are now in use several risk-return indicators, which are utilized to rank historical returns of portfolios. Some popular ones are the Sharpe Ratio, the Sortino Ratio, Omega and the Stutzer Index, among others. It is well known that portfolio log-returns, especially of alternative assets, are not normally (Gaussian) distributed. This is the reason for the development of indicators other than the Sharpe Ratio. The purpose of this paper is to evaluate the relationships between these indicators for both Gaussian and non-Gaussian distributions. We prove mathematically that rankings are essentially the same for these indicators in a Gaussian environment, and different in a non-Gaussian one, which is as it should be. We are able to compute an implied utility function for the indicators and find that it is the same for all of them, something not very intuitive. We then propose a utility function, which corresponds more with what we expect investors to desire. We conclude by showing how to relate our results to the Markowitz MPT.

### 9.1 INTRODUCTION

In this paper we discuss different criteria for ranking portfolios including the Sharpe ratio (Sharpe, 1964), the Sortino ratio (Sortino and Van Der Meer, 1991; Sortino and Price, 1994; Sortino and Forsey, 1996), the kappa ratio (Kaplan and Knowles, 2004), the omega ratio (Shadwick and Keating, 2002; Sortino, 2001; Wilmott, 2000) and the Stutzer index (Amenc, Malaise, Martellini and Vaisse). We prove that in a world where portfolio returns are Gaussian distributions, all of the above ranking systems are equivalent in the sense that although they produce different numbers they will produce the same ranking order. We also prove that all of

the above ranking systems implicitly assume a non-natural utility function that attributes the same utility to any positive return (utility equals to +1) and to all negative returns (utility equals to -1).

We propose a more natural utility function from which we derive a different ranking system for Gaussian portfolios that is not equivalent to the Sharpe ratio or any of the other rankings considered. Using the Berry–Esseen theorem we prove that our ranking system, embodied in equation (24), is applicable to portfolios with non-Gaussian returns under the condition that one plans to hold the portfolio for a sufficiently long time.

Finally we show how to apply our findings to Markowitz' Modern Portfolio Theory (Markowitz, 1952; Wilmott, 2000).

## 9.2 CONVENTIONS AND DEFINITIONS

Given a portfolio  $A$  whose historic values are  $\{S_i\}$  we will indicate with  $p(x) : R \rightarrow R^*$  the probability mass function of each of the random variables  $x_i = \log(S_{i+1}/S_i)$ . The probability mass function is normalized to 1. We also define as  $F(x) = \int_{-\infty}^x p(z)dz$  the usual cumulative distribution.

Throughout this paper we will also assume that  $r$  is the risk-free interest rate.

**Definition: (Ranking).** We define a ranking as a functional  $R(p) : R \times R^* \rightarrow R$  that maps the probability mass function  $p$  associated to a portfolio into a real number.

**Definition: (Equivalence).** We say that two rankings  $R_1$  and  $R_2$  are equivalent in a domain  $D$  if there exists a monotonic increasing function  $h$  such that for every probability mass function  $p$  in the domain  $D$ ,  $R_1(p) = h(R_2(p))$ .

If two rankings are equivalent we will use the notation

$$R_1 \sim R_2$$

The equivalence relation as defined is symmetric and transitive.

**Definition: (Sharpe ratio).** Given a portfolio characterized by a probability mass function  $p(x)$  the Sharpe (1964) ratio is defined as

$$R_{\text{Sharpe}}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{\sigma} \quad (1)$$

where

$$\mu = \int_{-\infty}^{+\infty} xp(x)dx \quad (2)$$

$$\sigma = \sqrt{\int_{-\infty}^{+\infty} (x - \mu)^2 p(x) dx} \quad (3)$$

Herein we denote the Sharpe ratio by  $y$ , with no explicit reference to the risk-free rate  $r$ . Note that in the equations which follow, the letter  $r$  can also represent a minimal acceptable return.

**Definition:** (*Sortino ratio*). Given a portfolio characterized by a probability mass function  $p(x)$  the Sortino ratio (Sortino and Van Der Meer, 1991; Sortino and Price, 1994; Sortino and Forsey, 1996) is defined as

$$R_{\text{Sortino}}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{\sqrt{\int_{-\infty}^r (r - x)^2 p(x) dx}} \quad (4)$$

**Definition:** (*Kappa-n ratio*). Given a portfolio characterized by a probability mass function  $p(x)$  the kappa ratio (Kaplan and Knowles, 2004) is defined as

$$R_{\text{kappa}-n}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{\left[ \int_{-\infty}^r (r - x)^n p(x) dx \right]^{1/n}} \quad (5)$$

Note that for  $n = 2$ ,  $R_{\text{kappa}-2}(p, r) \equiv R_{\text{Sortino}}(p, r)$ .

**Definition:** (*Omega ratio*). Given a portfolio characterized by a probability mass function  $p(x)$  the omega ratio (Shadwick and Keating, 2002; Sortino, 2001; Wilmott) is defined as

$$R_{\text{Omega}}(p) \stackrel{\text{def}}{=} \frac{\int_{-\infty}^r (1 - F(x)) dx}{\int_{-\infty}^r F(x) dx} \quad (6)$$

**Definition:** (*Stutzer index*). Given a portfolio characterized by a probability mass function  $p(x)$  the Stutzer index (Amenc, Malaise, Martellini and Vaisse) is defined as

$$R_{\text{Stutzer}}(p) \stackrel{\text{def}}{=} \lim_{T \rightarrow \infty} \frac{-\log F(rT)}{T} \quad (7)$$

The Stutzer index ranks portfolios according to the speed with which the probability of a negative return (when compared with  $rT$ ) tends to zero when time grows ( $T \rightarrow \infty$ ).

### 9.3 EQUIVALENCE IN A GAUSSIAN WORLD

In this section we will restrict to only domain

$$D = \left\{ p \mid p(x) = p_{\text{Gaussian}}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \right\} \quad (8)$$

portfolios having a Gaussian distributions of returns.

**Theorem 1.**  $R_{\text{Sortino}} \sim R_{\text{Sharpe}}$ .

*Proof.* By explicit integration<sup>1</sup>

$$R_{\text{Sortino}}(p) = h_1(R_{\text{Sharpe}}(p))$$

Where

$$h_1(y) = \frac{\sqrt{2}y}{\sqrt{1 - \sqrt{2}/\pi} e^{-y^2/2} y + y^2 - (1 + y^2) \operatorname{erf}(y/\sqrt{2})} \quad (9)$$

Recall that  $y = (\mu - r)/\sigma$  is the Sharpe ratio. Since  $h_i'(y) > 0$  for every finite real  $y$  proves the equivalence.

Figure 1 shows a plot of  $h_1$  and  $h'_1$ . It also shows that compared with the Sharpe ratio, the Sortino is relatively sensitive to changes of  $y$  for large values of  $y$ , but it becomes insensitive to  $y$  for negative values of  $y$ .

**Theorem 2.**  $R_{\kappa - n} \sim R_{\text{Sharpe}}$  for every  $n$ .

*Proof.* By explicit integration

$$R_{\kappa-n}(p) = h_2(R_{\text{Sharpe}}(p)) \quad (10)$$

$$h_2(y) = \frac{\pi^{1/(2n)} y}{\left[2^{(n-2)/2} e^{-y^2/2} g(y)\right]^{1/n}}$$

$$g(y) = \Gamma\left(\frac{1+n}{2}\right) {}_1F_1\left(\frac{1+n}{2}, \frac{1}{2}, y^2/2\right) - \sqrt{2}y {}_1F_1\left(\frac{1+n}{2}, \frac{3}{2}, y^2/2\right)$$

and  ${}_1F_1(a, b, x)$  is a member of the family of hypergeometric functions.  $h'_2(y) > 0$  for every even integer  $n$  and every finite real  $y$ . Figure 2 shows a plot of  $h_2$  and  $h'_2$  for  $n = 1, 2, 3$ . It also shows how the  $K_n$  ratio exhibits the same sensitivity to  $y$  as the Sortino does, but the higher the value of  $n$ , the lower the sensitivity to  $y$ .

**Theorem 3.**  $R_{\text{Omega}} \sim R_{\text{Sharpe}}$ .

*Proof.* By explicit integration

$$R_{\text{Omega}}(p) = h_3(R_{\text{Sharpe}}(p)) \quad (11)$$

$$h_3(y) = 1 + \frac{2y}{\sqrt{2/\pi} e^{-y^2/2} - y + y \operatorname{erf}(y/\sqrt{2})}$$

$h'_3(y) > 0$  for finite real  $y$ . Figure 3 shows a plot of  $h_3$  and  $h'_3$ .

$$\operatorname{erf}(z) \stackrel{\text{def}}{=} \frac{2}{\sqrt{\pi}} \int_0^z e^{-x^2} dx$$

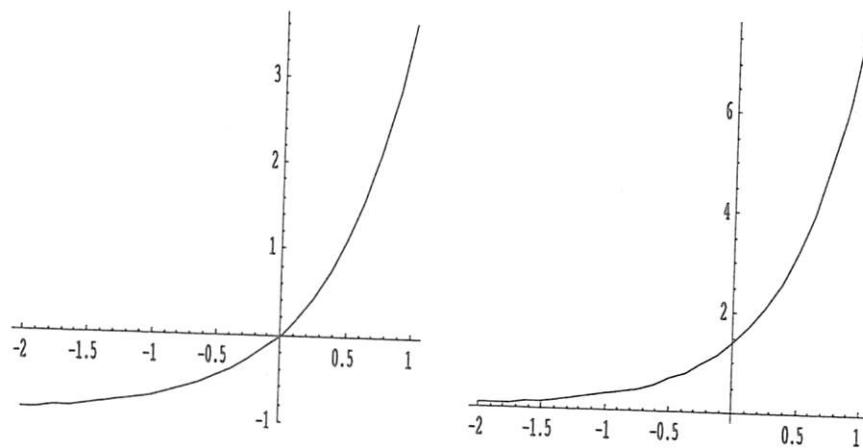


Figure 1 Plot of  $h_1(y)$  (left) and  $h'_1(y)$  (right).  $h_1$  is the Sortino ratio as function of the Sharpe ratio for a portfolio with Gaussian returns

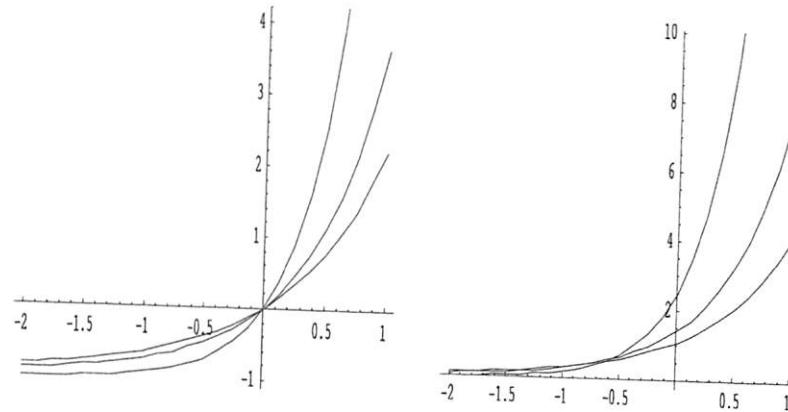


Figure 2 Plot of  $h_2(y)$  (left) and  $h'_2(y)$  (right) for  $n = 1, 2, 3$

**Theorem 4.**  $R_{\text{Stutzer}} \sim R_{\text{Sharpe}}$  if and only if  $y > 0$ .

*Proof.* By explicit integration

$$R_{\text{Stutzer}}(p) = h_4(R_{\text{Sharpe}}(p))$$

$$h_4(y) = \begin{cases} y^2/2 & \text{for } y > 0 \\ 0 & \text{for } y \leq 0 \end{cases} \quad (12)$$

$h'_4(y) > 0$  for real  $y > 0$ . Figure 4 shows a plot of  $h_4$  and  $h'_4$ . For a portfolio with Gaussian mass function, the Stutzer index is unable to rank portfolios with negative  $y$ .

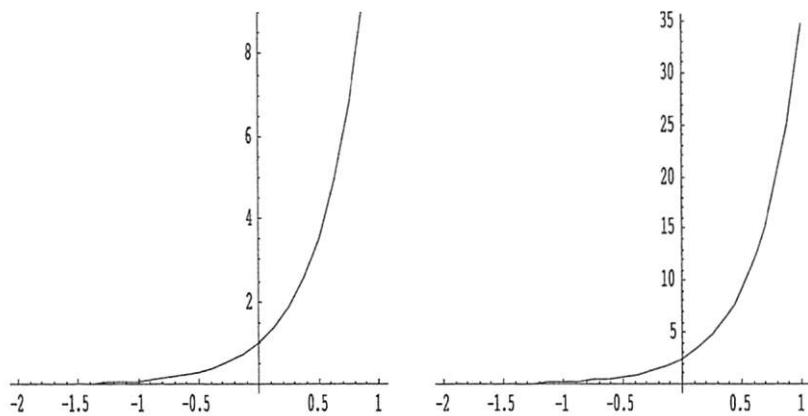


Figure 3 Plot of  $h_3(y)$  (left) and  $h'_3$  (right).  $h_3$  is the omega ratio as function of the Sharpe ratio for a portfolio with Gaussian returns

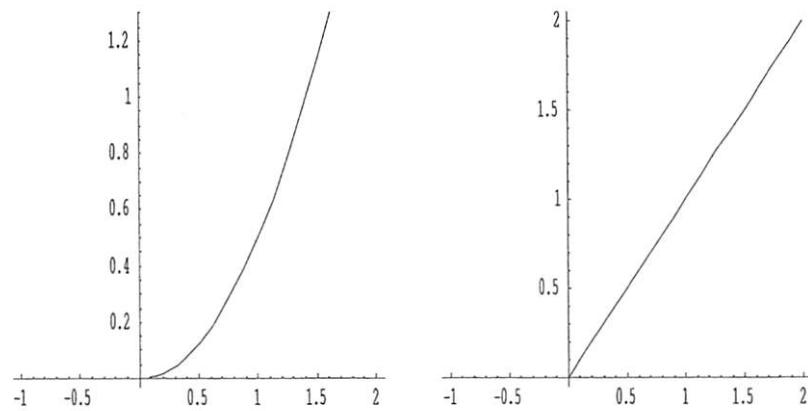


Figure 4 Plot of  $h_4(y)$  (left) and  $h'_4$  (right).  $h'_4$  is the Stutzer index as function of the Sharpe ratio for a portfolio with Gaussian returns

So far, we have shown how in the domain of portfolios with Gaussian returns

$$R_{\text{Sharpe}} \sim R_{\text{Sortino}} \sim R_{\text{Kappa-n}} \sim R_{\text{Omega}} \quad (13)$$

and in the subdomain with portfolios with positive  $R_{\text{Sharpe}}$

$$R_{\text{Sharpe}} \sim R_{\text{Stutzer}} \quad (14)$$

Moreover, we have shown how these indices tend to be more and more sensitive to  $y$  and for larger positive  $y$  and less and less sensitive for more negative  $y$ .

Three questions will be addressed in the following sections:

- Given that all of the above systems are equivalent, are they good measures to rank portfolios?
- If not, what is a better system to rank portfolios?
- How do we extend these results to portfolios with non-Gaussian returns?

#### 9.4 RANKING AND RISK AVERSION

Consider a situation where the risk-free interest rate is  $r = 5\%$  and four possible future scenarios for a portfolio:

- #1: The portfolio out-performs  $r$  with a return  $x = 20\%$
- #2: The portfolio out-performs  $r$  with a return of  $x = 10\%$
- #3: The portfolio under-performs  $r$  with a return of  $x = 3\%$
- #4: The portfolio under-performs  $r$  with a return of  $x = -2\%$ .

It is clear that we prefer #1 to #2, #2 to #3 and #3 to #4 but how do we quantify this preference? How much more do we prefer #1 to #3 when compared to #2? How bad is #4 when compared with #3? The answers to these questions have nothing to do with probability (we are not discussing here the likelihood of one scenario over the other) but have to do with subjective choice and one's perception of risk.

This choice is equivalent to the choice of an implied utility function that we will indicate by  $W(x)$ . It returns one's subjective utility of the scenario in which the portfolio has a fixed return  $x$ . For logical reasons, we value higher returns more than lower returns. So  $W'(x) \geq 0$  and this is the only *a priori* condition we wish to impose.

Given a utility function it is natural to rank a portfolio by evaluating a weighted average of  $W(x)$  over all possible future scenarios. The weight factor is the probability of a future scenario with return  $x$ . This induced ranking can be expressed as

$$R_W(p) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} W(x)p(x) dx \quad (15)$$

Now the question becomes: in a Gaussian world, which choice of  $W(x)$  produces a ranking equivalent to the Sharpe ratio (and all the other rankings equivalent to the Sharpe ratio)?

Surprisingly, the answer is

$$W(x) = W_{\text{naive}}(x) \stackrel{\text{def}}{=} (x - r)/|x - r| \quad (16)$$

which implies

$$\begin{aligned}
 R_{\text{naive}}(x) &= \int_{-\infty}^{+\infty} W_{\text{naive}}(x) p(x) dx \\
 &= \int_{-\infty}^{+\infty} \frac{(x - r)e^{-(x-\mu)^2/2\sigma^2}}{|x - r| \sqrt{2\pi}\sigma} dx \\
 &= \text{erf}((\mu - r)/\sigma\sqrt{2})
 \end{aligned} \tag{17}$$

and

$$R_{\text{naive}}(p) = h_5(R_{\text{Sharpe}}(p)) \tag{18}$$

$$h_5(y) = \text{erf}(y/\sqrt{2})$$

We just proved that for portfolios with Gaussian returns  $h'_5(y) > 0$ .

**Theorem 1.**  $R_{\text{naive}} \sim R_{\text{Sharpe}}$ .

This finding is surprising because it implies that in a Gaussian world, ranking portfolios according to the Sharpe ratio, the Sortino, the Kappa, the Omega or the Stutzer index is equivalent to having the utility function in equation (16). This utility function is plotted in Figure 5.

$W_{\text{naive}}$  is not a risk averse utility function. Referring to the examples of the four scenarios at the beginning of the section this naive utility function implies that we like scenarios #1 and #2 equally (utility  $W = +1$ ) and we dislike scenarios #3 and #4 equally (utility  $W = -1$ ).

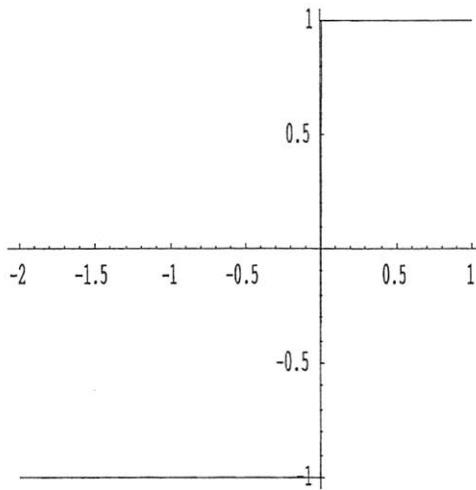


Figure 5 Plot of  $W_{\text{naive}}(x)$ , the utility function (function of the portfolio return) that is equivalent to ranking portfolios using the Sharpe ratio

### 9.5 A BETTER UTILITY FUNCTION

Clearly, the utility function induced by the Sharpe ratio is not natural and does not capture the natural risk aversion of many investors. There is an extensive literature in economics describing more rational choices for utility functions. Because this is a subjective choice we cannot claim that any one utility function is better than all others, but we can select a utility function that is more natural than the one in equation (16) and exhibits the following desirable characteristics:

- $W'(x) > 0$ ; the higher the return of a given scenario the higher the utility of the scenario.
- $W(r) = 0$ ; a scenario in which the return is the same as the risk-free rate has zero utility.
- $\lim_{x \rightarrow \infty} W'(x) = 0$ ; we became insensitive to  $x$  for large positive returns.
- $\lim_{x \rightarrow -\infty} W'(x) = \infty$ ; we are extremely sensitive to  $x$  for large negative returns.

A utility function that exhibits all of the above characteristics is the constant absolute risk aversion (CARA) (Wilmott)

$$W_{\text{CARA}}(x) \stackrel{\text{def}}{=} -e^{-m(x-r)} \quad (19)$$

Here  $k$  is a subjective number that measures one's risk aversion. The larger the  $k$ , the more risk averse one is. The CARA utility function  $W_{\text{CARA}}(x)$ , is shown in Figure 6.

When we substitute equation (19) into equation (15) for a Gaussian  $p(x)$  we obtain

$$R_{\text{CARA}}(p) = \int_{-\infty}^{+\infty} W_{\text{CARA}}(x) p(x) dx \quad (20)$$

$$= \int_{-\infty}^{+\infty} -e^{m(r-x)} \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} dx \quad (21)$$

$$= -e^{m(r-\mu)+m^2\sigma^2/2} \quad (22)$$

In order to have the ranking to be a pure number in the range  $(-\infty, +\infty)$ , we define

$$R_{\text{best}}(p) \stackrel{\text{def}}{=} \mu/r - 1 - m\sigma^2/(2r) \quad (23)$$

which is equivalent to  $R_{\text{CARA}}(p)$  because

$$R_{\text{best}}(p) = h_6(R_{\text{CARA}}(p)) \quad (24)$$

$$h_6(z) = -\log(-z)/(mr) \quad (25)$$

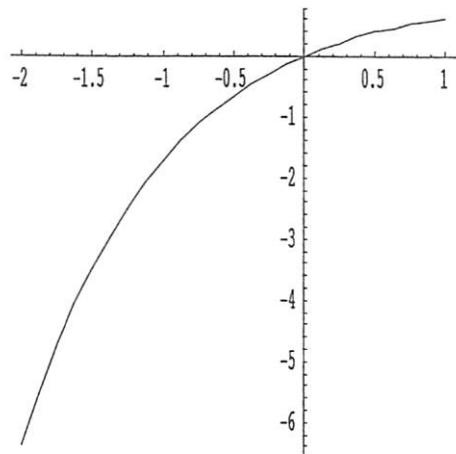


Figure 6 Plot of  $W_{\text{CARA}}(x)$ , the CARA utility function used to derive  $R_{\text{best}}(p) \sim R_{\text{CARA}}(p)$

and  $h'_6(z) > 0$  for  $mr > 0$  and  $z < 0$ .

Note that, for the risk-free asset,  $R_{\text{best}} \approx 0$ . “Good” portfolios rank above 0 and “bad” portfolios rank below 0.

It is important to notice how the Sharpe ratio, as well as the Sortino, the kappa, the omega and the Stutzer index do not depend on any subjective parameters except for minimal acceptable return  $r$  (and  $n$  in the case of kappa), while Equation (23) depends also on the subjective value of  $m$ . This is not surprising since the latter incorporates a scale, represented by  $m$ , that encodes information about how better scenario #1 is when compared with scenario #2. The Sharpe ranking does not incorporate any information about this scale and it assumes that scenario #1 is as good as scenario #2, and scenario #3 is as bad as scenario #4. The introduction of at least one parameter (in our case  $m$ ) is necessary to quantify the cost of taking a risk. The higher the value of  $m$  the higher will be the cost of risk.

The ranking scheme  $R_{\text{best}}$  is not equivalent to  $R_{\text{Sharpe}}$  as shown for the following portfolios (Gaussian returns,  $r = 5\%$  and  $m = 2, 4, 8$ , the latter choices will be explained later).

Note that in all the schemes portfolio E ranks better than A (because it has the same risk but higher return) and I (because it has the same return but lower risk). For the same reasons F ranks better than B and J, G than C and K, and H than D and L as was expected. Nevertheless, the relative ranking of portfolios is different for different schemes and different choices of  $m$ :

- $R_{\text{Sharpe}} \Rightarrow E, A, F, I, G, B, J, H, C, K, D, L$
- $R_{\text{best}} \text{ and } m = 2 \Rightarrow H, L, D, G, K, C, F, J, B, E, I, A$
- $R_{\text{best}} \text{ and } m = 4 \Rightarrow H, G, D, L, K, C, F, J, B, E, I, A$
- $R_{\text{best}} \text{ and } m = 8 \Rightarrow F, E, B, I, J, A, G, C, K, H, D, L$

	$\mu$	$\sigma$	$R_{Sharpe}$	$R_{best}$ ( $m = 2$ )	$R_{best}$ ( $m = 4$ )	$R_{best}$ ( $m = 8$ )
A	16%	9%	1.22	2.04	1.88	1.55
B	21%	14%	1.14	2.81	2.42	1.63
C	28%	21%	1.10	3.72	2.82	1.07
D	32%	25%	1.08	4.15	2.90	0.40
E	17%	9%	1.33	2.24	2.08	1.75
F	22%	14%	1.21	3.01	2.62	1.83
G	29%	21%	1.14	3.92	3.04	1.27
H	33%	25%	1.12	4.35	3.10	0.60
I	17%	10%	1.20	2.20	2.00	1.60
J	22%	15%	1.13	2.95	2.50	1.60
K	29%	22%	1.09	3.83	2.86	0.93
L	33%	26%	1.08	4.25	2.90	0.19

(26)

According to CARA for  $m = 4$  the best portfolio is H and according to Sharpe it is E.

## 9.6 EXTENSION TO NON-GAUSSIAN DISTRIBUTIONS

In the real world the random variables  $x_i$  are not Gaussian and one may wonder how this affects our conclusions.

First of all the statement that the Sharpe ratio, the Sortino, the kappa, the omega and the Stutzer index are equivalent is not true any more and each corresponds to a different implicit choice of a utility function. One cannot answer the question to which one is better because there is no objective benchmark any more.

Anyway the real world is “close” to Gaussian if returns are computed over relatively long time periods (this is shown later) and all of these ranking systems are inappropriate in the Gaussian limit. Therefore, there is no reason to believe they should be appropriate for non-Gaussian or close-to-Gaussian distributions.

On the other hand, despite the fact that in the preceding section, equations (20)–(22) and equation (23) are derived assuming a Gaussian  $p(x)$ , the ranking induced by equation (23) is still correct for non-Gaussian distributions, providing that one plans to hold portfolio a long enough time.

First of all it is important to realize that in the Gaussian world equations (20–22) do not depend on the time scale, since the probability distribution for the 1-, the 2- or the 100-day return is always Gaussian. In a non-Gaussian world the probability distribution for 1-day returns is different than the probability distribution for 2-day returns, etc. In order to take this into account we define  $p(x)$  as the probability mass function for 1-day returns and, in general,  $p_T(X)$  as the probability mass function of

$T$ -day returns where  $X = \log(S_T/S_0) = \sum_{i=0}^{i < T} x_i$  and  $x_i = \log(S_{i+1}/S_i)$ . We will assume all the random variables  $x_i$  are independent and follow the distribution  $p$ . We also define  $\mu$  and  $\sigma$  as the mean and standard deviation of  $p$ .

We can now rewrite equation (15) as the weighted utility at the end of period  $T$

$$R_W(p, T) = \int_{-\infty}^{+\infty} W(X/T) p_T(X) dX \quad (27)$$

Owing to the Central Limit Theorem and the Berry–Esseen Theorem, when  $T \rightarrow \infty$ ,  $p_T$  approaches a Gaussian distribution with mean  $\mu T$  and standard deviation  $\sigma\sqrt{T}$ . This is demonstrated in Figure 7 for an initial triangular distribution. Therefore for our choice  $W = W_{\text{CARA}}$ , equation (27) implies

$$\lim_{T \rightarrow \infty} R_{\text{CARA}}(p, T) = -e^{m(r-\mu)+m^2\sigma^2/2} \sim R_{\text{best}}(p) \quad (28)$$

It remains to address how fast  $R_{\text{CARA}}$  approaches the limit. From the Berry–Esseen theorem it follows that:

**Lemma 1.** For every probability mass function  $p$  and every  $\varepsilon > 0$  there exists a  $c > 0$  such that

$$|R_{\text{CARA}}(p, T) + e^{m(r-\mu)+m^2\sigma^2/2}| < c/\sqrt{T} + \varepsilon \quad (29)$$

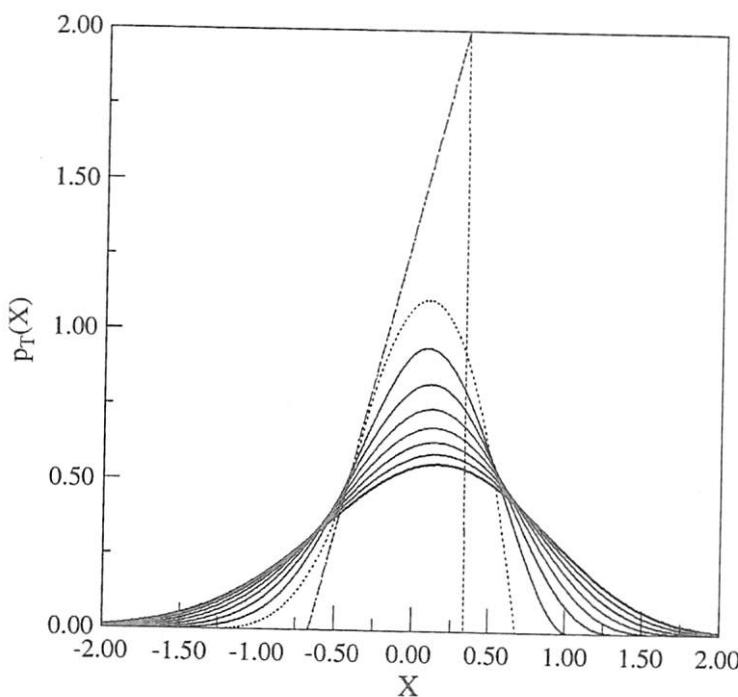


Figure 7  $p_T(x)$  for  $T = 1, \dots, 10$  is shown given a triangular distribution for  $p(x)$  (the dashed line). When  $T \rightarrow \infty$ ,  $p_T(x)$  approaches a Gaussian distribution

where  $c$  depends on  $\varepsilon$  and the shape of  $p$ .

Finally from equation (28) and the above lemma we conclude that:

**Proposition 2.** In the general case of portfolios with random returns having non-Gaussian distributions one should rank the portfolios by explicit integration of equation (27). Nevertheless, for our choice of the CARA utility function and for a long holding period  $T$ , this is equivalent to ranking the portfolios using  $R_{\text{best}}$ , equation (23). The error incurred is proportional to  $1/\sqrt{T}$ .

### 9.7 MARKET DETERMINATION OF $m$

In our notation  $m$  is a positive number that encodes the investor's risk tolerance. A low value of  $m$  (close to zero) indicates a high tolerance of risk, while a high value of  $m$  indicates a low tolerance of risk (risk aversion).

Despite the fact that  $m$  is subjective one can ask if there is something like an "average market value" for  $m$ . In order to address this question we considered the Dow Jones industrial (*DJI*) average index in the range from Oct 1984 until Oct 2004 and we compute the average yearly return  $\mu$  and the average yearly volatility  $\sigma$  (standard deviation) of the weekly lognormal returns. We find

$$\mu = 0.1076 (= 10.76\%) \quad (30)$$

$$\sigma = 0.1620 (= 16.20\%) \quad (31)$$

we then assume that the *DJI* has the same ranking as a risk-free interest ( $= 0$ )

$$R_{\text{best}}(\text{DJI}) = \frac{\mu}{r} - 1 - m \frac{\sigma^2}{2r} = 0 \quad (32)$$

From equation (32) and a reasonable guess  $r \approx 0.05$  (5%) we obtain

$$m \approx 4 \quad (33)$$

Therefore, in this paper we consider empirical values of  $m$  in the range from 2 (for a risk lover investor) to 8 (for a very risk averse investor) and a typical value  $m = 4$  for an average investor.

### 9.8 MODERN PORTFOLIO THEORY

Finally we wish to clarify the role that equation (23) plays in the context of Markowitz' modern portfolio theory (MPT) (Markowitz, 1952; Wilmott, 2000).

In a typical problem one is given a set of  $N$  assets,  $\mu_i$  being the expected return from asset  $i$  and  $\sigma_{ij}$ , the covariance between asset  $i$  and  $j$ . The risk-free rate is  $r$ . What is the optimal portfolio?

The MPT establishes that the optimal portfolio is a combination of the risk-free asset (with weight  $\alpha$ ) and the Markowitz portfolio (with weight  $1 - \alpha$ ). The

Markowitz portfolio is a linear combination of the assets (excluding the risk-free one) with weights equal to

$$w_i \stackrel{\text{def}}{=} \sum_j (\sigma^{-1})_{ij} (\mu_j - r) \quad (34)$$

the mean and variance of the Markowitz portfolio are given by

$$\mu_M \stackrel{\text{def}}{=} \sum_j w_i \mu_i \text{ and } \sigma_M \stackrel{\text{def}}{=} \sqrt{\sum_j w_i w_j \sigma_{ij}} \quad (35)$$

The MPT decouples the problem of finding the optimal combination of risky assets with that of finding the optimal combination of risk-free asset and risky assets. The MPT solves the first problem but the solution of the second problem is not uniquely determined because it leaves  $\alpha$  undetermined.

Since  $\alpha$  is not determined by the Markowitz's method, its value is subjective and, in its own way,  $\alpha$  measures the attitude towards risk of the investor. We see  $\alpha$  is related to our value of  $m$ .

Let  $M_\alpha$  be a portfolio, which is a linear combination of the risk-free asset with weight  $\alpha$  and the Markowitz portfolio (as computed by the MPT) with weight  $(1 - \alpha)$ . The probability mass function associated to this portfolio is

$$p_\alpha(x) = \alpha \delta(x - r) + (1 - \alpha) \frac{1}{\sqrt{2\pi}\sigma_M} e^{-(x-\mu_M)^2/2\sigma_M^2} \quad (36)$$

where  $\delta$  is the Dirac delta function. We can now determine  $\alpha$  by maximizing  $R_{\text{best}}(p_\alpha)$ . This procedure is represented graphically in Figure 8. The solution can be found by explicit computation as stated in the following:

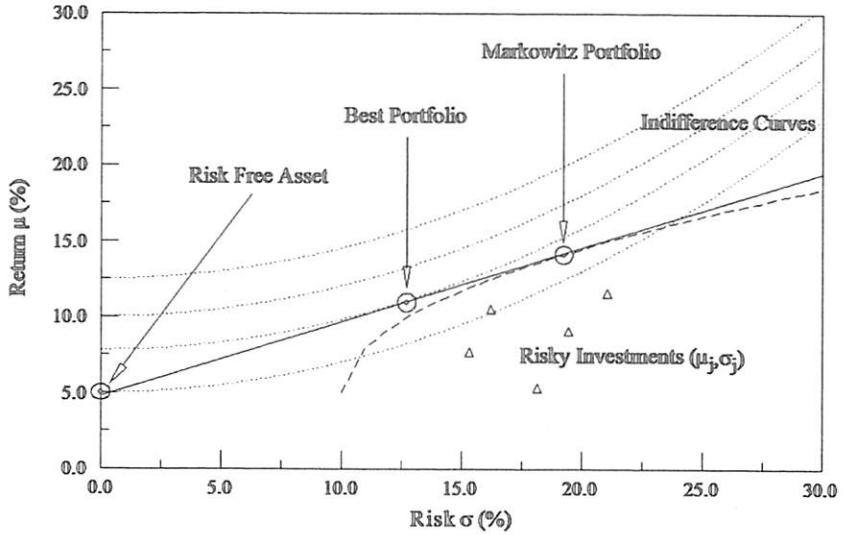
**Theorem 1.** Given a risk-free rate  $r$  and a Markowitz portfolio  $(\mu_M, \sigma_M)$ , according to the  $R_{\text{best}}$  ranking, the optimal portfolio consists of holding a fraction  $\alpha$  of the risk-free asset where

$$\alpha = 1 - \frac{\mu_M - r}{m\sigma_M^2} \quad (37)$$

and a fraction  $(1 - \alpha)$  of the Markowitz portfolio. This optimal portfolio has an expected average return and volatility given by

$$\mu_{\text{best}} = r + \frac{(\mu_M - r)^2}{m\sigma_M^2} \text{ and } \sigma_{\text{best}} = \frac{\mu_M - r}{m\sigma_M} \quad (38)$$

Note that the Sharpe ratio cannot be used to solve the problem of determining  $\alpha$  for two reasons: (1) the Sharpe ratio is undetermined for the risk-free asset; and (2) in the return versus risk plot the indifference curves associated with constant rankings are straight lines, therefore they cannot be tangent to the Markowitz line parameterized by  $\alpha$ .



**Figure 8** The MPT on a  $\mu$ ,  $\sigma$  plane is shown, where the ranking  $R_{\text{best}}(p)$  is used to determine the investor's indifference curves. All portfolios on the line are equivalent according to MPT. The concave parabolas represent indifference curves (sets of portfolios having the same ranking). The higher the parabola, the higher the ranking. The best portfolio can be determined by finding the indifference curve tangent to the line or indifference curves. All portfolios on the line are equivalent according to MPT. The concave parabolas represent indifference curves (sets of portfolios having the same ranking). The higher the parabola, the higher the ranking. The best portfolio can be determined by finding the indifference curve tangent to the line figure

### 9.9 CONCLUSIONS

In this paper we proved that using the Sharpe ratio, the Sortino, the kappa, the omega or the Stutzer index to rank portfolios are equivalent choices when portfolios have Gaussian returns (that eventually is true for sufficiently long time  $T$ ). Nevertheless, all of these ranking systems implicitly assume a utility function that is not consistent with the risk aversion of investors. Moreover these ranking schemes are not useful for many important practical applications (such as finding indifference curves for use in MPT).

With our choice of the CARA utility function  $W(x) = -e^{-m(x-r)}$  (where  $r$  is the risk-free rate and  $m$  parametrizes our risk attitude) we determined a ranking formula

$$R_{\text{best}}(p) = \mu/r - 1 - m\sigma^2/(2r) \quad (39)$$

that is applicable to portfolios with Gaussian and non-Gaussian returns. In the latter case the formula is valid but introduces a numerical error in the ranking that is proportional to  $T^{-1/2}$ .  $T$  is the time one is planning to hold the portfolio. This error can be arbitrarily reduced by holding the portfolio for a sufficiently long time.

Finally, we performed an analysis of the *DJI* average index (data from 1984 until today) and determined an average market value for  $m = 4$ . Further studies

on the time dependence of  $m$  and its correlation with other market indicators are required.

According to the EDHEC (Amenc, Malaise, Martellini and Vaisse) 69% of European Hedge Funds reports the Sharpe ratio to their investors and 22% report the Sortino ratio. We believe these numbers are misleading and should be used with caution. In particular:

- One should not at all rank portfolios that are correlated, in this case one should use the MPT or the CAPM models.
- If one is considering a set of uncorrelated portfolios (or portfolios with unknown correlations) in order to select the best and invest funds in both the selected portfolio and the risk-free asset, one should use the Sharpe ratio to rank the portfolios. In this case one also needs to select a utility function to decide how to partition the funds between the selected portfolio and the risk-free asset.
- If one is considering a set of uncorrelated portfolios (or portfolios with unknown correlations) in order to invest a fixed amount of money in the selected portfolio, one should use equation (39).

Some of the ideas discussed in this paper are implemented in the form of computer programs and can be accessed through the web page:

<http://www.metacryption.com/schemes.html>

## REFERENCES

- Amenc, N., Malaise, P., Martellini, L. and Vaisse, M., Fund of Hedge Fund Reporting, EDHEC, <http://www.edhec.com>
- Kaplan, P.D. and Knowles, J.A. (2004) Kappa: a generalized downside risk-adjusted performance measure, *Journal of Performance Measurement*, 8(3).
- Kazemi, H., Schneeweis, T. and Gupta, R. (2003) Omega as performance measure, *CISDM 2003, Proceedings*.
- Markowitz, H.M. (1952) Portfolio selection, *Journal of Finance*, 7(1), 77–91.
- Shadwick, W. and Keating, C. (2002) A universal performance measure, *Journal of Performance Measurement*, (Spring) 6(3).
- Sharpe, W.F. (1964) Capital asset prices: a theory of market equilibrium under conditions of risk, *Journal of Finance*, 19(3), 425–442.
- Sortino, F.A. (2001) From alpha to omega, *Managing Downside Risk in Financial Markets*, F.A. Sortino and S.E. Satchell (eds), Reed Educational and Professional Publishing Ltd.
- Sortino, F.A. and Forsey, H.J. (1996) On the use and misuse of downside risk, *Journal of Portfolio Management*, 22(2), 35–42.
- Sortino, F.A. and Price, L.N. (1994) Performance measurement in a downside risk framework, *Journal of Investing*, 3(3), 59–64.
- Sortino, F.A. and Van Der Meer, R. (1991) Downside risk, *Journal of Portfolio Management*, 17(4), 27–32.
- Stutzer, M. (2000) A portfolio performance index, *Financial Analysts Journal*, 56 May/June.
- Wilmott, P. (2000) *On Quantitative Finance*, Wiley, New York.

# Onium Masses with Three Flavors of Dynamical Quarks

---

**Steven Gottlieb\*, L. Levkova**

*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

*E-mail:* sg@indiana.edu, llevkova@indiana.edu

**M. Di Pierro**

*School of Computer Science, Telecommunications and Information Systems, DePaul University,  
Chicago, Illinois 60604, USA*

*E-mail:* MDiPierro@cs.depaul.edu

**A. El-Khadra, D. Menscher**

*Physics Department, University of Illinois, Urbana, Illinois 61801, USA*

*E-mail:* axk@uiuc.edu, menscher@uiuc.edu

**A.S. Kronfeld, P.B. Mackenzie, J. Simone**

*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

*E-mail:* ask@fnal.gov, mackenzie@fnal.gov, simone@fnal.gov

We have greatly extended an earlier calculation of the charmonium spectrum on three flavor dynamical quark ensembles by using more recent ensembles generated by the MILC collaboration. The heavy quarks are treated using the Fermilab formulation. The charmonium state masses are in reasonable agreement with the observed spectrum; however, some of the spin splittings may still be too small.

*XXIIIrd International Symposium on Lattice Field Theory*

*25-30 July 2005*

*Trinity College, Dublin, Ireland*

---

\*Speaker.

## 1. Introduction

Calculating the spectrum of onium states is a significant challenge for lattice gauge theory. A number of levels can be studied for both charm and bottom quarks. However, dealing with heavy quarks requires special care [1, 2]. Using improved staggered sea quarks [3], it is possible to reproduce many of the most important features of the spectrum [4], which had not been done in the quenched approximation. This paper updates our work presented at Lattice 2003 [5].

## 2. Calculational Details

Ensembles for three lattice spacings were provided by the MILC Collaboration [6]:  $a \approx 0.18$  fm (“extra-coarse”),  $a \approx 0.12$  fm (“coarse”), and  $a \approx 0.086$  fm (“fine”). (See Table 1.) For the extra coarse  $am_q / am_s = 0.6, 0.4, 0.2$  and  $0.1$ ; for the coarse lattice, we also have  $0.14$ , but we have only analyzed two values  $0.4$  and  $0.2$  for the fine lattice. From 400 to 600 configurations have been analyzed in most ensembles. The most notable exception is the coarse ensemble with  $am_q / am_s = 0.1$ . For each of the lattice spacings, the scale of each ensemble with different sea quark masses was kept approximately fixed using the length  $r_1$  [7, 8] from the static quark potential. The absolute scale from the  $\Upsilon$  2S–1S splitting was determined by the HPQCD/UKQCD group [9, 4] on most of our ensembles implying  $r_1 = 0.318(7)$  fm.

$am_q / am_s$	$10/g^2$	size	volume	config.	$a$ (fm)
0.0492 / 0.082	6.503	$16^3 \times 48$	$(2.8 \text{ fm})^3$	401	0.178
0.0328 / 0.082	6.485	$16^3 \times 48$	$(2.8 \text{ fm})^3$	331	0.177
0.0164 / 0.082	6.467	$16^3 \times 48$	$(2.8 \text{ fm})^3$	645	0.176
0.0082 / 0.082	6.458	$16^3 \times 48$	$(2.8 \text{ fm})^3$	400	0.176
0.03 / 0.05	6.81	$20^3 \times 64$	$(2.4 \text{ fm})^3$	559	0.120
0.02 / 0.05	6.79	$20^3 \times 64$	$(2.4 \text{ fm})^3$	460	0.120
0.01 / 0.05	6.76	$20^3 \times 64$	$(2.4 \text{ fm})^3$	593	0.121
0.007 / 0.05	6.76	$20^3 \times 64$	$(2.4 \text{ fm})^3$	403	0.121
0.005 / 0.05	6.76	$24^3 \times 64$	$(2.9 \text{ fm})^3$	136	0.120
0.0124 / 0.031	7.11	$28^3 \times 96$	$(2.4 \text{ fm})^3$	261	0.0863
0.0062 / 0.031	7.09	$28^3 \times 96$	$(2.4 \text{ fm})^3$	472	0.0861

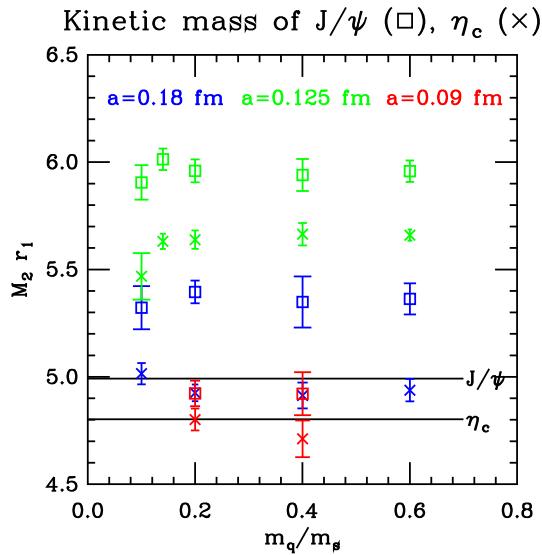
**Table 1:** Ensembles used in this calculation.

We use the Asqtad improved staggered sea quark action that has errors of  $O(\alpha_s a^2)$ . The improved gluon action has errors of  $O(\alpha_s^2 a^2)$ . For the heavy valence quarks, we use the Sheikholeslami-Wohlert action [10] (which has  $O(\alpha_s a)$  errors) with the Fermilab interpretation [2]. To compute heavy quark propagators, we use point and smeared sources and sinks. The smearing approximates 1S or 2S wavefunctions. At the sink, spatial momentum  $2\pi/(La)[p_x, p_y, p_z]$  is given to the onium state. We restrict the range of  $p$  such that  $\sum p_i^2 \leq 9$ .

To find the onium masses, we fit two channels simultaneously for the zero momentum states. A delta function and a 1S smearing wave function are used as the source and sink. The ground state and up to three excited states are included in the fit. The minimum and maximum distance from

the source are varied, and the best fit is selected based on the confidence level and size of error in the ground state and first excited state masses. After choosing the fit range, 250 bootstrap samples are generated to provide an error estimate.

We must tune the hopping parameter  $\kappa$  to the charm or bottom mass. For each lattice spacing, we select a sea quark mass independent value for  $\kappa$ . The tuning is done on an ensemble with small sea quark mass. In fact, as this project was done in conjunction with a study of heavy-light mesons, the tuning was done for the  $D_s$  mass. The precision of that tuning was only about 8%. Because of lattice artifacts that arise for heavy states, we distinguish between the rest mass  $aM_1$  and the kinetic mass  $aM_2$ . We use  $\kappa = 0.120, 0.119$  and  $0.127$  on the extra coarse, coarse and fine ensembles, respectively. The imprecision of our tuning is immediately seen in Fig. 1.



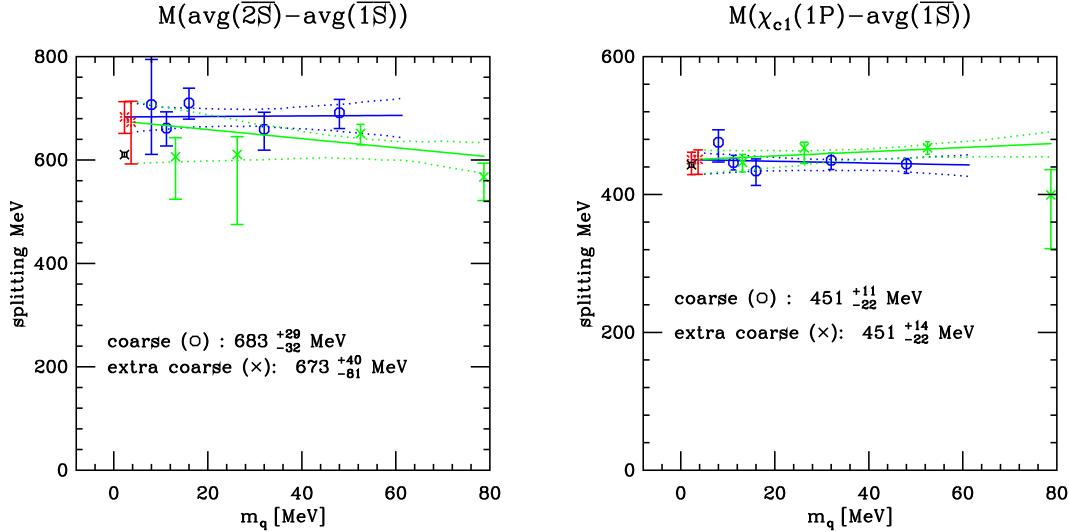
**Figure 1:** The kinetic masses of  $J/\psi$  and  $\eta_c$  on each ensemble plotted as a function  $m_q/m_s$  the light sea to strange quark mass ratio. Masses are in units of  $r_1$ . The physical masses are shown as lines.

The kinetic masses have two disadvantages: their statistical errors are large compared to those of the rest masses, and the pattern of systematic errors is more subtle [11]. However, for level splittings, a large discretization effect in the quark's rest mass drops out of the energy differences of hadron rest masses [12]. So, having tuned to approximately the right charm mass, we will now consider splittings based upon the rest masses of the various states. These states have been studied:  $\eta_c(1S)$ ,  $\eta_c(2S)$ ,  $\psi(1S)$ ,  $\psi(2S)$ ,  $h_c(1P)$ ,  $\chi_{c0}(1P)$  and  $\chi_{c1}(1P)$ . The  $\chi_{c2}(1P)$  is also under study with a nonrelativistic  $P$ -wave source [13]. Currently, results for  $\chi_{c2}(1P)$  are only available on one extra coarse ensemble. We also use the spin-averaged mass, *e.g.*,  $\overline{1S} = [3M_{\psi(1S)} + M_{\eta_c(1S)}]/4$  to display some of the splittings in the spectrum.

### 3. Results

For each lattice spacing, we plot the splittings as a function of the mass of the light sea quarks. A linear chiral fit is made and the splitting is extrapolated to the physical value of  $\hat{m} = (m_u + m_d)/2$

where the lattice spacing dependent value of  $\hat{m}$  is determined from analysis of  $\pi$  and  $K$  meson decays constants [14]. The light meson decay constant analysis has not yet been completed on the extra coarse ensembles, so the value of  $\hat{m}$  used there is only a rough estimate.



**Figure 2:** (left) The chiral extrapolation of the spin-averaged splitting between the 2S and 1S states on the extra coarse and coarse ensembles. The extrapolated values are shown in red, and the physical value in black.

**Figure 3:** (right) The splitting between the  $\chi_{c1}$  ( ${}^3P_1$ ) and spin-averaged 1S states on the extra coarse and coarse ensembles.

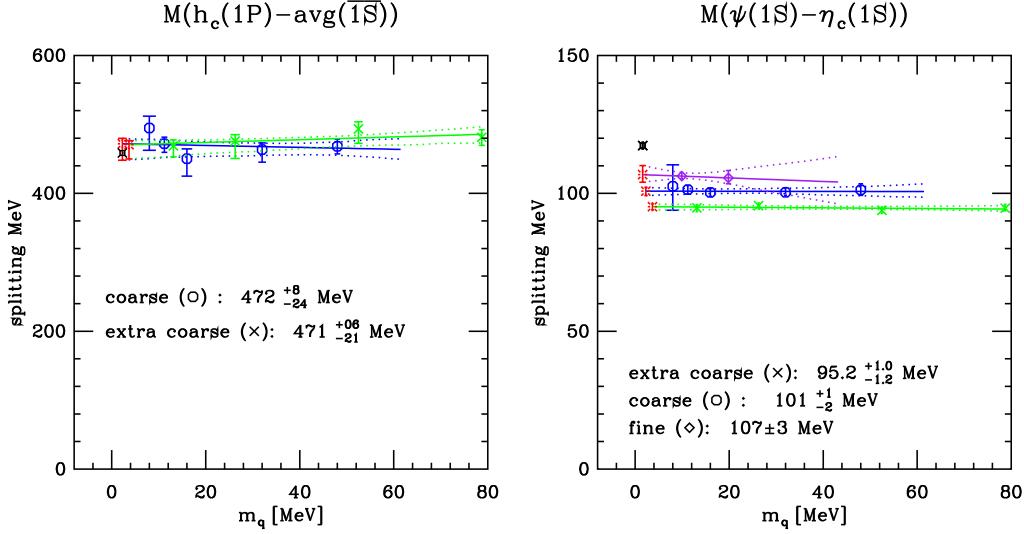
Within our current statistical uncertainties, we see reasonable agreement with the experimental value of the splittings of the spin-averaged 2S and 1S levels. The coarse value is about  $2\sigma$  high. (See Fig. 2.) As we do not yet have a full set of results for the  $\chi_{c2}$ , we cannot construct the spin average of the 1P states. Instead, we use the  $\chi_{c1}$  and the  $h_c$ . In nonrelativistic potential models, these two states are degenerate with each other and the spin average. The experimental splittings are well reproduced for these states. (See Figs. 3 and 4.)

As seen in Fig. 5, the spin splittings are too small. For  $J/\psi$  and  $\eta_c$  it amounts to about 10–22 MeV. The splitting is 19% too small for the extra coarse ensemble, 14% too small for the coarse, and 9% too small for the fine. The splitting seems to systematically improve as the lattice spacing decreases. We have not yet attempted a continuum extrapolation.

The overall agreement between this calculation with dynamical quarks and the observed spectrum is very good. The most obvious issue is the smallness of spin splittings, as seen in the  $J/\psi - \eta_c$  splitting, and the mass of the  $\chi_{c0}$  state. There is some evidence of improvement as the lattice spacing is reduced.

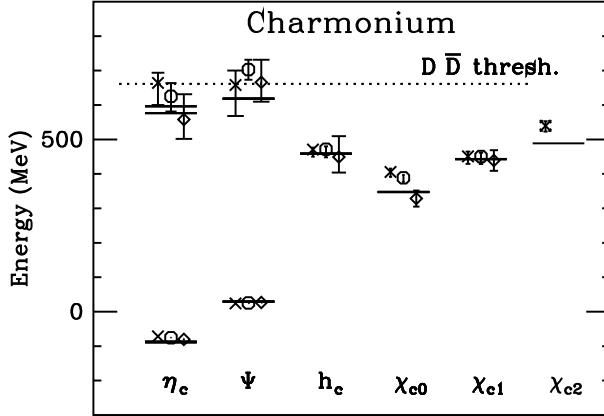
#### 4. Outlook

There are several ways to improve this calculation in the near future: We can include another fine ensemble with  $m_q = 0.1m_s$ . This more chiral ensemble is still being generated, but is far enough along that it would be worth starting the analysis. We also need to examine alternative fits to the ones that were selected by our automated procedure. Fermilab/MILC are almost finished



**Figure 4:** (left) Same as Fig. 3, except for the  $h_c(^1P_1)$ .

**Figure 5:** (right) Splitting between  $J/\psi$  and  $\eta_c$  for all three lattice spacings. There are only two ensembles for the fine lattice spacing, shown in purple.



**Figure 6:** Charmonium spectrum for all three lattice spacings compared with experimental values. Energy is offset so that zero represents the spin-averaged 1S energy. From left to right for each state, crosses, octagons and diamonds are from the extra coarse, coarse and fine ensembles, respectively. The extra coarse  $\chi_{c2}$  value without chiral extrapolation is the fancy cross.

generating a new set of ensembles at a lattice spacing between extra coarse and coarse. Production running on additional ensembles for the new  $P$ -wave code will be done. We also plan to use heavier quarks to study bottomonium, which has already been studied on these configurations using NRQCD [9].

In the longer term, MILC is generating new ensembles with  $a \approx 0.06$  fm that should help us better understand the continuum limit. However, in the current calculation, lattice spacing dependence does not seem very large compared with statistical errors for most states.

We gratefully acknowledge the support of the Department of Energy. In addition, the Fermilab

Computing Division, Indiana University (AVIDD cluster) and National Center for Supercomputing Applications provided computing support.

## References

- [1] G.P. Lepage *et al.*, *Improved nonrelativistic QCD for heavy quark physics* Phys. Rev. D **46**, (1992) 4052 [hep-lat/9205007].
- [2] A. El-Khadra, A.S. Kronfeld, P.B. Mackenzie, *Massive fermions in lattice gauge theory*, Phys. Rev. D **55**, (1997) 3933 [hep-lat/9604004].
- [3] T. Blum *et al.* [MILC], *Improving flavor symmetry in the Kogut-Susskind hadron spectrum*, Phys. Rev. D **55**, (1997) 1133 [hep-lat/9609036]; K. Orginos and D. Toussaint, *Testing improved actions for dynamical Kogut-Susskind quarks*, Phys. Rev. D **59** (1999) 014501 [hep-lat/9805009]; K. Orginos, D. Toussaint and R.L. Sugar, *Variants of fattening and flavor symmetry restoration*, Phys. Rev. D **60** 054503 (1999) [hep-lat/9903032]; G.P. Lepage, *Flavor-symmetry restoration and Symanzik improvement for staggered quarks*, Phys. Rev. D **59** (1999) 074502 [hep-lat/9809157]; J.F. Lagae and D.K. Sinclair, *Improved staggered quark actions with reduced flavour symmetry violations for lattice QCD*, Phys. Rev. D **59** (1999) 014511 [hep-lat/9806014]; C. Bernard *et al.* [MILC], *Quenched hadron spectroscopy with improved staggered quark action*, Phys. Rev. D **58** (1998) 014503 [hep-lat/9712010].
- [4] C. Davies *et al.* [Fermilab Lattice, HPQCD, MILC, UKQCD], *High-precision lattice QCD confronts experiment*, Phys. Rev. Lett. **92** 022001 (2004). [hep-lat/0304004].
- [5] M. di Pierro *et al.*, *Properties of charmonium in lattice QCD with 2+1 flavors of improved staggered sea quarks*, Nucl. Phys. B (Proc. Suppl.) **129**, (2004) 340 [hep-lat/0310042].
- [6] C. Bernard *et al.* [MILC], *The QCD spectrum with three quark flavors*, Phys. Rev. D **64** (2001) 054506 [hep-lat/0104002]; C. Aubin *et al.* [MILC], *Light hadrons with improved staggered quarks: Approaching the continuum limit*, Phys. Rev. D **70**, 094505 (2004) [hep-lat/0402030].
- [7] R. Sommer, *A new way to set the energy scale in lattice gauge theories ...*, Nucl. Phys. **B411**, 839 (1994) [hep-lat/9310022].
- [8] C. Bernard *et al.* [MILC], *The static quark potential in three flavor QCD*, Phys. Rev. D **62**, (2000) 034503 [hep-lat/0002028].
- [9] A. Gray *et al.* [HPQCD, UKQCD], *The  $\Upsilon$  spectrum and  $m_b$  from full lattice QCD*, [hep-lat/0507013].
- [10] B. Sheikholeslami and R. Wohlert, *Improved continuum limit lattice action for QCD with Wilson fermions*, Nucl. Phys. **B259**, 572 (1985).
- [11] A.S. Kronfeld, *Binding energies in nonrelativistic field theories*, Nucl. Phys. Proc. Suppl. **53** (1997) 401 [hep-lat/9608139].
- [12] A.S. Kronfeld, *Application of heavy-quark effective theory to lattice QCD. I: Power corrections*, Phys. Rev. D **62**, 014505 (2000) [hep-lat/0002008].
- [13] C.T.H. Davies *et al.*, *Precision  $\Upsilon$  spectroscopy from nonrelativistic lattice QCD*, Phys. Rev. D **50**, (1994) 6963 [hep-lat/9406017].
- [14] C. Aubin *et al.* [HPQCD, MILC, and UKQCD], *First determination of the strange and light quark masses from full lattice QCD*, Phys. Rev. D **70**, 031504(R) (2004) [hep-lat/0405022].

# Predictions from Lattice QCD

FERMILAB-CONF-05-428-T

A.S. Kronfeld<sup>\*a</sup>, I.F. Allison<sup>b</sup>, C. Aubin<sup>c,d</sup>, C. Bernard<sup>d</sup>, C.T.H. Davies<sup>b</sup>, C. DeTar<sup>e</sup>,  
M. Di Pierro<sup>f</sup>, E.D. Freeland<sup>g</sup>, Steven Gottlieb<sup>h</sup>, A. Gray<sup>i</sup>, E. Gregory<sup>j</sup>, U.M. Heller<sup>k</sup>,  
J.E. Hetrick<sup>l</sup>, A.X. El-Khadra<sup>m</sup>, L. Levkova<sup>h</sup>, P.B. Mackenzie<sup>a</sup>, F. Maresca<sup>e</sup>,  
D. Menscher<sup>m</sup>, M. Nobes<sup>n,o</sup>, M. Okamoto<sup>a</sup>, M.B. Oktay<sup>m</sup>, J. Osborn<sup>e</sup>, D. Renner<sup>j</sup>,  
J.N. Simone<sup>a</sup>, R. Sugar<sup>p</sup>, D. Toussaint<sup>j</sup>, H.D. Trottier<sup>o</sup>

E-mail: ask@fnal.gov

<sup>a</sup>Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

<sup>b</sup>Department of Physics and Astronomy, Glasgow University, Glasgow, Scotland G12 8QQ, UK

<sup>c</sup>Physics Department, Columbia University, New York, New York 10027, USA

<sup>d</sup>Department of Physics, Washington University, St. Louis, Missouri 63130, USA

<sup>e</sup>Physics Department, University of Utah, Salt Lake City, Utah 84112, USA

<sup>f</sup>School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois 60604, USA

<sup>g</sup>Liberal Arts Department, School of the Art Institute, Chicago, Illinois 60603, USA

<sup>h</sup>Department of Physics, Indiana University, Bloomington, Indiana 47405, USA

<sup>i</sup>Department of Physics, The Ohio State University, Columbus, Ohio 43210, USA

<sup>j</sup>Department of Physics, University of Arizona, Tucson, Arizona 85721, USA

<sup>k</sup>American Physical Society, Ridge, New York 11961, USA

<sup>l</sup>Physics Department, University of the Pacific, Stockton, California 95211, USA

<sup>m</sup>Physics Department, University of Illinois, Urbana, Illinois 61801, USA

<sup>n</sup>Laboratory of Elementary-Particle Physics, Cornell University, Ithaca, New York 14853, USA

<sup>o</sup>Physics Department, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada

<sup>p</sup>Department of Physics, University of California, Santa Barbara, California 93106, USA

## Fermilab Lattice, MILC, and HPQCD Collaborations

In the past year, we calculated with lattice QCD three quantities that were unknown or poorly known. They are the  $q^2$  dependence of the form factor in semileptonic  $D \rightarrow K l \nu$  decay, the decay constant of the  $D$  meson, and the mass of the  $B_c$  meson. In this talk, we summarize these calculations, with emphasis on their (subsequent) confirmation by experiments.

XXIIIrd International Symposium on Lattice Field Theory

25-30 July 2005

Trinity College, Dublin, Ireland

<sup>\*</sup>Speaker.

## 1. Introduction and Background

In recent years, lattice QCD has reached the stage where many calculations of hadron masses, mass splittings, and operator matrix elements agree with experimental measurements. The key has been the inclusion of sea quarks. The progress has been especially striking [1] when the light quarks (sea and valence) are implemented as staggered quarks, with an improved action.

Some of the ingredients of these calculations are controversial. Staggered quarks come in four tastes, three of which must be removed to obtain each individual flavor. For sea quarks, this is done by taking the fourth root of the fermion determinant; for valence quarks, by projecting onto the desired taste sector. Furthermore chiral perturbation theory must be modified [2]. Although evidence for the validity of these “tricks” is slowly accumulating, a proof remains at large [3]. In addition, much of the success of Ref. [1] comes from hadrons with heavy quarks. Although debate on heavy quarks in lattice QCD seems to have subsided, checks are still useful.

In this paper, we discuss three calculations, with emphasis on their subsequent experimental confirmation. They are the normalization and  $q^2$ -dependence of the  $D \rightarrow K\ell\nu$  form factor; the decay constants of the  $D^+$  and  $D_s$  mesons; and the mass of the  $B_c$  meson. Each tests a somewhat different combination of the ingredients, and the following table gives an informal guide:

calculation	light sea	light valence	heavy
semileptonic $f_+(q^2)$	**	**	**
leptonic $f_D$	**	***	**
$B_c$ mass	**	—	***

The chiral extrapolation, which is more sensitive to valence quarks than sea quarks, turned out to be more important for the decay constant than the form factor. The  $B_c$  meson has no light valence quarks at all, but one should expect an accurate calculation only if heavy-quark discretization effects are under control.

Successful predictions are, of course, not a substitute for a proof. They are still useful. Even if the experts are confident of all the elements of their numerical calculations, non-experts are interested in an end-to-end check [4]. The quantities discussed here are ideal candidates: they are straightforward to compute; the first “good” experimental measurements were not expected until this year; and new physics is unlikely to contribute significantly.

## 2. Semileptonic $D$ Decays

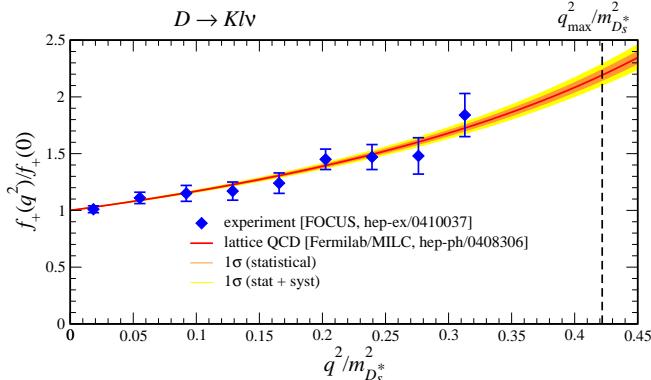
Semileptonic decays such as  $D \rightarrow K\ell\nu$  are mediated by electroweak vector currents. The matrix element  $\langle K|V^\mu|D\rangle$  is parametrized by form factors. For a vector current there are two, but experimentally only the one called  $f_+(q^2)$  is accessible; the rate from the other one,  $f_0(q^2)$ , is suppressed by  $m_l^2$ . Here  $q^2$  is the momentum transferred to the lepton-neutrino system, falling in the range  $0 \leq q^2 \leq q_{\max}^2 = (m_D - m_K)^2$ . In lattice QCD, discretization effects are smallest when the momentum  $\mathbf{p}$  of the kaon is small, and then  $q^2$  is not too far from  $q_{\max}^2$ .

Experiments usually measure the branching fraction and quote the normalization  $f_+(0)$ , after making assumptions about the  $q^2$  dependence. While our results were still preliminary [5], experimental results came out for the normalization of  $D \rightarrow K\ell\nu$  [6] and  $D \rightarrow \pi\ell\nu$  [7]. The agreement

with our final results [8] is excellent. For example, we find  $f_+^{D \rightarrow K}(0) = 0.73(3)(7)$  [8] while BES measures  $f_+^{D \rightarrow K}(0) = 0.78(5)$  [6]. Our calculations of the normalization are also consistent with the soft pion theorem, which states  $f_0(q_{\max}^2) = f_D/f_\pi$ .

In principle, the shape of the form factors can be computed directly in lattice QCD. In practice, we calculated at a few values of  $\mathbf{p}$  and used the Bećirević-Kaidalov (BK) form [9] to fix the full  $q^2$  dependence of  $f_+$  and  $f_0$ . Then the normalization of  $f_+$  comes mainly from  $f_0$  through a kinematic constraint  $f_+(0) = f_0(0)$ . The BK Ansatz and calculations near  $q_{\max}^2$  determine the shape. It was important, therefore, to measure the  $q^2$  dependence experimentally. In photoproduction of charm off fixed nuclear targets, the FOCUS Collaboration was able to collect high enough statistics to trace out the  $q^2$  distribution of the decay [10]. This setup does not yield an absolutely normalized branching ratio, so one is left to compare  $f_+(q^2)/f_+(0)$ .

In Fig. 1 we plot our result for  $f_+(q^2)/f_+(0)$  vs.  $q^2/m_{D_s^*}^2$ . The errors from  $f_+(0)$  must be propagated to non-zero  $q^2$ , so for  $f_+(q^2)/f_+(0)$  the errors grow with  $q^2$ . Figure 1 shows 1- $\sigma$  bands of statistical (orange) and all uncertainties (yellow) added in quadrature [11]. As one can see, the  $q^2$  dependence of lattice QCD (curve and error band) and experiment (points) agree excellently, although the uncertainties are still several per cent.



**Figure 1:** Shape of form factor  $f_+(q^2)/f_+(0)$  vs.  $q^2/m_{D_s^*}^2$ , compared with experiment [10].

POS(LAT2005)206

### 3. Leptonic $D$ Decays

We also computed the hadronic matrix element for the leptonic decay of charmed mesons,  $f_{D^+}$  and  $f_{D_s}$ . The first (experimental) measurements of  $f_{D^+}$  appeared in 2004, with three events from BES [12] and eight from CLEO [13]. Neither provides a stringent test of QCD, but CLEO- $c$  was just starting its run and promised 5–8 times higher statistics by the Summer 2005 Lepton-Photon Symposium [4]. At Lattice 2004 [14], we presented preliminary results for  $f_{D^+}$ , based on one lattice spacing,  $a \approx 0.125$  fm. Our aim was to extend the running to two other lattice spacings and, of course, to improve our understanding of other aspects of the calculation, such as the chiral extrapolation. Details are given in the ensuing publication [15]. We find

$$f_{D^+} = 201 \pm 3 \pm 17 \text{ MeV}, \quad (3.1)$$

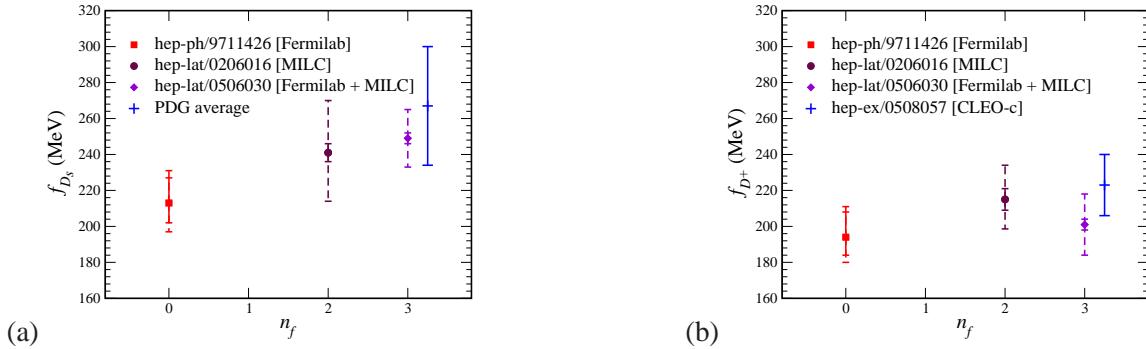
where the first error is from finite Monte Carlo statistics, the second is a sum in quadrature of several systematics. A conservative (but not naïve) estimate of heavy-quark discretizations effects,

as discussed in Ref. [16], is the second largest (largest) systematic on  $f_{D^+}$  ( $f_{D_s}$ ). A few days after our paper was posted on the arXiv, CLEO-*c* announced its new measurement [17]

$$f_{D^+} = 223 \pm 17 \pm 3 \text{ MeV}, \quad (3.2)$$

based on  $47 \pm 8$  events. At the  $1-\sigma$  level, the agreement between Eqs. (3.1) and (3.2) is fine. One should keep in mind that the experiment actually determines  $|V_{cd}| f_{D^+}$ . CLEO-*c* [17] assumes that  $|V_{cd}| = |V_{us}|$  and uses a recent average of  $|V_{us}|$  from semileptonic  $K$  decay.

It is interesting to look at the  $n_f$  dependence of  $f_{D_s}$ , shown in Fig. 2(a). Of course, quenched results vary widely, but we show one [18] carried out with similar choices for heavy quarks, renormalization factors, etc. One sees a trend of  $f_{D_s}$  to increase with  $n_f$ . A similar comparison of  $f_{D^+}$ , in Fig. 2(b), is less instructive, because the chiral extrapolations in Refs. [18, 19] started at large quark masses and are, hence, less reliable than in the present work.



**Figure 2:** Dependence of (a)  $f_{D_s}$  and (b)  $f_{D^+}$  on the number  $n_f$  of sea flavors. Quenched ( $n_f = 0$ ) [18];  $n_f = 2$  [19];  $n_f = 3$  [15]. Solid (dashed) error bars are statistical (statistical+systematic).

#### 4. Mass of the $B_c$ Meson

The pseudoscalar  $B_c^+$  meson is the lowest-lying bound state of a charmed quark and a  $b$  quark. CDF [20] first observed it during Run I of the Tevatron in the semileptonic decay  $B_c^+ \rightarrow J/\psi l^+ \nu$ . During Run II, DØ has confirmed the discovery in the same mode [21]. Because the neutrino is undetected, the mass resolution in semileptonic modes is poor,  $\pm(300\text{--}400)$  MeV. Now, however, the upgraded detectors are able to reconstruct hadronic modes, such as  $B_c^+ \rightarrow J/\psi \pi^+$ , which give much better precision on  $m_{B_c}$  [22].

At Lattice 2004 we presented results in nearly final form [23], and posted the final results on the arXiv in mid-November [24]:

$$m_{B_c} = 6304 \pm 12^{+18}_- 0 \text{ MeV}, \quad (4.1)$$

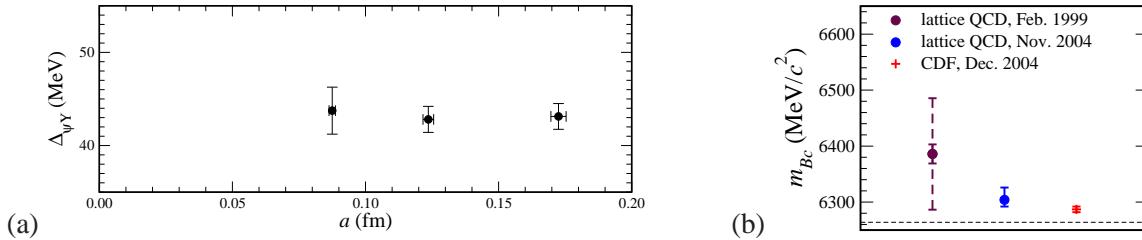
where the last error is a rough estimate of residual heavy-quark discretization effects. Soon afterwards, CDF announced a precise mass measurement. They find [25]

$$m_{B_c} = 6287 \pm 5 \text{ MeV}, \quad (4.2)$$

which agrees with Eq. (4.1) at slightly more than  $1-\sigma$ .

Two comments are in order. First, the agreement at the gross level of the calculation with experiment shows that discretization effects are well under control with lattice NRQCD [27] and the

Fermilab method [28]. Of course, this follows from the careful application of effective field theories for heavy quarks [29, 30]. Indeed, as seen in Fig. 3(a), almost no lattice spacing dependence is seen in the splitting  $\Delta_{\psi\gamma} = m_{B_c} - (\bar{m}_\psi + m_\gamma)/2$  that is at the crux of the calculation [26]. Moreover, it is striking how much the splitting  $\Delta_{\psi\gamma}$  changes when sea quarks are included. Figure 3(b) compares Eq. (4.1) with an old quenched calculation [26] (and the measurement [25]). The solid error bar shows the non-quenching errors, and the dashed includes the estimate of the quenching error. The inclusion of sea quarks has reduced the splitting by a factor of three or four, bringing an essentially discrepant result into agreement.



**Figure 3:** (a) Dependence of the splitting  $\Delta_{\psi\gamma}$  on the lattice spacing  $a$ . (b) Comparison of the quenched [26],  $n_f = 2 + 1$  [24], and experimental [25] values of  $m_{B_c}$ ; the dashed line denotes the baseline  $(\bar{m}_\psi + m_\gamma)/2$ .

## 5. Conclusions

In the past year, three lattice-QCD calculations have been confirmed by experiment. FOCUS [10] confirmed the  $q^2$ -dependence of the  $D \rightarrow Kl\nu$  form factor [8]; CLEO-c [17] confirmed the  $D$ -meson decay constant [15]; and CDF [25] confirmed the mass of the  $B_c$  meson [24]. To obtain these results it is essential to have heavy-quark discretization effects under control, as one expects from theoretical foundations [27, 28, 29, 30]. Furthermore, the comparison of quenched QCD, QCD with 2+1 staggered flavors, and experiment shows that sea quarks are needed to obtain agreement, and that staggered quarks (in these cases) capture the needed effect.

This work has been supported in part by the U.S. National Science Foundation, the Office of Science of the U.S. Department of Energy (DOE), and the U.K. Particle Physics and Astronomy Research Council. Fermilab is operated by Universities Research Association Inc., under contract with the DOE.

## References

- [1] C. T. H. Davies *et al.* [HPQCD, MILC, and Fermilab Lattice Collaborations], Phys. Rev. Lett. **92**, 022001 (2004) [hep-lat/0304004].
- [2] C. Aubin and C. Bernard, Phys. Rev. D **68**, 034014 (2003) [hep-lat/0304014]; *ibid.*, 074011 (2003) [hep-lat/0306026].
- [3] S. Dürr, “Theoretical issues with staggered fermion simulations,” hep-lat/0509026.
- [4] I. Shipsey, Nucl. Phys. Proc. Suppl. **140**, 58 (2005) [hep-lat/0411009].
- [5] M. Okamoto *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Nucl. Phys. Proc. Suppl. **129**, 334 (2004) [hep-lat/0309107].

- [6] M. Ablikim *et al.* [BES Collaboration], Phys. Lett. B **597**, 39 (2004) [hep-ex/0406028].
- [7] G. S. Huang *et al.* [CLEO Collaboration], Phys. Rev. Lett. **94**, 011802 (2005) [hep-ex/0407035].
- [8] C. Aubin *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Phys. Rev. Lett. **94**, 011601 (2005) [hep-ph/0408306].
- [9] D. Bećirević and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000) [hep-ph/9904490].
- [10] J. M. Link *et al.* [FOCUS Collaboration], Phys. Lett. B **607**, 233 (2005) [hep-ex/0410037].
- [11] P.B. Mackenzie *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], work in progress.
- [12] M. Ablikim *et al.* [BES Collaboration], Phys. Lett. B **610**, 183 (2005) [hep-ex/0410050].
- [13] G. Bonvicini *et al.* [CLEO Collaboration], Phys. Rev. D **70**, 112004 (2004) [hep-ex/0411050].
- [14] J. N. Simone *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Nucl. Phys. Proc. Suppl. **140**, 443 (2005) [hep-lat/0410030].
- [15] C. Aubin *et al.*, [Fermilab Lattice, MILC, and HPQCD Collaborations], Phys. Rev. Lett. **95**, 122002 (2005) [hep-lat/0506030];  
J.N. Simone *et al.*, [Fermilab Lattice, MILC, and HPQCD Collaborations], these proceedings.
- [16] A. S. Kronfeld, Nucl. Phys. Proc. Suppl. **129**, 46 (2004) [hep-lat/0310063].
- [17] M. Artuso, talk at the XXII International Symposium on Lepton-Photon Interactions at High Energy, Uppsala, Sweden, <http://1p2005.ts1.uu.se/~1p2005/> (June 30–July 5, 2005);  
M. Artuso *et al.* [CLEO Collaboration], “Improved measurement of  $B(D^+ \rightarrow \mu^+\nu)$  and the pseudoscalar decay constant  $f_{D^+}$ ,” hep-ex/0508057.
- [18] A. X. El-Khadra, A. S. Kronfeld, P. B. Mackenzie, S. M. Ryan, and J. N. Simone, Phys. Rev. D **58**, 014506 (1998) [hep-ph/9711426].
- [19] C. Bernard *et al.* [MILC Collaboration], Phys. Rev. D **66**, 094501 (2002) [hep-lat/0206016].
- [20] F. Abe *et al.* [CDF Collaboration], Phys. Rev. Lett. **81**, 2432 (1998) [hep-ex/9805034].
- [21] DØ Collaboration, DØ Note 4539-CONF (August 14, 2004);  
E. Cheu [DØ Collaboration], Int. J. Mod. Phys. A **20**, 3664 (2005).
- [22] K. Anikeev *et al.*, “ $B$  physics at the Tevatron: Run II and beyond,” hep-ph/0201071.
- [23] I. F. Allison *et al.* [HPQCD and Fermilab Lattice Collaborations], Nucl. Phys. Proc. Suppl. **140**, 440 (2005) [hep-lat/0409090].
- [24] I. F. Allison *et al.* [HPQCD and Fermilab Lattice Collaborations], Phys. Rev. Lett. **94**, 172001 (2005) [hep-lat/0411027].
- [25] D. Acosta *et al.* [CDF Collaboration], “Evidence for the exclusive decay  $B_c^\pm \rightarrow J/\psi\pi^\pm$  and measurement of the mass of the  $B_c$  meson,” hep-ex/0505076.
- [26] H. P. Shanahan, P. Boyle, C. T. H. Davies, and H. Newton [UKQCD Collaboration], Phys. Lett. B **453**, 289 (1999) [hep-lat/9902025].
- [27] G. P. Lepage and B. A. Thacker, Nucl. Phys. Proc. Suppl. **4**, 199 (1987);  
B. A. Thacker and G. P. Lepage, Phys. Rev. D **43**, 196 (1991).
- [28] A. X. El-Khadra, A. S. Kronfeld, and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997) [hep-lat/9604004].
- [29] G. P. Lepage *et al.*, Phys. Rev. D **46**, 4052 (1992) [hep-lat/9205007].
- [30] A. S. Kronfeld, Phys. Rev. D **62**, 014505 (2000) [hep-lat/0002008].

Semileptonic  $D \rightarrow \pi/K$  and  $B \rightarrow \pi/D$  decays in 2+1 flavor lattice QCD

M. Okamoto<sup>a</sup>, C. Aubin<sup>b</sup>, C. Bernard<sup>b</sup>, C. DeTar<sup>c</sup>, M. Di Pierro<sup>d</sup>, A. X. El-Khadra<sup>e</sup>, Steven Gottlieb<sup>f</sup>, E. B. Gregory<sup>g</sup>, U. M. Heller<sup>h</sup>, J. Hetrick<sup>i</sup>, A. S. Kronfeld<sup>a</sup>, P. B. Mackenzie<sup>a</sup>, D. P. Menscher<sup>e</sup>, M. Nobes<sup>j</sup>, M. B. Oktay<sup>e</sup>, J. Osborn<sup>c</sup>, J. N. Simone<sup>a</sup>, R. Sugar<sup>k</sup>, D. Toussaint<sup>g</sup>, H. D. Trottier<sup>j</sup>

<sup>a</sup> Fermi National Accelerator Laboratory, P.O. Box 500, Batavia, IL 60510

<sup>b</sup> Department of Physics, Washington University, St. Louis, Missouri 63130

<sup>c</sup> Physics Department, University of Utah, Salt Lake City, Utah 84112

<sup>d</sup> School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois 60604

<sup>e</sup> Department of Physics, University of Illinois, Urbana, IL 61801

<sup>f</sup> Department of Physics, Indiana University, Bloomington, IN 47405

<sup>g</sup> Department of Physics, University of Arizona, Tucson, Arizona 85721

<sup>h</sup> American Physical Society, One Research Road, Box 9000, Ridge, New York 11961-9000

<sup>i</sup> University of the Pacific, Stockton, California 95211

<sup>j</sup> Physics Department, Simon Fraser University, Vancouver, British Columbia, Canada

<sup>k</sup> Department of Physics, University of California, Santa Barbara, California 93106

We present results for form factors of semileptonic decays of  $D$  and  $B$  mesons in 2 + 1 flavor lattice QCD using the MILC gauge configurations. With an improved staggered action for light quarks, we successfully reduce the systematic error from the chiral extrapolation. The results for  $D$  decays are in agreement with experimental ones. The results for  $B$  decays are preliminary. Combining our results with experimental branching ratios, we then obtain the CKM matrix elements  $|V_{cd}|$ ,  $|V_{cs}|$ ,  $|V_{cb}|$  and  $|V_{ub}|$ . We also check CKM unitarity, for the first time, using only lattice QCD as the theoretical input.

## 1. INTRODUCTION

Semileptonic decays of  $B$  and  $D$  mesons play crucial roles in CKM phenomenology. The  $B$  decays such as  $B \rightarrow \pi l\nu$  and  $B \rightarrow D l\nu$  determine  $|V_{ub}|$  and  $|V_{cb}|$ , which are essential to constrain the CKM unitarity triangle. On the other hand, the  $D$  decays such as  $D \rightarrow \pi l\nu$  and  $D \rightarrow K l\nu$  provide a good test of lattice calculations because corresponding CKM matrix elements  $|V_{cd}|$  and  $|V_{cs}|$  are relatively well determined. In this paper, we report lattice calculations of semileptonic decays in unquenched ( $n_f = 2+1$ ) QCD. By using a staggered-type fermion, which is fast to simulate,

$$\begin{pmatrix} |V_{ud}| & |V_{us}| & |V_{ub}| \\ |V_{cd}| & |V_{cs}| & |V_{cb}| \\ 0.24(3)(2) & 0.97(10)(2) & 3.8(1)(6) \times 10^{-2} \\ |V_{td}| & |V_{ts}| & |V_{tb}| \end{pmatrix}$$

Figure 1. Result for CKM matrix. The first errors are theoretical, and the second experimental.

for light quarks, we are able to reduce uncertainties from the “chiral” ( $m_l \rightarrow m_{ud}$ ) extrapolation. We calculate form factors for the above 4 different decays, from which the 4 CKM matrix elements are determined, as summarized in Fig. 1. The results for  $D$  decays are published in Ref. [1].

## 2. SIMULATION DETAILS

We use  $n_f = 2 + 1$  dynamical gauge configurations obtained with an improved staggered (“Asqtad”) quark action on a lattice with  $a^{-1} \approx 1.6$  GeV, generated by the MILC collaboration [2]. For the valence light quarks we use the same staggered quark action, with the valence light quark ( $u, d$ ) mass  $m_l^{\text{val}}$  equal to the dynamical light quark mass  $m_l^{\text{sea}}$ . The light quark masses we simulate range  $\frac{m_s}{8} \leq m_l \leq \frac{3}{4}m_s$ , where  $m_s$  is the strange quark mass. For the valence charm( $c$ ) and bottom( $b$ ) quarks we use a tadpole-improved clover action with the Fermilab interpretation [3]. The hopping parameter for the  $c(b)$  quark is fixed from the  $D_s(B_s)$  mass.

To form the heavy-light bilinears from the staggered-type light quark and the Wilson-type heavy quark, we convert the staggered-type quark to the naive-type quark, as in Refs. [4,5]. Relevant 3-point functions are then computed in the initial state meson rest frame using local sources and local sinks. We typically accumulate about 500 configurations, and results at 2-4 source times are averaged to increase the statistics.

For the matching factor of vector current  $Z_{V_\mu}^{ab}$ , we follow the method in Refs. [6,7], writing  $Z_{V_\mu}^{ab} = \rho_{V_\mu} (Z_V^{aa} Z_V^{bb})^{1/2}$ . The flavor-conserving renormalization factors  $Z_V^{aa(bb)}$  are determined nonperturbatively from charge normalization conditions. For the remaining factor  $\rho_{V_\mu}$  we use results in one-loop perturbation theory [8].

## 3. RESULTS

### 3.1. $D \rightarrow \pi(K)$ and $B \rightarrow \pi$

The heavy-to-light decay amplitudes are parameterized as

$$\begin{aligned} \langle P|V^\mu|H\rangle &= f_+(q^2)(p_H + p_P - \Delta)^\mu + f_0(q^2)\Delta^\mu \\ &= \sqrt{2m_H} [v^\mu f_\parallel(E) + p_\perp^\mu f_\perp(E)] \end{aligned}$$

with  $q = p_H - p_P$ ,  $\Delta^\mu = (m_H^2 - m_P^2)q^\mu/q^2$ ,  $v = p_H/m_H$ ,  $p_\perp = p_P - Ev$  and  $E = E_P$ . The differential decay rate  $d\Gamma/dq^2$  is proportional to  $|V_{CKM}|^2 |f_+(q^2)|^2$ . Below we briefly describe our analysis procedure; see Ref. [1] for details.

We first extract the form factors  $f_\parallel$  and  $f_\perp$ , as in Ref. [6], and carry out the chiral extrapolation

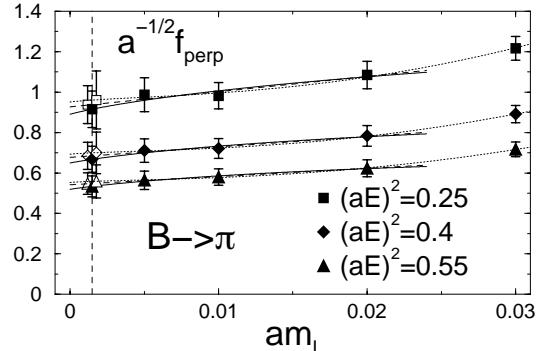


Figure 2.  $m_l$ -dependence and chiral fits for  $f_\perp^{B \rightarrow \pi}$ .

in  $m_l$  for them at fixed  $E$ . To this end, we interpolate and extrapolate the results for  $f_\parallel$  and  $f_\perp$  to common values of  $E$  using the parametrization of Becirevic and Kaidalov (BK) [9]. We perform the chiral extrapolation using the NLO correction in staggered chiral perturbation theory ( $S\chi PT$ ) [10]. We try various fit forms [1], as shown in Fig. 2, and the differences between the fits are taken as associated systematic errors.

We then convert the results for  $f_\perp$  and  $f_\parallel$  at  $m_l = m_{ud}$ , to  $f_+$  and  $f_0$ . To extend  $f_+$  and  $f_0$  to functions of  $q^2$ , we again make a fit using BK parameterization [9],

$$f_+(q^2) = \frac{f_+}{(1 - \tilde{q}^2)(1 - \alpha\tilde{q}^2)}, \quad f_0(q^2) = \frac{f_+}{1 - \tilde{q}^2/\beta},$$

where  $\tilde{q}^2 = q^2/m_{H^*}^2$ . We obtain

$$f_+^{B\pi} = 0.23(2), \quad \alpha^{B\pi} = 0.63(5), \quad \beta^{B\pi} = 1.18(5),$$

for the  $B \rightarrow \pi$  decay, and

$$\begin{aligned} f_+^{D\pi} &= 0.64(3), \quad \alpha^{D\pi} = 0.44(4), \quad \beta^{D\pi} = 1.41(6), \\ f_+^{DK} &= 0.73(3), \quad \alpha^{DK} = 0.50(4), \quad \beta^{DK} = 1.31(7), \end{aligned}$$

for the  $D$  decays, where the errors are statistical only. To estimate the error from BK parameterization, we also make an alternative analysis, where we perform a 2-dimensional polynomial fit in  $(m_l, E)$ . A comparison between the two analyses are shown in Fig. 3.

Finally we determine the CKM matrix elements (Fig. 1) by integrating  $|f_+(q^2)|^2$  over  $q^2$  and using experimental branching ratios [11,12]. For  $|V_{ub}|$  we use the branching ratio for  $q^2 \geq 16$  GeV $^2$  in Ref. [12]. The systematic errors are summarized in Table 1. The results for  $D$  decays agree with experimental results [1].

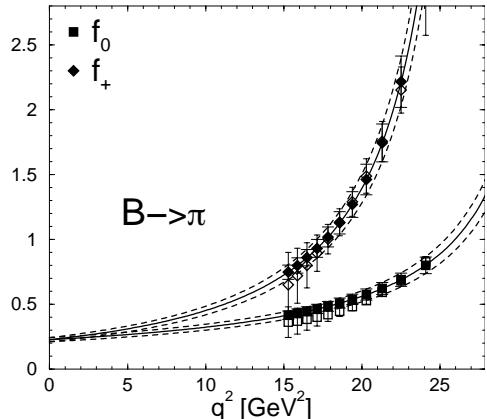


Figure 3.  $B \rightarrow \pi$  form factors from BK-based (filled) and non-BK-based (open) analyses.

### 3.2. $B \rightarrow D$

The  $B \rightarrow D$  amplitude is parameterized as

$$\langle D | V^\mu | B \rangle = \sqrt{m_B m_D} \times [h_+(w)(v + v')^\mu + h_-(w)(v - v')^\mu],$$

where  $v = p_B/m_B$ ,  $v' = p_D/m_D$  and  $w = v \cdot v'$ . The differential decay rate of  $B \rightarrow D l \bar{\nu}$  is proportional to the square of  $\mathcal{F}(w)$ , which is a linear combination of  $h_+(w)$  and  $h_-(w)$ . We calculate the form factors at  $w = 1$  by employing the double ratio method [13]. The light quark mass dependence for  $\mathcal{F}(1)$  is shown in Fig. 4. Extrapolating the result linearly to  $m_l \rightarrow 0$ , we obtain

$$\mathcal{F}_{B \rightarrow D}^{n_f=2+1}(1) = 1.074(18)(16), \quad (1)$$

where the first error is statistical, and the second is systematic summarized in Table 1. The systematic error associated with finite lattice spacing is estimated by doing quenched calculations at different lattice spacings and using different quark actions, and found to be small.

Using Eq. (1) and an experimental result for  $|V_{cb}| \mathcal{F}(1)$  [14], we obtain  $|V_{cb}|$  as given in Fig. 1.

Table 1  
Systematic errors.

decay	$D \rightarrow \pi(K)$	$B \rightarrow \pi$	$B \rightarrow D$
3-pt function	3%	3%	1%
BK fit	2%	4%	
$m_l$ extrap	3%(2%)	4%	1%
matching	<1%	1%	1%
$a$ uncertainty	1%	1%	
finite $a$ error	9%	9%	<1%
total	10%	11%	2%

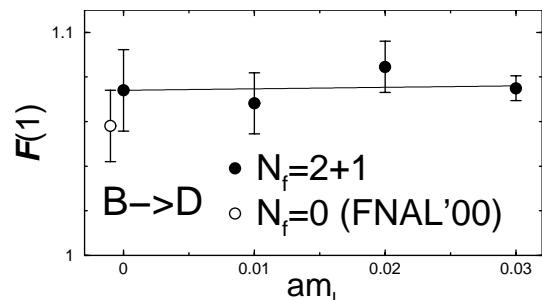


Figure 4.  $m_l$ -dependence for  $\mathcal{F}_{B \rightarrow D}(1)$ .

Since we have *all* 3 elements of the second row of CKM matrix, we are able to check a CKM unitarity using *only* our results as theoretical inputs;  $(|V_{cd}|^2 + |V_{cs}|^2 + |V_{cb}|^2)^{1/2} = 1.00(10)(2)$ .

**Acknowledgments:** We thank the Fermilab Computing Division, the SciDAC program, the Theoretical High Energy Physics Programs at the DOE and NSF, and URA for their support.

## REFERENCES

1. C. Aubin *et al.*, arXiv:hep-ph/0408306.
2. C. Bernard *et al.*, Phys. Rev. D **64**, 054506 (2001).
3. A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
4. M. Wingate *et al.*, Phys. Rev. D **67**, 054505 (2003).
5. M. Okamoto *et al.*, Nucl. Phys. Proc. Suppl. **129**, 334 (2004).
6. A. X. El-Khadra *et al.*, Phys. Rev. D **64**, 014502 (2001).
7. J. Harada *et al.*, Phys. Rev. D **65**, 094514 (2002); *ibid.* **65**, 094513 (2002).
8. M. Nobes *et al.*, work in progress.
9. D. Becirevic and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000).
10. C. Aubin and C. Bernard, arXiv:hep-lat/0409027, and work in progress.
11. S. Eidelman *et al.*, Phys. Lett. B **592**, 1 (2004).
12. S. B. Athar *et al.*, Phys. Rev. D **68**, 072003 (2003). (See also K. Abe *et al.*, hep-ex/0408145.)
13. S. Hashimoto *et al.*, Phys. Rev. D **61**, 014502 (2000).
14. K. Abe *et al.*, Phys. Lett. B **526**, 258 (2002).

# A study of the $B_s - \bar{B}_s$ mass and width difference in 2+1 flavor lattice QCD

---

**R. T. Evans\*** and **A. X. El-Khadra**

*Physics Department, University of Illinois, Urbana, Illinois, USA*

*E-mail:* rtevans@uiuc.edu

**M. Di Pierro**

*School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, Illinois, USA*

**Fermilab Lattice and MILC Collaborations**

We present a preliminary calculation of the hadronic matrix elements relevant to  $B_s - \bar{B}_s$  mixing.

The calculation is done on MILC lattices with 2+1 sea quarks. We use the Asqtad action for the light valence quarks and the Fermilab action for the  $b$  quark.

PoS(LAT2006)081

*XXIVth International Symposium on Lattice Field Theory*

*July 23-28, 2006*

*Tucson, Arizona, USA*

---

\*Speaker.

## 1. Introduction

Mixing in the  $B_s - \bar{B}_s$  system is sensitive to the CKM matrix element  $V_{ts}$  and can help to constrain CP violating effects. It is the mass and width difference ( $\Delta m_s$  and  $\Delta\Gamma_s$ ) between the mass eigenstates of this system that are measurable.

The  $B_s - \bar{B}_s$  meson system's mass difference,  $\Delta m_s$ , is parametrized via an effective hamiltonian as

$$\Delta m_s = \frac{G_F^2}{6\pi^2} m_W^2 \eta_B(\mu_B) S_0(m_t, m_W) |V_{ts} V_{tb}^*|^2 \langle \bar{B}_s | Q(\mu_B) | B_s \rangle, \quad (1.1)$$

where  $\eta_B$  is a Wilson coefficient and  $S_0(m_t, m_W)$  is known as the Inami-Lim function, and the scale  $\mu_B \approx m_B$ .

The hadronic matrix element is conventionally parametrized as

$$\langle \bar{B}_s | Q | B_s \rangle = \langle \bar{B}_s | \bar{b} \gamma_\mu (1 - \gamma_5) s \bar{b} \gamma^\mu (1 - \gamma_5) s | B_s \rangle = \frac{8}{3} m_{B_s}^2 f_{B_s}^2 B_{B_s}. \quad (1.2)$$

$f_{B_s}$  is the decay constant of the  $B_s$  meson, and  $B_{B_s}$  is the bag parameter.

The difference in decay rates of the eigenstates in the neutral  $B_s$  meson system is another measurable quantity which is sensitive to CP violation. The width difference

$$\Delta\Gamma_{B_s} = \frac{G_F^2 m_b^2}{6\pi^2 m_{B_s}} |V_{cb}^* V_{cs}|^2 [F\left(\frac{m_c^2}{m_b^2}\right) \langle \bar{B}_s | Q | B_s \rangle + F_S\left(\frac{m_c^2}{m_b^2}\right) \langle \bar{B}_s | Q_S | B_s \rangle] [1 + O\left(\frac{\Lambda_{QCD}}{m_b}\right)] \quad (1.3)$$

is determined by two hadronic matrix elements at the leading order in the heavy quark expansion.  $Q$  is the familiar operator from the mass difference, and  $Q_S$  is parametrized in terms of  $B_S$  or  $B'_S$  in a similar way

$$\langle \bar{B}_s | Q_S | B_s \rangle = \langle \bar{B}_s | \bar{s}(1 - \gamma_5) b \bar{s}(1 - \gamma_5) b | B_s \rangle = -\frac{5}{3} \frac{m_{B_s}^2}{(m_b + m_s)^2} m_{B_s}^2 f_{B_s}^2 B_S = -\frac{5}{3} m_{B_s}^2 f_{B_s}^2 B'_S. \quad (1.4)$$

Precise measurements of  $\Delta m_s$  have recently been made [7], yielding a determination of  $V_{ts}$  and a significant reduction of the allowed region in the  $\rho - \eta$  plane due to this constraint [9]. The errors on  $V_{ts}$  are now completely dominated by theoretical uncertainties from lattice QCD calculations of  $f_{B_s}^2 B_{B_s}$ .  $\Delta\Gamma_s$  has also already been measured and we can expect improved measurements by the end of the current Tevatron run [8]. The goal of this work is to use the unquenched MILC lattices for a precise determination of the above matrix elements in the  $B_s$  and  $B_d$  systems.

## 2. Lattice Parameters

We performed our calculations on the MILC coarse lattices ( $a = 0.12\text{ fm}$ ) with 2+1 sea quarks on 592 configurations. The sea quarks are simulated using the Asqtad improved action, where errors are introduced at  $O(a^4, \alpha_s a^2)$ . The valence light quark propagators were created using the Asqtad action, and the heavy  $b$  quark is handled using the Fermilab action, with errors starting at  $O(a^2, \alpha_s a)$ .

We used pre-existing staggered propagators of two valence quark masses,  $m_q = 0.0415$  and  $0.005$ . The first mass value is very close to the physical  $s$  quark mass and the second mass value is the closest available to the physical  $d$  quark mass, giving us a rough comparison between the

B<sub>s</sub> and B<sub>d</sub> systems. We used  $\kappa_b = 0.086$  for the heavy quark. We performed the calculation for  $m_q = 0.0415$  at two different time sources, whereas only one time source was used for  $m_q = 0.005$ . We used both a 1S wavefunction and a delta function to smear the heavy quark at the sink.

For tree level  $O(\frac{\Lambda_{\text{QCD}}}{m_b})$  improvement of the operator we found that a rotation of the b quark via [1] is all that is necessary. In order to include  $O(\alpha_s)$  effects additional six dimensional operators must be included. These additional matrix elements can be constructed from the open meson propagator described in Section 3.

### 3. The Open Meson Propagator

All possible 24 matrix elements that can be formed from the general 4-quark operator

$$\langle \bar{H}_q(x) | O_q^{\Delta h=2} | H_q(y) \rangle = \langle \bar{H}_q(x) | \bar{q} \Gamma_1 h \bar{q} \Gamma_2 h(0) | H_q(y) \rangle \quad (3.1)$$

can be calculated using only one inversion per quark flavor by placing the operator at the origin. After performing the Wick contractions and Fourier transforming Eq. (3.1) we obtain

$$\begin{aligned} \sum_{\vec{x}, \vec{y}} \langle \bar{H}_q(x) | O_q^{\Delta h=2} | H_q(y) \rangle &= \mathbf{Tr}[\Gamma_1 E_{hq}(t_x)] \mathbf{Tr}[\Gamma_2 E_{hq}(t_y)] + \mathbf{Tr}[\Gamma_1 E_{hq}(t_y)] \mathbf{Tr}[\Gamma_2 E_{hq}(t_x)] \\ &+ \mathbf{Tr}[\Gamma_1 E_{hq}(t_x) \Gamma_2 E_{hq}(t_y)] + \mathbf{Tr}[\Gamma_1 E_{hq}(t_y) \Gamma_2 E_{hq}(t_x)] \end{aligned} \quad (3.2)$$

where the traces are over color and spin indices. The open meson propagator

$$E_{hq,ij}^{ab}(t_x) = \gamma_5^{ac} G_{h,ki}^{*dc}(t_x, 0) G_{q,kj}^{db}(t_x, 0) \quad (3.3)$$

is all that is needed to construct the matrix element. This object is very small in size and can easily be saved and later used to construct all two-point and three-point functions necessary for our calculation [4].

### 4. Matrix Element Extraction

The mixing matrix element, Eq. (1.2), is extracted from the three-point function

$$C_Q(t_1, t_2) = \sum_{\vec{x}, \vec{y}} \langle \bar{b}(\vec{x}, t_1) \gamma_5 q(\vec{x}, t_1) [Q(0)] \bar{b}(\vec{y}, t_2) \gamma_5 q(\vec{y}, t_2) \rangle. \quad (4.1)$$

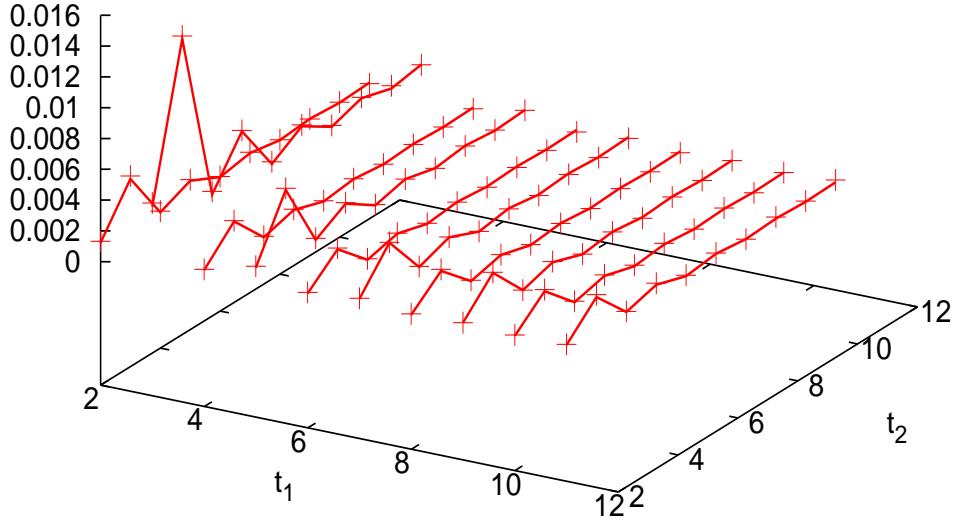
The correlation function has naive valence quarks which contain doublers that cause higher energy 0<sup>+</sup> states to contribute. As can be seen in Fig. 1, these states oscillate in Euclidean time and make a significant contribution to the correlation function [2].

Fig 2. depicts the ratio of the three-point function and two-point functions

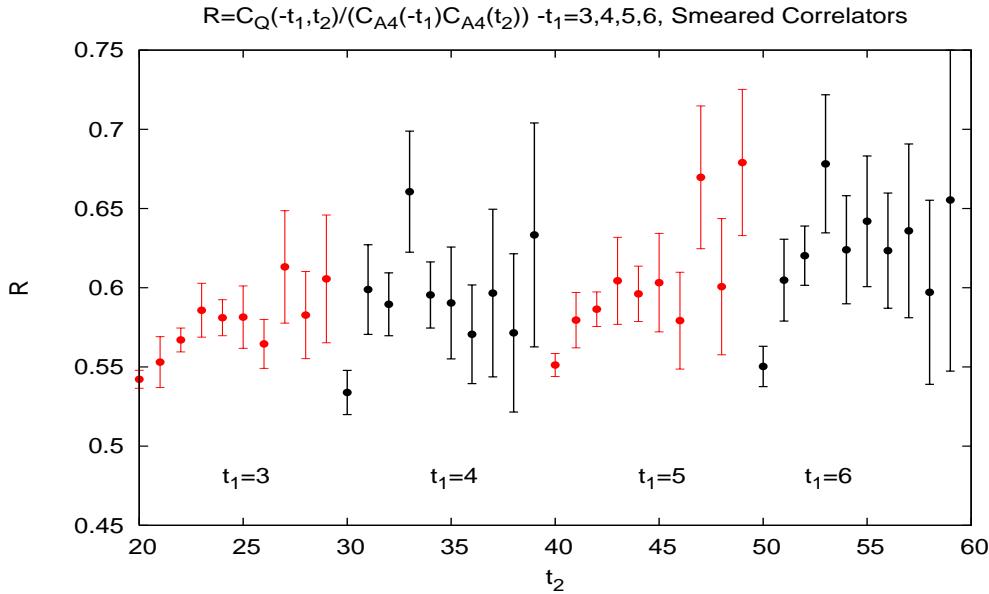
$$R(-t_1, t_2) = \frac{3}{8} \frac{C_Q(-t_1, t_2)}{C_{A4}(-t_1) C_{A4}(t_2)}, \quad (4.2)$$

where

$$C_{A4}(t) = \sum_{\vec{x}} \langle \bar{b}(\vec{x}, t) \gamma_5 q(\vec{x}, t) \bar{q}(0) \gamma_6 \gamma_5 b(0) \rangle. \quad (4.3)$$



**Figure 1:**  $C_Q(t_1, t_2) e^{m_0 t_1 + m_0 t_2}$  with delta function sink,  $m_0$  = ground state mass. Crosses are data and lines are the best fit to the data.



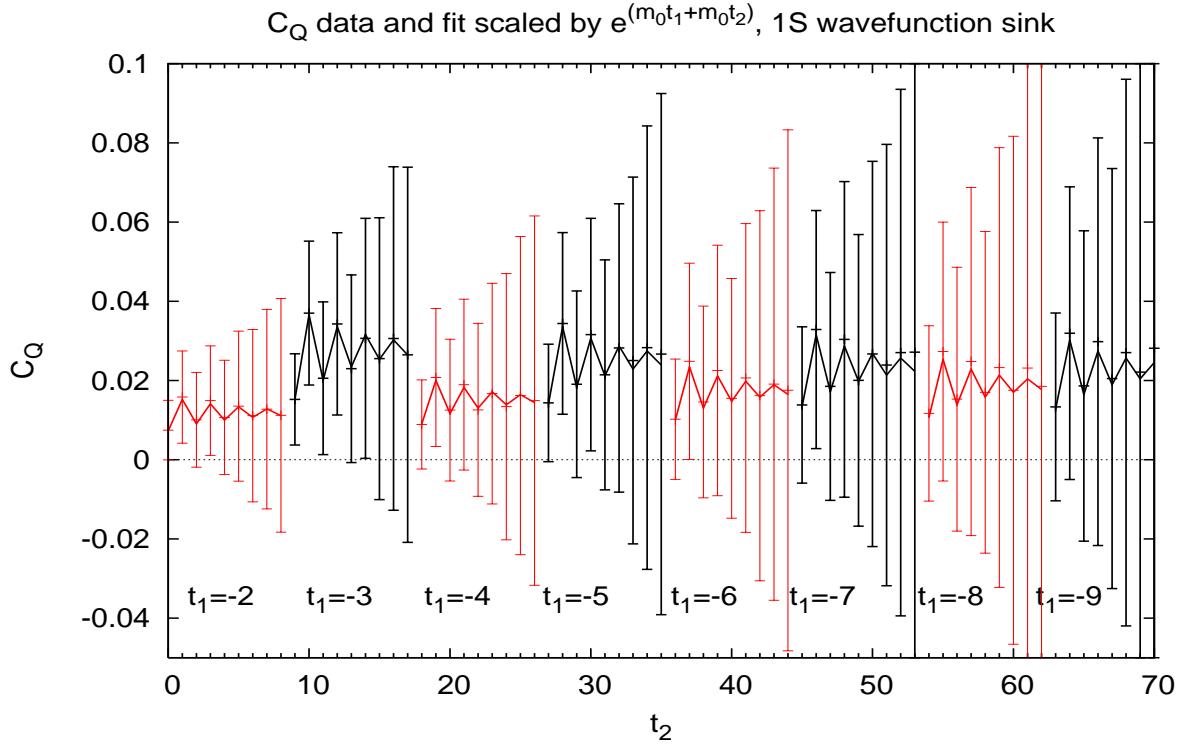
**Figure 2:**  $B$  is placed at the source and  $\bar{B}$  at the sink,  $t_2 = 2 - 11$ .

In the limit  $-t_1, t_2 \rightarrow \infty$   $R$  becomes the bag parameter

$$B_{B_q} = \frac{3}{8} \frac{\langle \bar{B}_q | \bar{b} \gamma_\mu (1 - \gamma_5) q \bar{b} \gamma^\mu (1 - \gamma_5) q | B_q \rangle}{m_{B_q}^2 |\langle \bar{b} \gamma_0 \gamma_5 q | B_q \rangle|^2}, \quad (4.4)$$

which we would hope to see as a plateau in Fig. 2. The oscillating states make the clear identification of a plateau in the ratio and fitting to it difficult. We are examining other ratios to determine the possibility of fitting to these, but are currently fitting to  $C_Q$  directly.

In order to extract the matrix elements of interest we performed constrained fits [3] simultaneously to 3 correlation functions: the two-point functions



**Figure 3:**  $C_Q$ , 1S wavefunction sink,  $t_2 = 2 - 11$

$$C_Z(t) \rightarrow_{t \rightarrow \infty} \frac{1}{2m_{B_q}} |\langle \bar{q}\gamma_5 b | B_q \rangle|^2 e^{-m_{B_q} t}, \quad C_{A_4}(t) \rightarrow_{t \rightarrow \infty} \frac{1}{2m_{B_q}} \langle \bar{q}\gamma_5 b | B_q \rangle \langle B_q | \bar{b}\gamma_0\gamma_5 q \rangle e^{-m_{B_q} t} \quad (4.5)$$

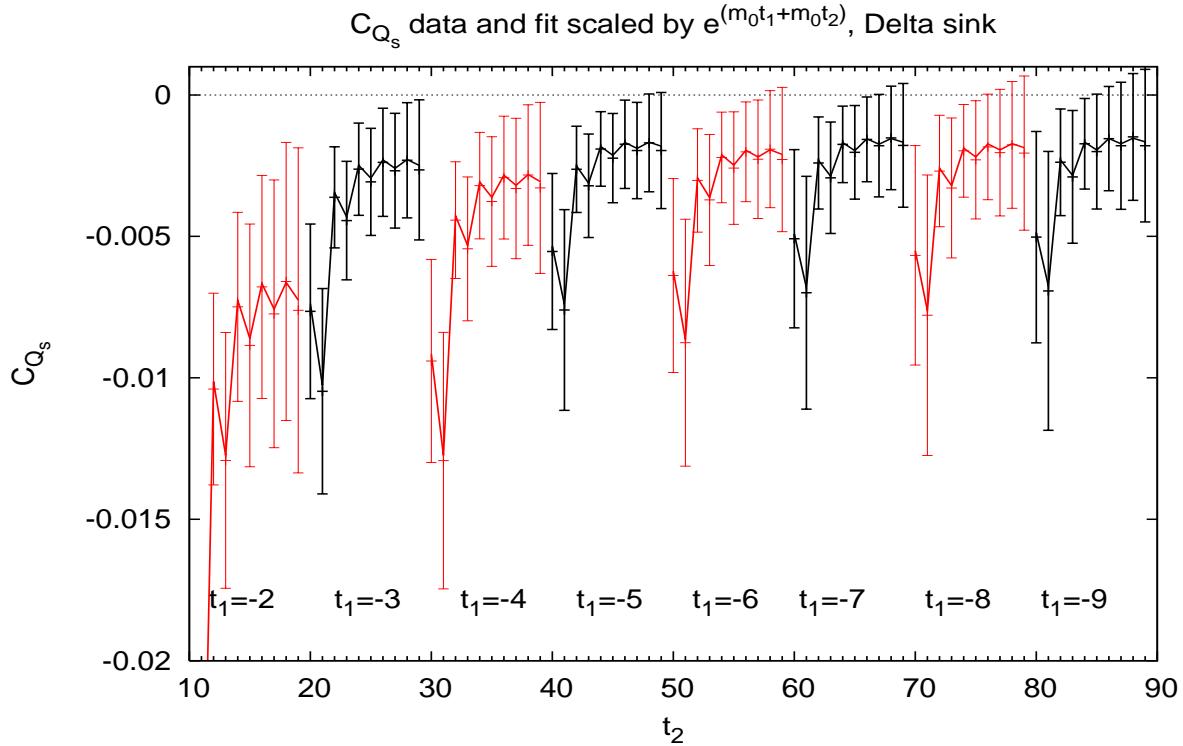
and three-point function

$$C_Q(-t_1, t_2) \rightarrow_{t_1, t_2 \rightarrow \infty} \frac{1}{(2m_{B_q})^2} |\langle \bar{q}\gamma_5 b | B_q \rangle|^2 \langle \bar{B} | Q | B_q \rangle e^{-m_{B_q} t_1} e^{-m_{B_q} t_2}. \quad (4.6)$$

$C_Z$  allows the overlap parameters in  $C_Q$  to be removed and the matrix element isolated.  $C_{A_4}$  is used to determine  $f_{B_q}$  and can be used to isolate  $B_{B_q}$ . The parameter most directly of phenomenological interest,  $f_{B_q} \sqrt{B_{B_q}}$ , can be extracted by combining just  $C_Z$  and  $C_Q$ .

In addition to the ground state other excited states contribute, in particular the opposite parity oscillating states arising from the naive valence quark. For our best fits to the data with a delta function sink we included the first 6 states (3 regular and 3 oscillating), with  $t_1$  and  $t_2$  taken over  $t_{min} = 1$ ,  $t_{max} = 12$ , giving a  $\chi^2 \approx 1.0$ . The best fits using 1S smeared data were obtained by including the first 4 states from  $t_{min} = 1$ ,  $t_{max} = 11$ , also resulting in a  $\chi^2 \approx 1.0$ . As illustrated in Figs. 1, 3, and 4 our fits give a reliable description of the data over almost the entire  $t_1 - t_2$  plane. The parameter values extracted from these fits are listed in Table 1.

The fit results using different numbers of states, 2 – 8, were typically consistent within 50% of the error bars of the best fit. The larger errors observed in the  $m_q = 0.005$  fits are to be expected, as only one time source was used and statistical fluctuations of the data increase closer to the chiral



**Figure 4:**  $C_{Q_s}$ , delta function sink,  $t_2 = 2 - 12$

limit. With more statistics and experience we hope to improve the robustness of the fits further. We calculated the matrix element of  $Q_S$  in an identical way, with similar results (see Fig. 4).

The errors reported in Table 1 are  $\chi^2$  errors from the fitting. The results do not include the renormalization coefficients or chiral extrapolations. The  $m_q = 0.0415$  fit results are greatly improved by using smearing, with the errors being halved in some cases.

Smearing	$m_q$	$B_{B_q}$	$f_{B_q} \sqrt{B_{B_q}}$	$B'_S$	$f_{B_q} \sqrt{B'_S}$
delta	0.0415	0.62 +/- 0.06	0.160 +/- 0.006	2.44 +/- 0.15	0.329 +/- 0.007
	0.005	0.62 +/- 0.10	0.150 +/- 0.009	2.36 +/- 0.29	0.306 +/- 0.013
1S	0.0415	0.59 +/- 0.03	0.160 +/- 0.003	2.40 +/- 0.10	0.325 +/- 0.005
	0.005	0.69 +/- 0.13	0.144 +/- 0.010	2.64 +/- 0.39	0.274 +/- 0.014

**Table 1:** Smeared and unsmeared results in lattice units. The  $m_q = 0.005$  results are derived from half the time sources of the  $m_q = 0.0415$  results.

## 5. Summary and Outlook

The statistical uncertainties of this calculation are straightforward to reduce. Specifically, we plan to repeat the calculation on the same ensemble, but with more time sources. Improving the fitting procedure may also aid in reducing errors.

The calculation thus far is done with  $O(a)$  improvement and only tree-level matching. We are planning to include the perturbative matching at one loop order at which point additional operators must be included. The three-point functions for the additional operators can easily be constructed from our stored open meson propagators, making the inclusion of these operators straightforward once their coefficients have been calculated. The NLO operators in the  $(\frac{\Lambda_{\text{QCD}}}{m_b})$  expansion contribute significantly to  $\Delta\Gamma_s$ , and will also have to be calculated [6].

We are also planning to repeat this calculation on the available MILC ensembles for various sea quark/light valence quark masses and lattice spacings in order to observe the light quark mass and lattice spacing dependence of our results. A comparison of our  $m_q = 0.0415$  and  $0.005$  results shows a mild dependence, although it should be stressed that the errors in the  $m_q = 0.005$  fits are very large. With the full data set we plan to use staggered chiral perturbation theory to extract the parameters at the physical masses.

## Acknowledgments

The numerical simulations for this work were carried out on the Fermilab lattice QCD clusters, which are a computing resource of the USQCD collaboration and are funded by the DOE. We are grateful to the Fermilab Computing Division for operating and maintaining the clusters. This work was supported in part by the DOE under grant no. DE-FG02-91ER40677.

## References

- [1] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55** (1997) 3933 [arXiv:hep-lat/9604004].
- [2] M. Wingate, J. Shigemitsu, C. T. H. Davies, G. P. Lepage and H. D. Trottier, Phys. Rev. D **67** (2003) 054505 [arXiv:hep-lat/0211014].
- [3] G. P. Lepage, B. Clark, C. T. H. Davies, K. Hornbostel, P. B. Mackenzie, C. Morningstar and H. Trottier, Nucl. Phys. Proc. Suppl. **106** (2002) 12 [arXiv:hep-lat/0110175].
- [4] M. Di Pierro *et al.* [FermiQCD Collaboration], Nucl. Phys. Proc. Suppl. **129** (2004) 832 [arXiv:hep-lat/0311027].
- [5] A. Gray, C. Davies, E. Gulez, G. P. Lepage, J. Shigemitsu and M. Wingate, “B leptonic decays and B - anti-B mixing with 2+1 flavors of dynamical Nucl. Phys. Proc. Suppl. **140** (2005) 446 [arXiv:hep-lat/0409040].
- [6] M. Beneke, G. Buchalla and I. Dunietz, Phys. Rev. D **54** (1996) 4419 [arXiv:hep-ph/9605259].
- [7] [CDF - Run II Collaboration], Phys. Rev. Lett. **97** (2006) 062003 [arXiv:hep-ex/0606027].
- [8] V. M. Abazov *et al.* [D0 Collaboration], [arXiv:hep-ex/0604046].
- [9] J. Charles *et al.* [CKMfitter Group], “CP violation and the CKM matrix: Assessing the impact of the asymmetric B Eur. Phys. J. C **41** (2005) 1 [arXiv:hep-ph/0406184].

PROCEEDINGS OF  
THE 2006 INTERNATIONAL CONFERENCE ON SCIENTIFIC  
COMPUTING

# CSC'06

**Editor**

**Hamid R. Arabnia**

**Associate Editors**

**George A. Gravvanis**  
**Ashu M. G. Solo**

Las Vegas, Nevada, USA

June 26-29, 2006

©CSREA Press

# Matrix Distributed Processing and Applications

Massimo Di Pierro

School of Computer Science, Telecommunications and Information Systems

DePaul University, 243 S. Wabash Av., Chicago, IL 60604, USA

April 30, 2006

## Abstract

Matrix Distributed Processing (MDP) is a C++ library for fast development of efficient parallel algorithms. MDP is based on MPI and consists of a collection of C++ classes and functions such as lattice, site and field. Algorithms using these components are automatically parallel and no explicit call to communication functions is required. MDP is particularly suitable for implementing parallel solvers for multi-dimensional differential equations and mesh-like problems. MDP includes a simulator that allows one to run and test parallel programs on a single node.

## 1 Introduction

Matrix Distributed Processing (MDP) [1] is a collection of classes and functions written in C++ for fast development of parallel algorithms such as solvers for partial differential equations, mesh-like algorithms, and various types of graph-based problems. These algorithms find frequent application in many sectors of physics, engineering, electronics and computational finance. MDP was originally developed to simplify the coding of parallel Lattice QCD algorithms, i.e. the numerical computation of properties of composite particles made of quarks, such as protons and neutrons. While Lattice QCD is currently one of the main applications of MDP, its range of applicability is not limited to it.

MDP is based on Message Passing Interface (MPI) but parallelization is transparent to the programmer, who does not need to use explicit calls to send/receive functions. It includes functions for linear algebra with support of a Maple-like syntax, statistical functions, and fitting functions. MDP also includes a Parallel SIMulator (PSIM) so that algorithms developed using MDP can be run in parallel on a single processor machine without MPI. The parallel processes are created by forking and communications are realized using local sockets.

MDP can be used on any machine with ANSI C++, POSIX, and MPI. No specific communication hardware is required, but a fast network switch is suggested.

The best way to introduce MDP is by example. Here is an example of how to solve a 3d Laplace equation recursively using MDP.

**Problem:** Consider the following Laplace equation:

$$\nabla^2 \varphi(x) = f(x) \quad (1)$$

where  $\varphi(x)$  is a field of  $2 \times 2$  complex matrices defined on a 3D space (space),  $x = (x_0, x_1, x_2)$  limited by  $0 \leq x_i < L_i$ , and

$$\begin{aligned} L &= \{10, 10, 10\}, \\ f(x) &= A \sin(2\pi x_1/L_1), \\ A &= \begin{pmatrix} 1 & i \\ 3 & 1 \end{pmatrix} \end{aligned} \quad (2)$$

The initial conditions are  $\varphi_{initial}(x) = 0$ . We will also assume that  $x_i + L_i = x_i$  (torus topology).

**Solution:** In order to solve eq. (1) we first discretize the Laplacian ( $\nabla^2 = \partial_0^2 + \partial_1^2 + \partial_2^2$ ) and rewrite it as

$$\sum_{\mu=0,1,2} [\varphi(x + \hat{\mu}) - 2\varphi(x) + \varphi(x - \hat{\mu})] = f(x) \quad (3)$$

where  $\hat{\mu}$  is a unit vector in the discretized space in direction  $\mu$ . Hence we solve it in  $\varphi(x)$  and obtain the following a recurrence relation

$$\varphi(x) = \frac{\sum_{\mu=0,1,2} [\varphi(x + \hat{\mu}) + \varphi(x - \hat{\mu})] - f(x)}{6} \quad (4)$$

The following is a typical MDP program that solves eq. (1) by recursively iterating eq. (4). Notice how the program is parallel but there are no explicit call to communication functions:

```

00 #include "mdp.h"
01
02 void main(int argc, char** argv) {
03     mdp.open_wormholes(argc,argv);      // open communications
04     int L[]={10,10,10};                // declare volume
05     mdp_lattice    space(3,L);         // declare lattice
06     mdp_site       x(space);          // declare site variable
07     mdp_matrix_field phi(space,2,2);  // declare field of 2x2
08     mdp_matrix     A(2,2);            // declare matrix A
09     A(0,0)=1;   A(0,1)=I;
10     A(1,0)=3;   A(1,1)=1;
11     forallsites(x)                  // loop (in parallel)
12         phi(x)=0;                  // initialize the field
13     phi.update();                  // communicate!
14
15     for(int i=0; i<1000; i++) {      // iterate 1000 times
16         forallsites(x)              // loop (in parallel)

```

```

17      phi(x)=(phi(x+0)+phi(x-0)+  

18          phi(x+1)+phi(x-1)+  

19          phi(x+2)+phi(x-2)-  

20          A*sin(2.0*Pi*x(1)/L[1]))/6; // equation  

21      phi.update();           // communicate!  

22  }  

23  phi.save(''field_phi.mdp''); // save field  

24  mdp.close_wormholes();     // close communications  

25 }

```

Notes:

- Line 00 includes the MDP library.
- Lines 03 and 25 respectively open and close the communication channels over the parallel processes.
- Line 04 declares the size of the box  $L = \{L_0, L_1, L_2\}$ .
- Line 05 declares a 3-dimensional lattice, called space, on the box  $L$ . MDP supports up to 10-dimensional lattices. By default, a lattice object is a mesh with torus topology. It is possible to specify alternate topologies, boundary conditions, or parallel partitioning for the lattice. Note that each lattice object contains a parallel random generator.
- Line 06 declares a variable site, called x, that will be used to loop over lattice sites, in parallel.
- Line 07 declares a field of  $2 \times 2$  matrices, called phi, over the lattice space. MDP is not limited to fields of matrices. It is easy to declare fields of any user-defined structure or class.
- Lines 08 through 10 define the matrix A.
- Lines 11 and 12 initialize the field phi. Notice that phi is distributed over the parallel processes and `forallsites` is a parallel loop.
- Line 13 performs communications so that each process becomes aware of changes in the field performed by other processes (*synchronization*).
- Lines 15 through 23 perform 1000 iterations to guarantee convergence. In real life applications one may want to implement some convergence criteria as stopping condition.
- Line 16 loops over all sites in parallel.
- Lines 17 through 20 implement eq. (4). Notice the similarity in notation. Here  $\phi(x)$  is a  $2 \times 2$  complex matrix.
- Line 21 performs *synchronization*.

```

00 #include "mdp.h"
01 void main(int argc, char** argv) {
02     mdp.open_wormholes(argc,argv);
03     int L[]={100};
04     mdp_lattice line(1,L);
05     mdp_field<int> spin(line);
06     mdp_site x(line);
07     int dE=0, H=L[0], dH=0;
08     float kappa=2.0;
09     forallsites(x) spin(x)=+1;
10     while(1) {
11         dH=0;
12         for(int parity= EVEN; parity<= ODD; parity++) {
13             forallsitesofparity(x,parity) {
14                 dE=2*spin(x)*(spin(x-0)+spin(x+0));
15                 if(exp(-kappa*dE)>mdp_random_plain())
16                     { spin(x)*=-1; dH=dH+2*spin(x); }
17             }
18             spin.update(parity);
19         }
20         mdp.add(dH);
21         H=H+dH;
22         mdp << "magnetization=" << H << endl;
23     }
24     mdp.close_wormholes();
25 }
```

In this example lines 3-4 declare a 1D lattice of 100 points (`line`). Line 5 declares a field of integers (`spin`) on this lattice. Line 9 sets all field variables to 1 and line 7 sets the total magnetization `H` for this initial spin configuration.

Line 13 computes the energy variation (`dE`) of each Markov Chain Monte Carlo (MCMC) step. Lines 15-16 perform the Monte Carlo accept-reject. If a change is accepted the spin at site `x` is flipped and the total magnetization changes (line 16).

Note how at each MCMC step, first the code tries to flip the spins at even locations then, after it updates the lattice sites, it tries to flip the spins at odd locations (line 13). This guarantees computation results are independent on parallelization of the lattice line.

Since this even-odd distinction is common in many lattice algorithms, MDP stores all even lattice sites and all odd lattice sites close together. This speeds up loops over one of the two subsets and also speeds up communication. In fact, in this example, we are able to limit the synchronization (`update`) to the site of a given parity (line 18).

- Line 23 saves the field. Note that all fields, including user-defined ones, inherit `save` and `load` methods from the basic `mdp_field` class.
- It should also be noted that all MDP classes and functions are both type and exception safe. Moreover MDP components can be used without knowledge of C pointers and pointer arithmetics.

## 2 Linear Algebra and Other Tools

MDP includes a Linear Algebra package and other tools. Some of the most important classes are:

- class `mdp_real`, that should be used in place of float or double;
- class `mdp_complex`, for complex numbers;
- class `mdp_array`, for vectors and/or multidimensional tensors;
- class `mdp_matrix`, for any kind of complex rectangular matrix;
- class `mdp_measure`, for error propagation;
- class `mdp_jackboot`, a container for jackknife and bootstrap algorithms.

The most notably difference between our linear algebra package and other existing packages is its natural syntax.

For example:

```
mdp_matrix A,B;
A=Random.SU(7);
B=exp(A)+inv(A)*hermitian(A)+5;
```

reads like

$A$  and  $B$  are matrices

$A$  is a random  $SU(7)$  matrix

$B = e^A + A^{-1} A^H + 5 \cdot 1$

Note that each matrix can be resized at will and is resized automatically when a value is assigned.

MDP includes functions for fitting such as the Levenberger-Marquardt algorithm.

## 3 Lattice, Site, and Field

An `mdp_lattice` is the class that describes the space on which fields are defined; it stores the *topology* of the space (by default that of a torus in  $d$  dimensions) and information about *partitioning* of the space over the parallel processes. In MDP, a lattice is a graph, defined as a collection of points (lattice *sites*) connected

by links (they specify the topology). Each site is uniquely mapped to one of the parallel processes.

The only restriction is that the graph must have a degree less than 20. From now on we will assume the default topology of a torus, therefore the lattice should be thought of as a mesh in  $d \leq 10$  dimensions.

The constructor class `mdp_lattice` determines on which process to store each site, determines the neighbors of each site, and the sizes of the buffers where each process keeps copies of those sites that are non-local but are neighbors of the local sites.

The constructor also allocates a parallel random number generator so that each site of the lattice has its own independent random number generator. This is important for parallel Monte Carlo applications of MDP and ensures reproducibility of computations on different architectures.

On each lattice it is possible to allocate fields. Some fields are built-in, for example `mdp_field<mdp_complex>`, i.e. the field of complex numbers. The user can declare any type of field. For example a field of 5 float per lattice site:

```
class S {
public: float S[5];
};
int L[]={10,10,10};
mdp_lattice cube(3,L);
mdp_field<S> psi(cube);
```

This code declares a  $10 \times 10 \times 10$  lattice (`cube`) and a field (`psi`), that lives on the `cube`. The site variables of `psi`,  $\psi(x)$  belong to class `S` (assuming `x` is an `mdp_site` on the `cube`).

User-defined fields can be saved:

```
psi.save("filename");
loaded
psi.load("filename");
and synchronized
psi.update();
```

Synchronization means that all processes will perform MPI communications to make sure all buffers that contain copies of non-local site variables are updated with the proper values. The method `update` should be called immediately after the local site variables of a field have been changed.

Once a field object is declared, in the field constructor, each process dynamically allocates memory for the buffers that store the copies of those sites that are non-local but are neighbors of the local sites. These buffers are created in such a way to ensure optimal communication patterns.

Every time a field changes, for example in a parallel loop such as

```
forallsites(x) phi(x)=0;
```

the program notifies the field that its values have been changed by calling  
`phi.update();`

The method `update` performs all required communication to copy site variables that need to be synchronized between each couple of overlapping processes.

Lattice sites are represented by objects of class `mdp_site`. Site objects can be looped over in parallel loops (such as `forallsites`) but it is also possible to explicitly address a specific site by specifying the site coordinates. Obviously only the process that stores a site locally should address a specific site. Class `mdp_site` has methods to check if a site is local, is it is non-local but a local copy is present and which process stores the site locally.

## 4 Optimal Communication Patterns

In MDP, the lattice objects, according with the lattice topology and the parallel partitioning, determines the optimal way to store site variables in memory and performing parallel communication. This information is then used by the field method `update` that performs the synchronization of the field variables.

Note that MDP does not attempt to overlap computation and communication. By optimal communication pattern we mean that, under the assumptions below, the method `update` minimize network traffic and data copies.

Current optimizations are based on the assumption that each processing node has one and only one network card at the same time and that the network is isotropic (latency and bandwidth for each couple of nodes is the same). This assumption is generally true for Ethernet and Myrinet clusters.

Communications are optimal in the sense that:

- Each process retrieves all non-local site variables in a single send/recv for each process that contains sites which are neighbors of local sites.
- Two processes that do not store neighbor sites do not communicate with each other.
- No process is involved in more than a single send and a single receive at one time.
- Each process stores close in memory those copies of non-local site variables which are local to the same process. In this way synchronization does not require the use of additional buffers receiving buffers.

Note that two processes may be overlapping (i.e., store neighbor sites) with respect to one lattice and not overlapping in respect to a different lattice within the same program. It is possible, in principle, to change the above communication patterns to optimize communication for various network topologies.

Although communication is currently based on MPI, they do not make use of communication tags. It is therefore possible, in principle, to speed-up communication by using a faster tagless and bufferless protocol such as Myricom GM.

Our communication patterns have the effect of making communication almost insensitive to network latency, and communication speed is dominated by network bandwidth. Benchmarks are very much application dependent since parallel efficiency is greatly affected by the lattice size, by the amount of computation performed per site, processor speed and type of interconnection. In many typical applications, like the one described in the preceding example, the drop in efficiency is less than 10% up to 8 nodes (processes) and less than 20% up to 32 (our tests are usually performed on a cluster of Pentium 4 PCs (2.2GHz) running Linux and connected by Myrinet).

## 5 MDP and PSIM

For portability reasons MDP is based on MPI. Nevertheless it is desirable to be able to run, test and debug MDP programs on a single node (with single or multi processor architecture) without having to install MPI. The latest version of MDP includes a Parallel SIMulator (PSIM). Despite the name this is not quite a simulator but an emulator, i.e. a message passing library that uses local (unix posix) socket pairs. PSIM is Objected Oriented and is not based on MPI.

When compiling with PSIM, the parallel processes are created at start-up by forking. The number of parallel processes is specified at runtime by passing the following command line argument to any MDP executable program

`-PSIM_NPROCS=4`

(this makes 4 parallel processes).

PSIM also creates a communication log that can be used for debugging. MDP with PSIM has been tested on Linux, Mac and Windows (with cygwin).

For single processor node, using PSIM does not introduce any speed-up but, for a small number of processes (2-16) it does not slow down the code either. For multi-process shared memory architecture, parallelization of PSIM should produce a speed-up comparable with MPI. We have been yet performed such tests.

Moreover, PSIM should perform well on openMosix clusters as soon as open-Mosix starts supporting migratable sockets since all communication between the parallel processes will be done by the operating system. Unfortunately open-Mosix does not support migratable sockets yet.

## 6 Example: Ising Model

As one more example of usage of MDP we report here a simple program for the Ising model.

# Computational Finance and its Applications

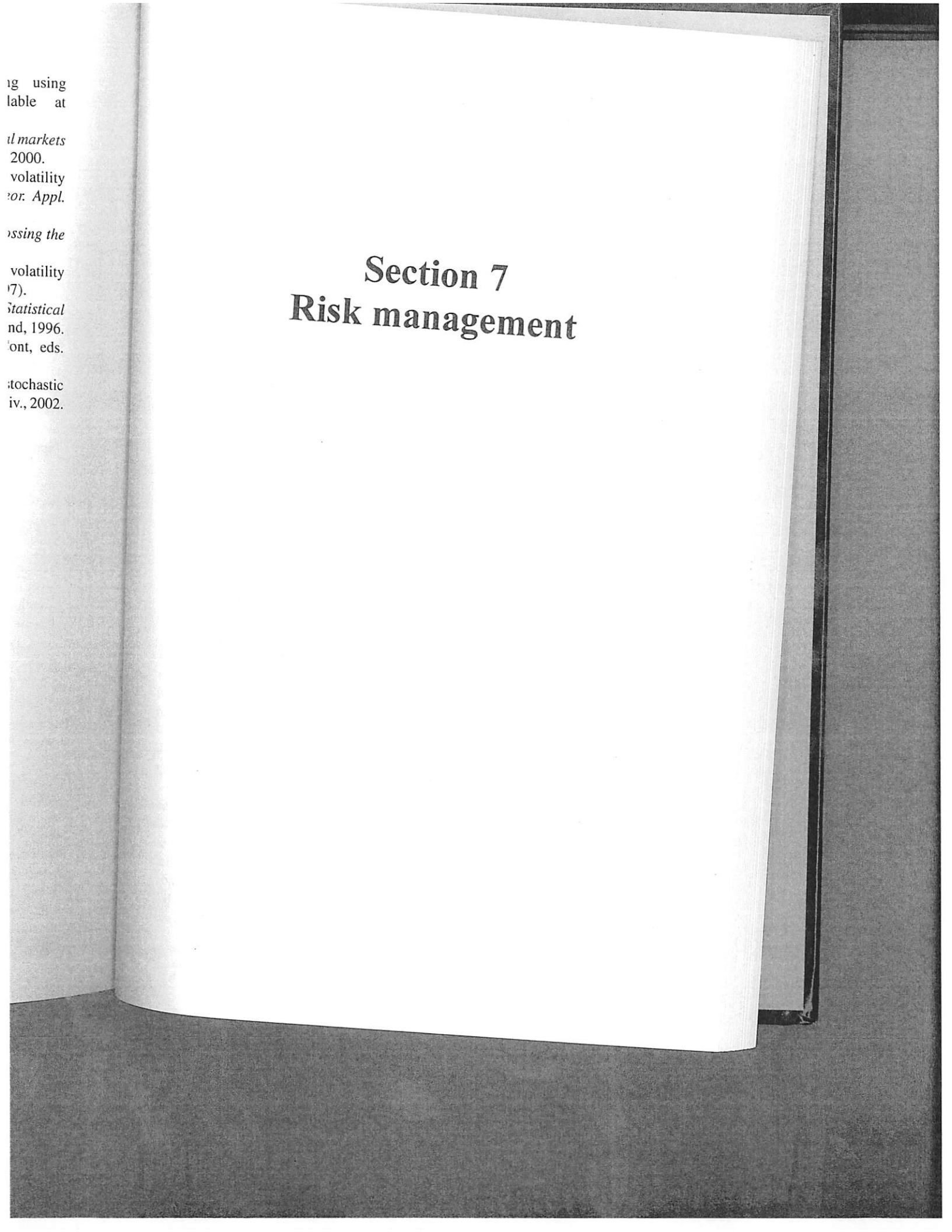
II

EDITORS

M. Costantino & C.A. Brebbia



WIT PRESS



## Section 7

# Risk management

## Monte Carlo risk management

M. Di Pierro & A. Nandy

*CTI, DePaul University, Chicago, IL, USA*

### Abstract

In this paper we propose a Monte Carlo based approach to Risk Management. Our approach applies to any system subject to random uncorrelated losses under very general conditions. Our methodology consists of the following steps: model the distribution of losses and the distribution of time intervals between losses via an analysis of historical data; perform a Monte Carlo simulation of a finite period of time in the future; use the Monte Carlo data to estimate the 99.9% VaR.

### 1 Introduction

In this paper we present an approach to Risk Management based on the Monte Carlo technique. The proposed approach is very general and it can be applied to any system characterized by discrete losses. We have made the following broad assumptions: a) loss events are independent; b) number of loss events occurring in any time interval  $\Delta T$  is independent of loss events occurring before the time interval considered; c) the probability of two events occurring at exactly same time is zero.

Discrete losses due to internal fraud (i.e. Operational Risk as defined by Basel II accord) for a particular department in a Bank provide a good example of application [1].

Our approach is based on the following steps: 1) model the time distribution and the severity distribution of losses; 2) simulate possible future scenarios compatible with the observed time and severity distributions; 3) compute the 99.9% VaR as the monetary amount that is greater than the total loss in 99.9% of the simulated scenarios.

Notice that our approach is free from bias or assumptions about the distribution of the total loss. The validity of the assumptions a), b) and c) can be verified directly from the data, as we show in the example of section 3.



In the next section of this paper we present the general approach and provide examples of distribution functions that can be used to model the time distribution and the severity distribution of losses.

In the third section we provide an example of application. We build a hypothetical portfolio of stocks and compute the 99.9% VaR due exclusively to mini-crashes. In the example we define a mini-crash as the event when all stocks considered move simultaneously downward. We stress again that our method is very general and can be applied seamlessly to any portfolio for any quantifiable definition of crashes [2] (as long as they can be identified as discrete events), as well as to the analysis of other types of losses (for example Operational Risk losses).

We want to stress that in the cases examined here, usual analytical formulas for computing the VaR without performing a simulation does not apply because the distribution of the total loss over a finite period of time is not Gaussian.

## 2 Model

Our approach is based on the following assumptions:

- a) loss events are independent.
- b) the number of loss events occurring in any time interval  $\Delta T$  is independent of the number of loss events occurring before the time interval considered;
- c) the probability of two events occurring at exactly same time is zero.

The consequence of assumptions a), b) and c) is that we can separately model the frequency distribution of losses and the severity distribution of losses. More specifically, assumptions b) and c) indicate that the process underlying the frequency distribution of losses is a Poisson process.

With the above assumptions, our approach requires the following steps that are described in detail in the following subsections:

1. Model the time distribution and severity distribution of losses separately;
2. Simulate a large set of possible future scenarios compatible with the observed time and severity distributions;
3. Compute the 99.9% VaR as the monetary amount that is greater than the total loss in 99.9% of the simulated scenarios.

### 2.1 Modeling time and severity distributions

Each loss event is characterized by the time when the event occurs,  $T_i$ , and by the severity of the loss,  $S_i$ . In the example below we assume  $T_i$  is measured in days and  $S_i$  is measured in dollars. We also assume that losses are sorted by time so that  $T_i < T_{i+1}$ . We model the time distribution of  $\{T_i\}$  and  $\{S_i\}$  separately.

As consequence of assumptions a), b) and c), the process underlying the frequency distribution of losses is a Poisson process. This means that the number of events occurring in any finite time frame  $T$  follows the Poisson distribution. It



also means that the distribution of the time intervals between subsequent losses,  $t_i = T_{i+1} - T_i$  is exponential

$$p(t) = \lambda e^{-\lambda t} \quad (1)$$

where  $\lambda$  is one of the parameter of our model and it equals the average number of events per unit of time (per day). It can be measured from the data as

$$\lambda = (E[t])^{-1} = \left( \frac{1}{N-1} \sum_{i=0}^{i < N-1} (T_{i+1} - T_i) \right)^{-1} \quad (2)$$

where  $N$  is the number of historical loss events.

Modeling the severity distribution is a more difficult task and it depends on the data set at hand. Some general characteristics are that losses are always positive and can be arbitrarily large or small, therefore we expect the underlying distribution to be appreciatively log-normal. In the most general case, consistently with extreme value theory [3–5], we expect the presence of a fat tail for high losses, as observed in [1, 6]. A fat tail is characterized by an inverse polynomial behavior as opposed to an exponential behavior as for a normal or lognormal distribution. The natural choice for modeling this fat tail is therefore a generalized Pareto distribution (GPD).

Our approach consists of modeling the severity distribution using the following distribution function

$$p(x) = a\theta(x - x_0) \frac{1}{x} e^{-\frac{(\log x - \mu)^2}{2\sigma^2}} + ab\theta(x_0 - x)(x + c)^{-\beta} \quad (3)$$

which is discussed in detail in Appendix A. This distribution is exactly log-normal for  $x < x_0$ , exactly GPD for  $x > x_0$ , and it is continuous and differentiable everywhere. It reduces to a log-normal for  $x_0 \rightarrow \infty$  and to a GPD for  $x_0 = 0$ . It depends on only four parameters:  $\mu$ ,  $\sigma$  (mean and variance of the log-normal contribution),  $x_0$  (the point where change of behavior is observed),  $\beta$  (the power of the polynomial behavior of the fat tail).  $a$ ,  $b$ , and  $c$  are constants that depend on the above four parameters and their explicit analytical expressions are reported in Appendix A.

Fig. (1) shows various plots of  $p(x)$  (left) and the corresponding cumulative distribution functions  $F(x)$  (right) in logarithmic scale, for different values of  $x_0$ , and fixed values of  $\mu$ ,  $\sigma$ , and  $\beta$ . The greater the value of  $x_0$ , the smaller the fat tail.

In summary, the parameters of our model are  $\lambda$ ,  $\mu$ ,  $\sigma$ ,  $x_0$ , and  $\beta$ . While  $\lambda$  is measured via eq. (2), the other 4 parameters can be measure via a fit of the cumulative distribution function implied by historical severity data.

## 2.2 Simulation

Once the model parameters are extracted from historical data, we proceed by performing a simulation of multiple future scenarios. Each simulated scenario



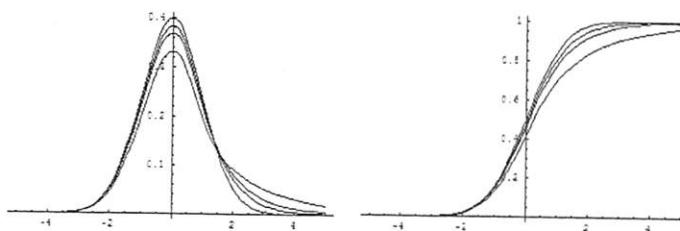


Figure 1: Log-plot of  $p(x)$  (left) and  $F(x) = \int_{-\infty}^x p(x)dx$  (right) for different values of  $\beta = 1.5, 2, 2.5$  at  $\mu = 0$ ,  $\sigma = 1$ , and  $z_0 = 1$  fixed, compared with a Gaussian distribution.

covers a finite time interval from 0 (today) until some future time  $T$ . This is the time over which we wish to compute the Value-at-Risk. Each simulated scenario is characterized by a set of simulated loss events occurring within the time interval  $T$ .

The simulation of each simulated scenario starts at time 0 and recursively proceeds to simulate the next loss event. The next loss event will occur in time  $t$  and will have a severity  $s$ .  $t$  and  $s$  are random numbers generated according to the frequency distribution and the severity distribution respectively. The simulation of each scenario proceeds until a loss event occurs beyond the time interval  $T$  being considered.

Multiple scenarios are simulated using the method described above. Then, for each simulated scenario  $k$ , we compute the total loss  $L_k$  by summing the discounted  $s$  losses occurring under scenario  $k$  within the considered time interval.

### 2.3 Computing the VaR

The 99.9% Value-at-Risk is defined as the value such that the probability of losing more than its value is 0.1%. Since our simulated scenarios are generated using the same procedure, they all occur with equal probability  $1/m$ , where  $m$  is the number of simulated scenarios. The 99.9% VaR can therefore be determined as that value that is greater than the total loss  $L_k$  in 99.9% of the simulated scenarios, and less the total loss in the remaining 0.1% of the simulated scenarios. This computation is done by sorting the scenarios according to their total loss  $L_k$  so that  $L_{k+1} > L_k$  and choosing (the notation  $\lceil x \rceil$  indicates excess rounding of  $x$ )

$$99.9\% \text{ VaR} = L_{\lceil 0.999m \rceil}$$

The same argument applies to the computation of the VaR for any other percentile. The larger is the number of simulated scenarios  $m$ , the more precise is the determination of the VaR.

Notice how, in general, the usual formula to compute the VaR in terms of the standard deviation of the losses does not apply. In particular that formula does not apply if the distribution of the total loss is not Gaussian. Fig. (2) shows that this is clearly not the case for the systems under consideration in this paper.



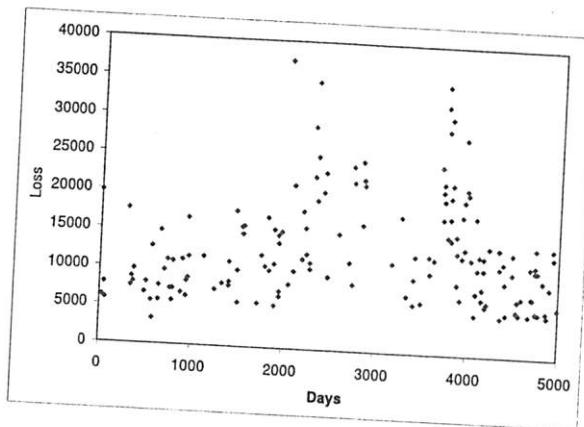


Figure 2: Loss events associated to the mini-crashes (all stocks moving downward) of our sample portfolio during the time frame March 16, 2001 - February 21, 2006.

### 3 Example

In this section, as an example application of our approach, we consider the computation of VaR due to mini-crashes in a portfolio of stocks. This number represents the economic capital that one would have to save to provide insurance against such events at 99.9% confidence level. We define a mini-crash as the event when all the stocks considered move simultaneously downward.

In order to provide a concrete case we consider a portfolio comprised of the following stocks:

MSFT, RTN, HCA, EP, AVP, HNZ, USB, GD

with a constant capital of \$100,000 invested in each of them.

The above set of stocks have been chosen because of their relative low correlation. If the stocks were uncorrected, the probability of all of them moving downward (a mini-crash) in one day would be about 0.4%. Our analysis of historical market data indicates that this occurred 159 times in the time period starting March 16, 1992 and ending February 21, 2006, which averages to approximately once every 31 calendar days, or once every 21 trading days. This corresponds to a mini-crash probability of 5%, ten times more likely than the naive expectation.

Fig. 2 shows past losses associated with our portfolio in the time frame considered.

Fig. 3 shows the distribution of time intervals between two consecutive past losses, superimposed with the exponential distribution using the value  $\lambda = 1/31$  extracted from the data. The plot indicates that the probability of two consecutive mini-crashes occurring in two consecutive days is more likely than predicted by the exponential distribution. This is a small deviation from the assumption and



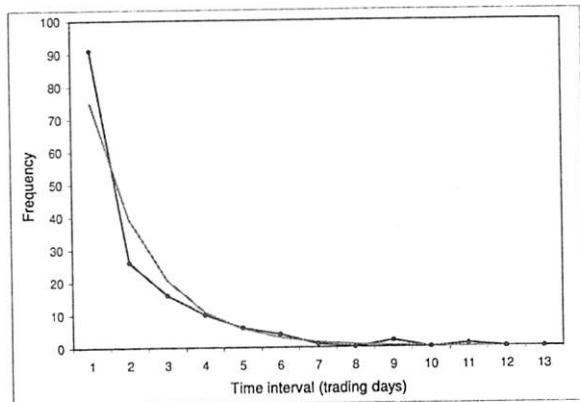


Figure 3: Distribution of time intervals between two consecutive past losses, superimposed to the exponential distribution at  $\lambda = 1/31$ .

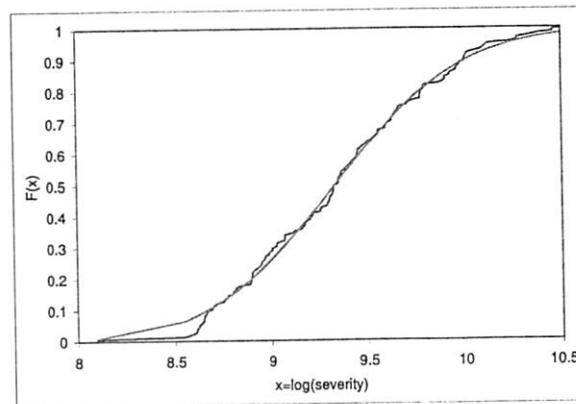


Figure 4: Log-plot of the cumulative distribution function  $F(x)$  for the severity of historical losses, and best fit using the distribution in eq. (3).

therefore it will be ignored in the rest of the paper, although it could be taken into account by correcting the frequency distribution.

Fig. 4 shows, in logarithmic scale, the cumulative distribution function for the severity of historical losses and its best fit using the distribution in eq. (3).

Fig. 5 shows the total loss for each simulated scenario for  $m = 10000$  simulations. The simulated scenarios are sorted according to their total loss. The 99.9% VaR reads directly from this plot as the  $y$ -axis value corresponding to the  $x$ -axis value  $[0.999m] = 9990$ .

Other VaR percentiles can be calculated in a similar fashion.

Our result is 99.9% VaR = \$360,000. This is 45% of the total funds invested in the portfolio. (The number exceeds the regular 99.9% VaR associated with the portfolio because it only consider events when all stocks jump downward. In fact,



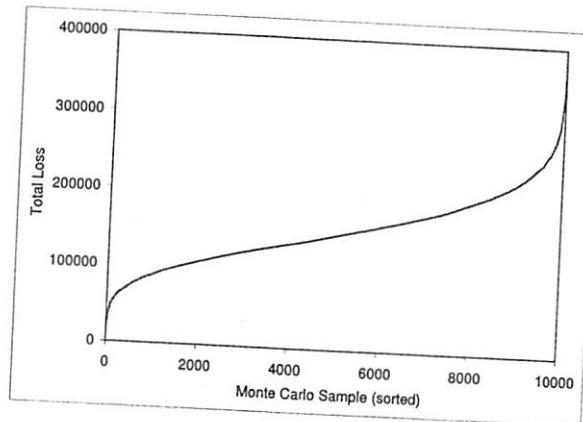


Figure 5: Total loss of each of the 10000 simulated scenarios. The 99.9% VaR reads directly from this plot as the  $y$ -axis value corresponding to the  $x$ -axis value  $[0.999m] = 9990$ .

in the example considered, positive jumps offset most of the risk discussed in this paper. Our approach allows us to isolate the risk associate to this specific type of discrete event.)

#### 4 Conclusion

In this paper we present a Monte Carlo approach to the computation of Value-at-Risk for systems subject to discrete losses. The main ingredients of our approach are that it model the frequency distribution of losses as a Poisson process and the severity distribution of losses using the novel fat-tail distribution discussed in this paper. The VaR is computed by performing a Monte Carlo simulation of future losses and sorting these scenarios according to their relative total loss. This procedure is very general and does not require any assumption about the distribution of the total loss.

In this paper, as an example, we have applied our technique to the computation of the VaR due to mini-crashes of a given stock portfolio.

#### Acknowledgements

We thank MetaCryption LLC for providing access to the financial data and access to the Monte Carlo Engine [7] used for our simulation.

#### Appendix A: Combined Log-normal + Pareto distribution

The particular distribution utilized in this paper to model the severity of losses has the following form

$$p(x) = a\theta(x - x_0) \frac{1}{x} e^{-\frac{(\log x - \mu)^2}{2\sigma^2}} + ab\theta(x_0 - x)(x + c)^{-\beta} \quad (4)$$

This distribution is exactly log-normal for  $x < x_0$ , exactly Pareto for  $x > x_0$ , and it is continuous and differentiable everywhere, including  $x = x_0$ .

$\mu, \sigma, x_0$  and  $\beta$  are free parameters, while  $a, b$  and  $c$  are coefficients that depend on those parameters via the following exact relations:

$$a = \left( \sqrt{\frac{\pi}{2}} \sigma \left( 1 + \operatorname{erf}\left(\frac{\log x_0 - \mu}{\sqrt{2}\sigma}\right) \right) + \frac{b}{\beta - 1} (x_0 + c)^{1-\beta} \right)^{-1} \quad (5)$$

$$b = \frac{(x_0 + c)^\beta}{x_0} e^{-\frac{(\log x_0 - \mu)^2}{2\sigma^2}} \quad (6)$$

$$c = \frac{x_0 \beta \sigma^2}{\sigma^2 + \log x_0 - \mu} - x_0 \quad (7)$$

where we defined

$$\operatorname{erf}z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \quad (8)$$

## Appendix B: Statistics about the sample portfolio

Average daily log-return and daily volatility in the time period starting March 16, 1992 and ending February 21, 2006.

	MSFT	RTN	HCA	EP	AVP	HNZ	USB	GD
$\mu/100$	7.2%	1.6%	3.8%	3.1%	6.3%	3.6%	5.6%	11.4%
$\sigma$	2.2%	2.1%	2.2%	2.0%	2.0%	1.4%	1.9%	1.8%

Average correlation of daily log-returns in the same period.

	MSFT	RTN	HCA	EP	AVP	HNZ	USB	GD
MSFT	-	8%	13%	13%	10%	12%	24%	16%
RTN	8%	-	13%	10%	14%	13%	17%	29%
HCA	13%	13%	-	9%	17%	18%	21%	14%
EP	13%	10%	9%	-	7%	12%	15%	12%
AVP	10%	14%	17%	7%	-	24%	20%	14%
HNZ	12%	13%	18%	12%	24%	-	22%	14%
USB	24%	17%	21%	15%	20%	22%	-	17%
GD	16%	29%	14%	12%	14%	14%	17%	-

## References

- [1] M. Di Pierro and A. Nandy, "Comprehensive Modeling of Operational Risk Data An Empirical Approach", (publication-pending)
- [2] M Rubinstein, "Portfolio Insurance and the Market Crash", Financial Analysts Journal, (1988)
- [3] J. Beirlant, J. Teugels, P. Vynckier, P. "Practical analysis of extreme values", Leuven University Press, Leuven (1996)
- [4] P. Embrechts, C. Klüppelberg, T. Mikosch, "Modeling extremal events for insurance and Finance", Springer, Berlin (1997)
- [5] A. Roehr, "Modelling Operational Losses", Algo Research Quarterly, Vol. 5, n.2 (2002)
- [6] de Fontnouvelle, Patrick, De Jesus-Rueff, Virginia, Jordan, John S. and Rosengren, Eric S., "Using Loss Data to Quantify Operational Risk" (April 2003). <http://ssrn.com/abstract=395083>
- [7] OpRisk Calculator, [www.metacryption.com](http://www.metacryption.com) (a software tool developed by MetaCryption LLC that implements the methodology described in this paper).



## PREDICTIVE LATTICE QCD

ANDREAS S. KRONFELD<sup>\*,a</sup>,

I.F. ALLISON<sup>b</sup>, C. AUBIN<sup>c,d</sup>, C. BERNARD<sup>d</sup>, C.T.H. DAVIES<sup>b</sup>, C. DETAR<sup>e</sup>,  
M. DI PIERRO<sup>f</sup>, E.D. FREELAND<sup>g</sup>, STEVEN GOTTLIEB<sup>h</sup>, A. GRAY<sup>i</sup>, E. GREGORY<sup>j</sup>,  
U.M. HELLER<sup>k</sup>, J.E. HETRICK<sup>l</sup>, A.X. EL-KHADRA<sup>m</sup>, L. LEVKOVA<sup>h</sup>, P.B. MACKENZIE<sup>a</sup>,  
F. MARESCA<sup>e</sup>, D. MENSCHER<sup>m</sup>, M. NOBES<sup>n,o</sup>, M. OKAMOTO<sup>a</sup>,  
M.B. OKTAY<sup>m</sup>, J. OSBORNE<sup>e</sup>, D. RENNER<sup>j</sup>, J.N. SIMONE<sup>a</sup>,  
R. SUGAR<sup>p</sup>, D. TOUSSAINT<sup>j</sup>, H.D. Trottier<sup>o</sup>

<sup>a</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

<sup>b</sup>*Department of Physics and Astronomy, Glasgow University, Glasgow, Scotland G12 8QQ, UK*

<sup>c</sup>*Physics Department, Columbia University, New York, New York 10027, USA*

<sup>d</sup>*Department of Physics, Washington University, St. Louis, Missouri 63130, USA*

<sup>e</sup>*Physics Department, University of Utah, Salt Lake City, Utah 84112, USA*

<sup>f</sup>*School of Computer Science, Telecommunications and Information Systems,  
DePaul University, Chicago, Illinois 60604, USA*

<sup>g</sup>*Liberal Arts Department, School of the Art Institute, Chicago, Illinois 60603, USA*

<sup>h</sup>*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

<sup>i</sup>*Department of Physics, The Ohio State University, Columbus, Ohio 43210, USA*

<sup>j</sup>*Department of Physics, University of Arizona, Tucson, Arizona 85721, USA*

<sup>k</sup>*American Physical Society, Ridge, New York 11961, USA*

<sup>l</sup>*Physics Department, University of the Pacific, Stockton, California 95211, USA*

<sup>m</sup>*Physics Department, University of Illinois, Urbana, Illinois 61801, USA*

<sup>n</sup>*Laboratory of Elementary-Particle Physics, Cornell University, Ithaca, New York 14853, USA*

<sup>o</sup>*Physics Department, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada*

<sup>p</sup>*Department of Physics, University of California, Santa Barbara, California 93106, USA*

Fermilab Lattice, MILC, and HPQCD Collaborations

In the past year, we calculated with lattice QCD three quantities that were unknown or poorly known. They are the  $q^2$  dependence of the form factor in semileptonic  $D \rightarrow Kl\nu$  decay, the decay constant of the  $D$  meson, and the mass of the  $B_c$  meson. In this talk, we summarize these calculations, with emphasis on their (subsequent) confirmation by experiments.

*Keywords:* quantum chromodynamics; lattice QCD;  $D$  meson decay;  $B_c$  meson mass

### 1. INTRODUCTION AND BACKGROUND

In recent years, lattice QCD has reached the stage where many calculations of hadron masses, mass splittings, and operator matrix elements agree with experi-

\*Speaker at the conference. E-mail: ask@fnal.gov

mental measurements. The key has been the inclusion of sea quarks. The progress has been especially striking<sup>1</sup> when the light quarks (sea and valence) are implemented as staggered quarks, with an improved action.

Some of the ingredients of these calculations are controversial. Staggered quarks come in four tastes, three of which must be removed to obtain each individual flavor. For sea quarks, this is done by taking the fourth root of the fermion determinant; for valence quarks, by projecting onto the desired taste sector. Furthermore chiral perturbation theory must be modified<sup>2</sup>. Although evidence for the validity of these “tricks” is slowly accumulating, a proof remains at large<sup>3</sup>. In addition, much of the success of Ref.<sup>1</sup> comes from hadrons with heavy quarks. Although debate on heavy quarks in lattice QCD seems to have subsided, checks are still useful.

In this paper, we discuss three calculations, with emphasis on their subsequent experimental confirmation. They are the normalization and  $q^2$ -dependence of the  $D \rightarrow K\ell\nu$  form factor; the decay constants of the  $D^+$  and  $D_s$  mesons; and the mass of the  $B_c$  meson. Each tests a somewhat different combination of the ingredients, and the following table gives an informal guide:

calculation	light sea	light valence	heavy
semileptonic $f_+(q^2)$	**	**	**
leptonic $f_D$	**	***	**
$B_c$ mass	**	—	***

The chiral extrapolation, which is more sensitive to valence quarks than sea quarks, turned out to be more important for the decay constant than the form factor. The  $B_c$  meson has no light valence quarks at all, but one should expect an accurate calculation only if heavy-quark discretization effects are under control.

Successful predictions are, of course, not a substitute for a proof. They are still useful. Even if the experts are confident of all the elements of their numerical calculations, non-experts are interested in an end-to-end check<sup>4</sup>. The quantities discussed here are ideal candidates: they are straightforward to compute; the first “good” experimental measurements were not expected until this year; and new physics is unlikely to contribute significantly.

## 2. SEMILEPTONIC $D$ DECAYS

Semileptonic decays such as  $D \rightarrow K\ell\nu$  are mediated by electroweak vector currents. The matrix element  $\langle K|V^\mu|D\rangle$  is parametrized by form factors. For a vector current there are two, but experimentally only the one called  $f_+(q^2)$  is accessible; the rate from the other one,  $f_0(q^2)$ , is suppressed by  $m_l^2$ . Here  $q^2$  is the momentum transferred to the lepton-neutrino system, falling in the range  $0 \leq q^2 \leq q_{\max}^2 = (m_D - m_K)^2$ . In lattice QCD, discretization effects are smallest when the momentum  $\mathbf{p}$  of the kaon is small, and then  $q^2$  is not too far from  $q_{\max}^2$ .

Experiments usually measure the branching fraction and quote the normalization  $f_+(0)$ , after making assumptions about the  $q^2$  dependence. While our re-

sults were still preliminary<sup>5</sup>, experimental results came out for the normalization of  $D \rightarrow Kl\nu$ <sup>6</sup> and  $D \rightarrow \pi l\nu$ <sup>7</sup>. The agreement with our final results<sup>8</sup> is excellent. For example, we find  $f_+^{D \rightarrow K}(0) = 0.73(3)(7)$ <sup>8</sup> while BES measures  $f_+^{D \rightarrow K}(0) = 0.78(5)$ <sup>6</sup>. Our calculations of the normalization are also consistent with the soft pion theorem, which states  $f_0(q_{\max}^2) = f_D/f_\pi$ .

In principle, the shape of the form factors can be computed directly in lattice QCD. In practice, we calculated at a few values of  $\mathbf{p}$  and used the Bećirević-Kaidalov (BK) form<sup>9</sup> to fix the full  $q^2$  dependence of  $f_+$  and  $f_0$ . Then the normalization of  $f_+$  comes mainly from  $f_0$  through a kinematic constraint  $f_+(0) = f_0(0)$ . The BK Ansatz and calculations near  $q_{\max}^2$  determine the shape. It was important, therefore, to measure the  $q^2$  dependence experimentally. In photoproduction of charm off fixed nuclear targets, the FOCUS Collaboration was able to collect high enough statistics to trace out the  $q^2$  distribution of the decay<sup>10</sup>. This setup does not yield an absolutely normalized branching ratio, so one is left to compare  $f_+(q^2)/f_+(0)$ .

In Fig. 1 we plot our result for  $f_+(q^2)/f_+(0)$  vs.  $q^2/m_{D_s^*}^2$ . The errors from  $f_+(0)$  must be propagated to non-zero  $q^2$ , so for  $f_+(q^2)/f_+(0)$  the errors grow with  $q^2$ . Figure 1 shows 1- $\sigma$  bands of statistical (orange) and all uncertainties (yellow) added in quadrature<sup>11</sup>. As one can see, the  $q^2$  dependence of lattice QCD (curve and error band) and experiment (points) agree excellently, although the uncertainties are still several per cent.

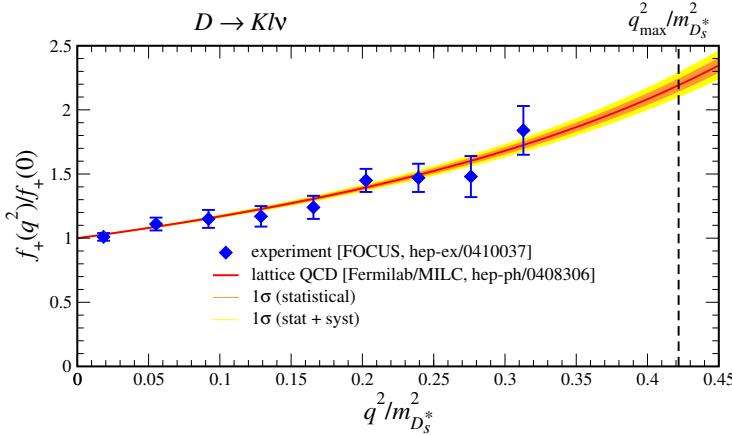


Fig. 1. Shape of form factor  $f_+(q^2)/f_+(0)$  vs.  $q^2/m_{D_s^*}^2$ , compared with experiment<sup>10</sup>.

### 3. LEPTONIC $D$ DECAYS

We also computed the hadronic matrix element for the leptonic decay of charmed mesons,  $f_{D^+}$  and  $f_{D_s}$ . The first (experimental) measurements of  $f_{D^+}$  appeared in 2004, with three events from BES<sup>12</sup> and eight from CLEO<sup>13</sup>. Neither provides a stringent test of QCD, but CLEO-*c* was just starting its run and promised 5–8 times higher statistics by the Summer 2005 Lepton-Photon Symposium<sup>4</sup>. At Lattice

2004<sup>14</sup>, we presented preliminary results for  $f_{D+}$ , based on one lattice spacing,  $a \approx 0.125$  fm. Our aim was to extend the running to two other lattice spacings and, of course, to improve our understanding of other aspects of the calculation, such as the chiral extrapolation. Details are given in the ensuing publication<sup>15</sup>. We find

$$f_{D+} = 201 \pm 3 \pm 17 \text{ MeV}, \quad (1)$$

where the first error is from finite Monte Carlo statistics, the second is a sum in quadrature of several systematics. A conservative (but not naive) estimate of heavy-quark discretizations effects, as discussed in Ref.<sup>16</sup>, is the second largest (largest) systematic on  $f_{D+}$  ( $f_{D_s}$ ). A few days after our paper was posted on the arXiv, CLEO-*c* announced its new measurement<sup>17</sup>

$$f_{D+} = 223 \pm 17 \pm 3 \text{ MeV}, \quad (2)$$

based on  $47 \pm 8$  events. At the  $1-\sigma$  level, the agreement between Eqs. (1) and (2) is fine. One should keep in mind that the experiment actually determines  $|V_{cd}| f_{D+}$ . CLEO-*c*<sup>17</sup> assumes that  $|V_{cd}| = |V_{us}|$  and uses a recent average of  $|V_{us}|$  from semileptonic  $K$  decay.

It is interesting to look at the  $n_f$  dependence of  $f_{D_s}$ , shown in Fig. 2(a). Of course, quenched results vary widely, but we show one<sup>18</sup> carried out with similar choices for heavy quarks, renormalization factors, etc. One sees a trend of  $f_{D_s}$  to increase with  $n_f$ . A similar comparison of  $f_{D+}$ , in Fig. 2(b), is less instructive, because the chiral extrapolations in Refs.<sup>18,19</sup> started at large quark masses and are, hence, less reliable than in the present work.

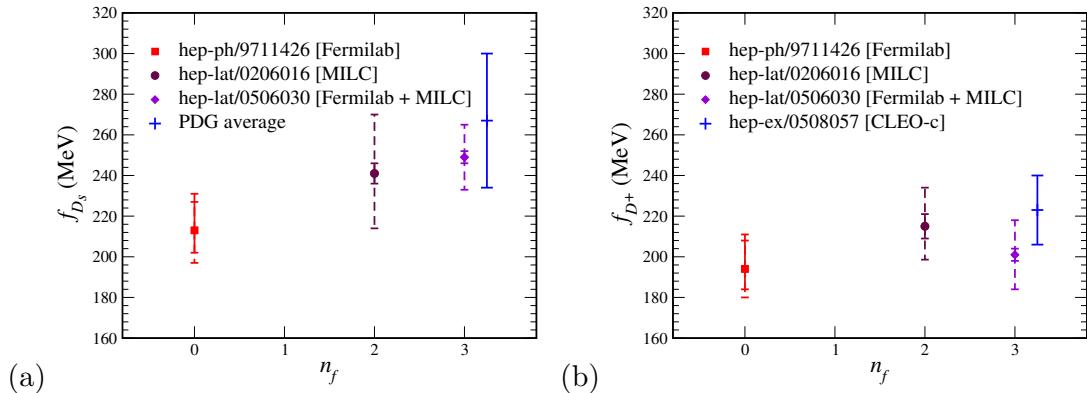


Fig. 2. Dependence of (a)  $f_{D_s}$  and (b)  $f_{D+}$  on the number  $n_f$  of sea flavors. Quenched ( $n_f = 0$ )<sup>18</sup>;  $n_f = 2$ <sup>19</sup>;  $n_f = 3$ <sup>15</sup>. Solid (dashed) error bars are statistical (statistical+systematic).

#### 4. MASS OF THE $B_c$ MESON

The pseudoscalar  $B_c^+$  meson is the lowest-lying bound state of a charmed quark and a  $b$  quark. CDF<sup>20</sup> first observed it during Run I of the Tevatron in the semileptonic decay  $B_c^+ \rightarrow J/\psi l^+ \nu$ . During Run II, DØ has confirmed the discovery in the same

mode<sup>21</sup>. Because the neutrino is undetected, the mass resolution in semileptonic modes is poor,  $\pm(300\text{--}400)$  MeV. Now, however, the upgraded detectors are able to reconstruct hadronic modes, such as  $B_c^+ \rightarrow J/\psi\pi^+$ , which give much much better precision on  $m_{B_c}$ <sup>22</sup>.

At Lattice 2004 we presented results in nearly final form<sup>23</sup>, and posted the final results on the arXiv in mid-November<sup>24</sup>:

$$m_{B_c} = 6304 \pm 12^{+18}_0 \text{ MeV}, \quad (3)$$

where the last error is a rough estimate of residual heavy-quark discretization effects. Soon afterwards, CDF announced a precise mass measurement. They find<sup>25</sup>

$$m_{B_c} = 6287 \pm 5 \text{ MeV}, \quad (4)$$

which agrees with Eq. (3) at slightly more than  $1-\sigma$ .

Two comments are in order. First, the agreement at the gross level of the calculation with experiment shows that discretization effects are well under control with lattice NRQCD<sup>27</sup> and the Fermilab method<sup>28</sup>. Of course, this follows from the careful application of effective field theories for heavy quarks<sup>29,30</sup>. Indeed, as seen in Fig. 3(a), almost no lattice spacing dependence is seen in the splitting  $\Delta_{\psi\Upsilon} = m_{B_c} - (\bar{m}_\psi + m_\Upsilon)/2$  that is at the crux of the calculation<sup>26</sup>. Moreover, it is striking how much the splitting  $\Delta_{\psi\Upsilon}$  changes when sea quarks are included. Figure 3(b) compares Eq. (3) with an old quenched calculation<sup>26</sup> (and the measurement<sup>25</sup>). The solid error bar shows the non-quenching errors, and the dashed includes the estimate of the quenching error. The inclusion of sea quarks has reduced the splitting by a factor of three or four, bringing an essentially discrepant result into agreement.

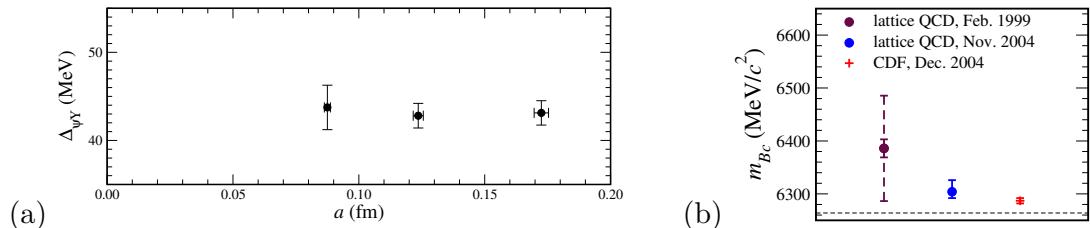


Fig. 3. (a) Dependence of the splitting  $\Delta_{\psi\Upsilon}$  on the lattice spacing  $a$ . (b) Comparison of the quenched<sup>26</sup>,  $n_f = 2 + 1$ <sup>24</sup>, and experimental<sup>25</sup> values of  $m_{B_c}$ ; the dashed line denotes the baseline  $(\bar{m}_\psi + m_\Upsilon)/2$ .

## 5. CONCLUSIONS

In the past year, three lattice-QCD calculations have been confirmed by experiment. FOCUS<sup>10</sup> confirmed the  $q^2$ -dependence of the  $D \rightarrow Kl\nu$  form factor<sup>8</sup>; CLEO-*c*<sup>17</sup> confirmed the *D*-meson decay constant<sup>15</sup>; and CDF<sup>25</sup> confirmed the mass of the

$B_c$  meson<sup>24</sup>. To obtain these results it is essential to have heavy-quark discretization effects under control, as one expects from theoretical foundations<sup>27,28,29,30</sup>. Furthermore, the comparison of quenched QCD, QCD with 2+1 staggered flavors, and experiment shows that sea quarks are needed to obtain agreement, and that staggered quarks (in these cases) capture the needed effect.

### Acknowledgments

This work has been supported in part by the U.S. National Science Foundation, the Office of Science of the U.S. Department of Energy (DOE), and the U.K. Particle Physics and Astronomy Research Council. Fermilab is operated by Universities Research Association Inc., under contract with the DOE. The speaker (A.S.K.) thanks Prof. Chuan Liu and his colleagues and students for hospitality and assistance during the conference.

### References

1. C. T. H. Davies *et al.* [HPQCD, MILC, and Fermilab Lattice Collaborations], Phys. Rev. Lett. **92**, 022001 (2004) [hep-lat/0304004].
2. C. Aubin and C. Bernard, Phys. Rev. D **68**, 034014 (2003) [hep-lat/0304014]; *ibid.*, 074011 (2003) [hep-lat/0306026].
3. S. Dürr, “Theoretical issues with staggered fermion simulations,” hep-lat/0509026.
4. I. Shipsey, Nucl. Phys. Proc. Suppl. **140**, 58 (2005) [hep-lat/0411009].
5. M. Okamoto *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Nucl. Phys. Proc. Suppl. **129**, 334 (2004) [hep-lat/0309107].
6. M. Ablikim *et al.* [BES Collaboration], Phys. Lett. B **597**, 39 (2004) [hep-ex/0406028].
7. G. S. Huang *et al.* [CLEO Collaboration], Phys. Rev. Lett. **94**, 011802 (2005) [hep-ex/0407035].
8. C. Aubin *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Phys. Rev. Lett. **94**, 011601 (2005) [hep-ph/0408306].
9. D. Bećirević and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000) [hep-ph/9904490].
10. J. M. Link *et al.* [FOCUS Collaboration], Phys. Lett. B **607**, 233 (2005) [hep-ex/0410037].
11. C. Aubin *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], work in progress.
12. M. Ablikim *et al.* [BES Collaboration], Phys. Lett. B **610**, 183 (2005) [hep-ex/0410050].
13. G. Bonvicini *et al.* [CLEO Collaboration], Phys. Rev. D **70**, 112004 (2004) [hep-ex/0411050].
14. J. N. Simone *et al.* [Fermilab Lattice, MILC, and HPQCD Collaborations], Nucl. Phys. Proc. Suppl. **140**, 443 (2005) [hep-lat/0410030].
15. C. Aubin *et al.*, [Fermilab Lattice, MILC, and HPQCD Collaborations], Phys. Rev. Lett. **95**, 122002 (2005) [hep-lat/0506030].
16. A. S. Kronfeld, Nucl. Phys. Proc. Suppl. **129**, 46 (2004) [hep-lat/0310063].
17. M. Artuso, talk at the XXII International Symposium on Lepton-Photon Interactions at High Energy, Uppsala, Sweden, <http://1p2005.ts1.uu.se/~1p2005/> (June 30–July 5, 2005);  
M. Artuso *et al.* [CLEO Collaboration], “Improved measurement of  $B(D^+ \rightarrow \mu^+\nu)$  and the pseudoscalar decay constant  $f_{D^+}$ ,” hep-ex/0508057.

18. A. X. El-Khadra, A. S. Kronfeld, P. B. Mackenzie, S. M. Ryan, and J. N. Simone, Phys. Rev. D **58**, 014506 (1998) [hep-ph/9711426].
19. C. Bernard *et al.* [MILC Collaboration], Phys. Rev. D **66**, 094501 (2002) [hep-lat/0206016].
20. F. Abe *et al.* [CDF Collaboration], Phys. Rev. Lett. **81**, 2432 (1998) [hep-ex/9805034].
21. DØ Collaboration, DØ Note 4539-CONF (August 14, 2004);  
E. Cheu [DØ Collaboration], Int. J. Mod. Phys. A **20**, 3664 (2005).
22. K. Anikeev *et al.*, “ $B$  physics at the Tevatron: Run II and beyond,” hep-ph/0201071.
23. I. F. Allison *et al.* [HPQCD and Fermilab Lattice Collaborations], Nucl. Phys. Proc. Suppl. **140**, 440 (2005) [hep-lat/0409090].
24. I. F. Allison *et al.* [HPQCD and Fermilab Lattice Collaborations], Phys. Rev. Lett. **94**, 172001 (2005) [hep-lat/0411027].
25. D. Acosta *et al.* [CDF Collaboration], “Evidence for the exclusive decay  $B_c^\pm \rightarrow J/\psi \pi^\pm$  and measurement of the mass of the  $B_c$  meson,” hep-ex/0505076.
26. H. P. Shanahan, P. Boyle, C. T. H. Davies, and H. Newton [UKQCD Collaboration], Phys. Lett. B **453**, 289 (1999) [hep-lat/9902025].
27. G. P. Lepage and B. A. Thacker, Nucl. Phys. Proc. Suppl. **4**, 199 (1987);  
B. A. Thacker and G. P. Lepage, Phys. Rev. D **43**, 196 (1991).
28. A. X. El-Khadra, A. S. Kronfeld, and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997) [hep-lat/9604004].
29. G. P. Lepage *et al.*, Phys. Rev. D **46**, 4052 (1992) [hep-lat/9205007].
30. A. S. Kronfeld, Phys. Rev. D **62**, 014505 (2000) [hep-lat/0002008].

**Copyright of International Journal of Modern Physics A: Particles & Fields; Gravitation; Cosmology; Nuclear Physics is the property of World Scientific Publishing Company and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.**

# Update on onium masses with three flavors of dynamical quarks

PoS(LAT2006)175

**Steven Gottlieb\***, **L. Levkova** †

*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*  
*E-mail: sg@indiana.edu*

**M. Di Pierro**

*School of Computer Science, Telecommunications and Information Systems, DePaul University,  
Chicago, Illinois 60604, USA*

**A. El-Khadra**

*Physics Department, University of Illinois, Urbana, Illinois 61801, USA*

**A.S. Kronfeld, P.B. Mackenzie, J. Simone**

*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

**Fermilab Lattice Collaboration and MILC Collaboration**

We update results presented at Lattice 2005 on charmonium masses. New ensembles of gauge configurations with 2+1 flavors of improved staggered quarks have been analyzed. Statistics have been increased for other ensembles. New results are also available for  $P$ -wave mesons and for bottomonium on selected ensembles.

*XXIVth International Symposium on Lattice Field Theory  
July 23-28, 2006  
Tucson, Arizona, USA*

---

\*Speaker.

†Current address: Physics Department, University of Utah, Salt Lake City, UT 84112, USA

## 1. Introduction

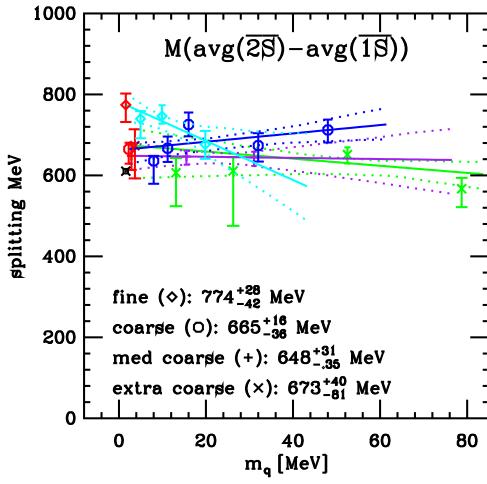
The calculation of the onium spectrum from first principles from lattice QCD represents a significant goal. There are a number of states that are stable to strong decay and far from thresholds (that might make finite size effects more significant). It should be possible to accurately calculate all of their masses. Further, since there are no valence up and down quarks, we have to consider only the sea quark mass dependence. However, one must be careful in dealing with heavy quarks on the lattice because  $aM$  is not small. Some early results using dynamical quark configurations were published in 2004 [1]. Using clover type quarks with the Fermilab interpretation [2], we have been calculating the onium spectrum for some time [3] on MILC gauge configurations [4]. The HPQCD/UKQCD collaborations have successfully been using NRQCD to treat the bottom quark on many of the same ensembles [5]. Recently, they have started to use highly improved staggered quarks (HISQ) to study charm [6].

At Lattice 2005 [7], we presented results for lattice spacings  $a \approx 0.18, 0.12$  and  $0.09$  fm. These are denoted extra coarse, coarse and fine, respectively. This year we have several new ensembles with a lattice spacing of  $\approx 0.15$  fm, denoted medium coarse in the plots. On these new ensembles, we have tuned the dynamical strange quark mass closer to its physical value based upon experience with the earlier ensembles. At the fine lattice spacing, we have new results on a more chiral ensemble with  $am_l = 0.0031$  and  $am_s = 0.031$  on a  $40^3 \times 96$  grid. All of our results for bottomonium are new. We also have some new results for the  $\chi_{c2}$ . (See Table 1 for details of the ensembles.)

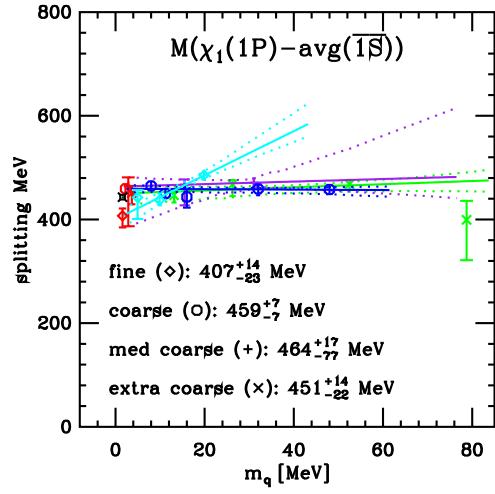
To calculate the onium spectrum, we need to find an appropriate value of the heavy quark hopping parameter  $\kappa$ . We do this by studying the  $D_s$  and  $B_s$  kinetic masses. We do this study on one ensemble for each lattice spacing and use the selected values of  $\kappa_c$  and  $\kappa_b$  for all the ensembles with that lattice spacing. Errors in the kinetic masses tend to be large, and so we calculate mass

$am_q / am_s$	$10/g^2$	$\approx a$	size	volume	config.
0.0492 / 0.082	6.503	0.18	$16^3 \times 48$	$(2.8 \text{ fm})^3$	401
0.0328 / 0.082	6.485	0.18	$16^3 \times 48$	$(2.8 \text{ fm})^3$	331
0.0164 / 0.082	6.467	0.18	$16^3 \times 48$	$(2.8 \text{ fm})^3$	645
0.0082 / 0.082	6.458	0.18	$16^3 \times 48$	$(2.8 \text{ fm})^3$	400
0.0194 / 0.0484	6.586	0.15	$16^3 \times 48$	$(2.4 \text{ fm})^3$	631
0.0097 / 0.0484	6.566	0.15	$20^3 \times 48$	$(3.0 \text{ fm})^3$	631
0.03 / 0.05	6.81	0.12	$20^3 \times 64$	$(2.4 \text{ fm})^3$	549
0.02 / 0.05	6.79	0.12	$20^3 \times 64$	$(2.4 \text{ fm})^3$	460
0.01 / 0.05	6.76	0.12	$20^3 \times 64$	$(2.4 \text{ fm})^3$	593
0.007 / 0.05	6.76	0.12	$20^3 \times 64$	$(2.4 \text{ fm})^3$	403
0.005 / 0.05	6.76	0.12	$24^3 \times 64$	$(2.9 \text{ fm})^3$	397
0.0124 / 0.031	7.11	0.09	$28^3 \times 96$	$(2.4 \text{ fm})^3$	517
0.0062 / 0.031	7.09	0.09	$28^3 \times 96$	$(2.4 \text{ fm})^3$	557
0.0031 / 0.031	7.08	0.09	$40^3 \times 96$	$(3.4 \text{ fm})^3$	504

**Table 1:** Ensembles used in this calculation.



**Figure 1:** Splitting between the spin-averaged 2S and 1S states of charmonium.



**Figure 2:** Splitting between the  $\chi_{c1}(1P)$  and spin-averaged 1S states.

splittings based on differences in the rest energy.

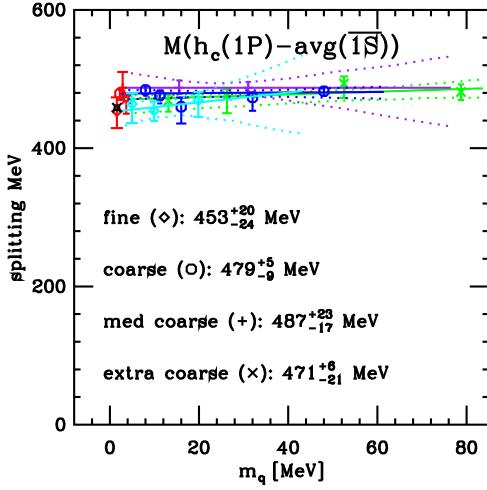
## 2. Charmonium Results

We will examine a number of splittings in the charmonium spectrum. Let's look at the difference between the spin averaged 2S and 1S states. In Fig. 1, we plot the splitting in MeV as a function of the light bare sea quark mass. The experimental value is shown as a black fancy cross. The points in red are linear extrapolations in quark mass. We find that at each lattice spacing (except for the fine lattice) the extrapolated value is in good agreement with experiment. On the fine lattice, there is a considerable slope and the extrapolated value is quite high. (We note that as the horizontal axis is the bare quark mass, the slope is not a physical quantity; it also reflects a change in the mass renormalization as the lattice spacing changes.)

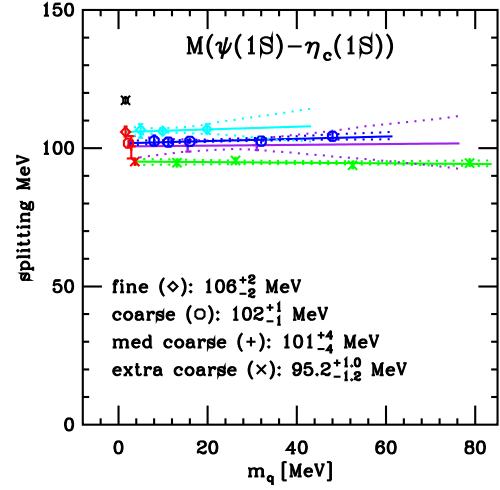
Next we turn to issues of fine structure and look at the splitting between the  $\chi_{c1}(1P)$  and the spin averaged 1S mass. It would be more natural to compare with the spin average of all the 1P states, but the  $\chi_{c2}$  mass is not yet available on all ensembles, so we compare with the 1S spin average. Using the same color scheme and symbols as in the previous figure, we see in Fig. 2, that there is good agreement with experiment except on the fine ensembles which again exhibit a substantial slope. In this case, the two more chiral ensembles are in good agreement with the experimental value, but the ensemble with the heaviest value of the light sea quark mass gives too large a value for the splitting, leading to a large slope and too small a chiral limit.

For the  $h_c(1P)$  state shown in Fig. 3, we find good agreement with the experimental value on all ensembles. For the finest lattice spacing, there is only a modest slope from the chiral extrapolation.

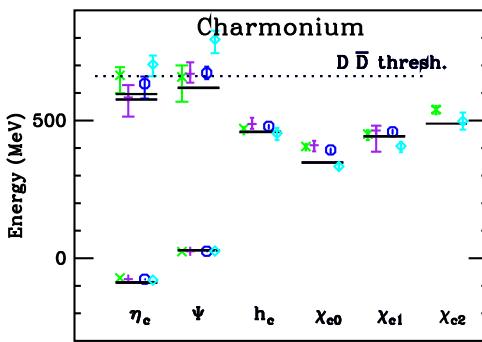
Turning next to the hyperfine splitting, we look at the  $J/\psi(1S) - \eta_c(1S)$  mass splitting in Fig. 4. In this case, we find that the splittings are systematically small, but that the value is increasing toward the experimental value as the lattice spacing decreases.



**Figure 3:** Splitting between the  $h_c(1P)$  and spin-averaged  $1S$  states.

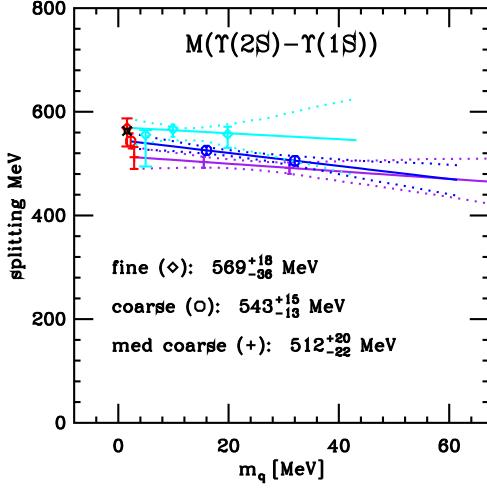


**Figure 4:** Hyperfine splitting of the  $1S$  states.

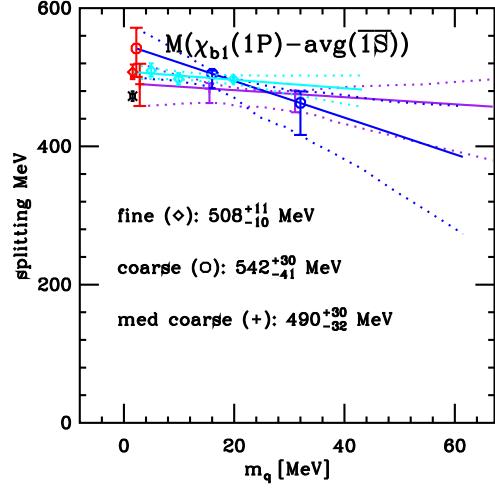


**Figure 5:** Summary of charmonium spectrum.

The  $\chi_{c2}(1P)$  has only been studied on two ensembles so far. We have new results on one fine ensemble. In Fig. 5, we summarize the results for all the states studied. Except for the  $\chi_{c2}(1P)$ , we plot results from our linear chiral extrapolation for each lattice spacing. For the ground states, if we focus our attention on the diamonds representing our smallest lattice spacing, we find the most serious discrepancy between our results and experiment is for the  $\chi_{c1}$ . We have seen that our linear chiral extrapolation may be the culprit here, as the two more chiral ensembles are in good agreement with the experimental value. The  $S$  wave first excited states are not that well determined, but are rather heavy compared to the observed values. We have seen that on the finest lattice spacing, the high slope of the chiral extrapolation is accentuating the difference between our calculation and observations. Furthermore, the observed states are quite close to the  $D\bar{D}$  threshold, which makes these states harder to calculate on the lattice without careful attention to finite volume effects. Thus, we are not seriously concerned about the high masses we are seeing for the  $2S$  states.



**Figure 6:** Chiral extrapolation of the  $\Upsilon(2S)$ - $\Upsilon(1S)$  splitting.



**Figure 7:** Chiral extrapolation of the  $\chi_{b1}(1P)$  - spin averaged 1S splitting.

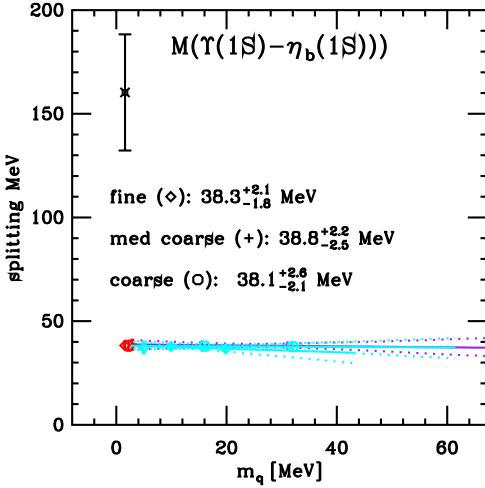
### 3. Bottomonium Results

We have new results this year for bottomonium on the three smallest lattice spacings. Some important ground states in this system, the  $\eta_b$  and  $h_b$ , have not been observed, so we shall modify some of the splittings we display. It is also interesting to compare our results with those using NRQCD for the heavy quarks [5]. We expect that our results have larger discretization errors, when comparing them to the results of Ref. [5] at a fixed lattice spacing, because the NRQCD action used in that work is more improved.

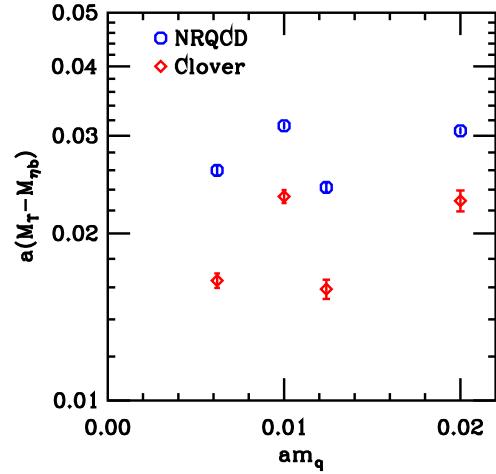
In Fig. 6, we plot the splitting between the  $\Upsilon(2S)$  and  $\Upsilon(1S)$  masses. On the fine ensembles, the result is in good agreement with experiment. With larger lattice spacings, the splitting is a bit low. We look only at the  $\Upsilon$  level since the  $\eta_b$  masses are not well measured. In the bottomonium system, the  $2S$  and  $3S$  states are both below the  $B\bar{B}$  threshold. Thus, possible finite size effects from a nearby threshold are not an issue, and we should get the  $2S$  levels right.

In considering the fine structure of the bottomonium spectrum we are faced with two issues. First, the  $h_b$  has not been observed so we cannot compare our results with the experimental value (but we can make a prediction). Second, we have no results for  $\chi_{b2}$  yet, so we can't compute a spin average of the  $P$  states. So, we consider the  $\chi_{b1}$ , in the middle of the three  $^3P_J$  states. We compare it to the spin averaged  $1S$  states assuming our  $\eta_b$  mass is correct. The result shown in Fig. 7 has the splitting about 35 MeV too large on the fine lattices.

The hyperfine splitting in the  $1S$  states is shown in Fig. 8. We find a value of about 38 MeV for this splitting for all  $a$ . The experimental result plotted comes from the PDG and is based on the observation of a single  $\eta_b$ . It should be noted that a preliminary result from CDF with higher statistics had a splitting of only 15 MeV. With this factor of 10 difference between the experimental results, it is hard to reach any definite conclusion about how accurate our hyperfine splitting is for the  $1S$  states. It turns out, however, that on four ensembles we can directly compare our calculation with the HPQCD results. In Fig. 9, we compare the splitting in lattice units. The results are plotted



**Figure 8:** Hyperfine splitting of the bottomonium 1S states.



**Figure 9:** Comparison of HPQCD's hyperfine splitting using NRQCD with this work.

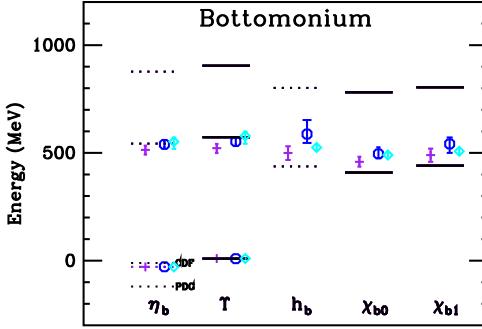
on a semilogarithmic scale so that equal vertical differences represent equal fractional differences in the splittings. The upper two octagons and diamonds are coarse lattice results. The lower two octagons and diamonds are on fine ensembles. Points with the same bare quark mass are calculated on the same ensembles, but the HPQCD collaboration did not analyze every available configuration, so the actual set of configurations selected for analysis differs. We see that the fraction difference in the hyperfine splittings is larger at the smaller lattice spacing and that the NRQCD hyperfine splitting is larger than that for clover. The difference is not unexpected, because the leading error for this splitting with the clover action is of order  $mv^6$ , whereas in Ref. [5] it is of order  $mv^8$ .

In Fig. 10, we summarize the spectrum of observed states using the spin-averaged 1S states (assuming a 38 MeV  $\Upsilon$ - $\eta_b$  splitting) to set the additive constant needed to go from splittings to masses. Solid lines on the plot are experimentally determined masses. Dashed lines on the plot denote unobserved (or poorly observed) states. In the case of the  $\eta_b$  we use dashed lines and mark them CDF and PDG as discussed above. We find good agreement with experiment for the  $\Upsilon(2S)$ . The  $\eta_b(2S)$  is in good agreement with theoretical expectations. Our masses for the  $P$  wave states  $h_b$ ,  $\chi_{b0}$  and  $\chi_{b1}$  are too large.

#### 4. Conclusions and Outlook

We observe only a mild dependence on the sea quark mass for most of the quantities studied here. In particular, the hyperfine splittings and the 2S-1S splittings appear to be essentially independent of the sea quark mass. Hence a linear extrapolation seems adequate. For the  $P$ -wave states, we do observe a sea quark mass dependence at the 10% level on some ensembles.

There are several positive features of the charmonium spectrum. The  $h_c$  and  $\chi_{c1}$  masses look quite good (although the latter is driven low by the chiral extrapolation on the fine ensembles). At the smallest lattice spacing studied so far, the  $P$  wave splitting looks good (but see previous sentence). The hyperfine splitting is too small but improving as the lattice spacing decreases.



**Figure 10:** Summary of the bottomonium spectrum for ensembles with  $a \approx 0.15, 0.12$  and  $0.09$  fm.

However, the  $2S$  states are not accurately calculated. For the bottomonium spectrum, the excited state splitting looks good. It is not yet possible to test the hyperfine splitting, but our splitting is smaller than that coming from NRQCD calculations. Our  $P$  wave states seem too heavy.

In the coming year, we expect to increase our statistics on a number of existing ensembles. In addition, MILC is generating ensembles with a lattice spacing of  $0.06$  fm and has plans to reduce the lattice spacing to  $0.045$ . There is also some chance that there will be some ensembles with  $a = 0.105$  fm. Currently, we are using an automatic criterion for picking the best fit. We need to consider alternative methods and whether picking a single fit properly reflects the systematic errors.

The  $P$  wave splitting in the bottomonium spectrum does not seem to be in agreement with experiment. It will be interesting to see if the bottomonium spectrum improves as we reduce the lattice spacing (and  $am_b$ ). Kronfeld and Oktay [8] have been developing a highly improved clover quark action that we hope to use in the future.

## References

- [1] C. Davies *et al.* [Fermilab Lattice, HPQCD, MILC, UKQCD], Phys. Rev. Lett. **92** (2004) 022001 [hep-lat/0304004].
- [2] A. El-Khadra, A.S. Kronfeld, P.B. Mackenzie, Phys. Rev. D **55** (1997) 3933 [hep-lat/9604004].
- [3] M. di Pierro *et al.*, Nucl. Phys. B (Proc. Suppl.) **129** (2004) 340 [hep-lat/0310042].
- [4] C. Bernard *et al.* [MILC], Phys. Rev. D **64** (2001) 054506 [hep-lat/0104002]; C. Aubin *et al.* [MILC], Phys. Rev. D **70** (2004) 094505 [hep-lat/0402030].
- [5] A. Gray *et al.* [HPQCD, UKQCD], Phys. Rev. D **72** (2005) 094507 [hep-lat/0507013].
- [6] E. Follana *et al.* [HPQCD] [hep-lat/0610092]; Nucl. Phys. B (Proc. Suppl.) **129** (2004) 447 [hep-lat/0311004]; **129** (2004) 384 [hep-lat/0406021].
- [7] S. Gottlieb *et al.*, PoS(LAT2005)203 [hep-lat/0510072].
- [8] A.S. Kronfeld and M.B. Oktay, PoS(LAT2006)159 [hep-lat/0610069]; M.B. Oktay *et al.*, Nucl. Phys. B (Proc. Suppl.) **129** (2004) 349 [hep-lat/0309107]; **119** (2003) 464 [hep-lat/0209150].

# A determination of the $B_s^0$ and $B_d^0$ mixing parameters in 2+1 lattice QCD

POS(LATTICE 2007)354

**R. Todd Evans, Elvira Gámiz<sup>\*</sup> and Aida X. El-Khadra***Department of Physics, University of Illinois, Urbana, IL 61801, USA**E-mail: rtevans@uiuc.edu, megamiz@uiuc.edu, axk@uiuc.edu***Massimo Di Pierro***School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, IL, USA**MDiPierro@cti.depaul.edu***Fermilab Lattice and MILC Collaborations**

We report on the advances in our unquenched calculation of the matrix elements relevant for the analysis of  $B^0 - \bar{B}^0$  mixing using the Asqtad (light quark) and Fermilab (heavy quark) actions. We have calculated the hadronic parameters for the mass and width differences in the neutral  $B$  meson system. Preliminary results are presented for  $f_{B_q}^2 B_q$  as well as for the ratio  $\xi^2 = f_{B_s}^2 B_{B_s} / f_{B_d}^2 B_{B_d}$ .

*The XXV International Symposium on Lattice Field Theory  
July 30-4 August 2007  
Regensburg, Germany*

---

<sup>\*</sup>Speaker.

## 1. Introduction

The very accurate experimental measurements of the mass differences between the heavy and light  $B_s^0$  and  $B_d^0$  mass eigenstates,  $\Delta M_s$  [1] and  $\Delta M_d$  [2], that describe the  $B_s^0 - \bar{B}_s^0$  and  $B_d^0 - \bar{B}_d^0$  mixings respectively, make improving the theoretical study of these quantities crucial. In the standard model (SM), the mass difference is given by [3]

$$\Delta M_{s(d)}|_{theor.} = \frac{G_F^2 M_W^2}{6\pi^2} |V_{ts(d)}^* V_{tb}|^2 \eta_2^B S_0(x_t) M_{B_{s(d)}} f_{B_{s(d)}}^2 \hat{B}_{B_{s(d)}}, \quad (1.1)$$

where  $x_t = m_t^2/M_W^2$ ,  $\eta_2^B$  is a perturbative QCD correction factor,  $S_0(x_t)$  is the Inami-Lim function and the products  $f_{B_{s(d)}}^2 \hat{B}_{B_{s(d)}}$  parameterize the hadronic matrix elements in the effective theory with  $f_{B_{s(d)}}$  the  $B_{s(d)}^0$  decay constants and  $\hat{B}_{B_{s(d)}}$  the (renormalization group invariant) bag parameters. The hadronic matrix elements can be calculated in lattice QCD. Our current knowledge of them limits the accuracy with which the CKM matrix elements appearing in Eqn. (1.1) can be determined from the experimental measurements of  $\Delta M_{s(d)}$ . The goal of our project is to calculate all the hadronic matrix elements which are relevant for the mass and width differences in the  $B_{s(d)}^0$  systems in unquenched lattice QCD at the few percent level.

Many of the uncertainties that affect the theoretical calculation of the decay constants and bag parameters cancel totally or partially if one takes the ratio  $\xi^2 = f_{B_s}^2 B_{B_s}/f_{B_d}^2 B_{B_d}$ . Hence, this ratio and therefore the combination of CKM matrix elements related to it by Eqn. (1.1) can be determined with a significantly smaller error than the individual matrix elements. This is a crucial ingredient in the unitarity triangle analysis. In these proceedings we report our preliminary results for the determination of  $\xi$ , as well as for the quantities  $f_{B_q}^2 B_{B_q}$ .

Other work on this subject using 2+1 lattice QCD methods can be found in [4].

## 2. Operators, actions and matching calculation

The whole set of operators whose matrix elements are needed to determine the  $B_{s(d)}^0$  mixing parameters are

$$\begin{aligned} Q^{s(d)} &= [\bar{b}^i \gamma_\mu (1 - \gamma_5) s^i(d^i)] [\bar{b}^j \gamma^\mu (1 - \gamma_5) s^j(d^j)], \\ Q_S^{s(d)} &= [\bar{b}^i (1 - \gamma_5) s^i(d^i)] [\bar{b}^j (1 - \gamma_5) s^j(d^j)], \\ Q_3^{s(d)} &= [\bar{b}^i (1 - \gamma_5) s^j(d^j)] [\bar{b}^j (1 - \gamma_5) s^i(d^i)], \end{aligned} \quad (2.1)$$

where  $i, j$  are color indices. In these proceedings we focus on the results for the first two pairs of operators, enough to determine  $\Delta M_{s(d)}$ , and leave the study of the third pair, needed for an improved determination of  $\Delta \Gamma_{s(d)}$ , for a forthcoming publication [5].

We use the Fermilab action [6] for the  $b$  valence quarks and the Asqtad action [7], for the light sea and valence quarks,  $u, d$  and  $s$ . The Fermilab action has errors starting at  $O(\alpha_s \Lambda_{QCD}/M)$  and  $O((\Lambda_{QCD}/M)^2)$ , while the errors of the Asqtad action are  $O(\alpha_s a^2, a^4)$ .

The products  $f_{B_{s(d)}}^2 B_{B_{s(d)}}^{\overline{MS}}$  in Eqn. (1.1) parametrize the matrix elements by

$$\langle \bar{B}_s^0 | Q^{s(d)} | B_s^0 \rangle^{\overline{MS}}(\mu) = \frac{8}{3} M_{B_{s(d)}}^2 f_{B_{s(d)}}^2 B_{B_{s(d)}}^{\overline{MS}}(\mu). \quad (2.2)$$

The lattice matrix elements  $\langle \bar{B}_{s(d)}^0 | Q^{s(d)} | B_{s(d)}^0 \rangle^{\text{lat}}$  determine  $f_{B_{s(d)}}^2 B_{B_{s(d)}}$  at tree level. Beyond tree-level, the operators  $Q^{s(d)}$ , mix with  $Q_S^{s(d)}$  both on the lattice and in the continuum. Including one-loop corrections, the renormalized matrix element is given by

$$\frac{a^3}{2M_{B_{s(d)}}} \langle Q^{s(d)} \rangle^{\overline{MS}}(\mu) = [1 + \alpha_s \cdot \rho_{LL}(\mu, m_b)] \langle Q^{s(d)} \rangle^{\text{lat}}(a) + \alpha_s \cdot \rho_{LS}(\mu, m_b) \langle Q_S^{s(d)} \rangle^{\text{lat}}(a). \quad (2.3)$$

The  $O\left(\frac{\Lambda_{QCD}}{M}\right)$  improvement is implemented by a rotation of the  $b$  quark as explained in [6], so the perturbative matching errors start at  $O(\alpha_s^2, \alpha_s \Lambda/aM)$ . The matching coefficients  $\rho_{LL}$  and  $\rho_{LS}$  are the differences between the continuum  $\overline{MS}$  and lattice renormalization coefficients calculated at one-loop order. We have calculated these coefficients for the same choice of lattice actions as used in the numerical simulations. We have checked that our results have the correct infrared behavior, that they are correct in the massless limit, and that they are gauge invariant. However, our results for the matching coefficients are still preliminary, because not all diagrams have been independently checked.

The optimal value of the strong coupling constant to be used in Eqn. (2.3) is  $\alpha_V(q^*)$ . Missing a calculation of  $q^*$  for the specific processes we are studying, we choose  $q^* = 2/a$ , very close to the  $q^*$  calculated for heavy-light currents. The specific values for  $\alpha_s$  we use are given in Table 1.

### 3. Simulation details

The matrix elements needed to determine both  $f_{B_q}^2 B_{B_q}$  and  $B_{B_q}$ , are extracted from the following three-point and two-point functions

$$C_O(t_1, t_2) = \sum_{\vec{x}, \vec{y}} \langle \bar{b}(\vec{x}, t_1) \gamma_5 q(\vec{x}, t_1) | O(0) | \bar{b}(\vec{y}, t_2) \gamma_5 q(\vec{y}, t_2) \rangle, \\ C_Z(t) = \sum_{\vec{x}} \langle \bar{b}(\vec{x}, t) \gamma_5 q(\vec{x}, t) \bar{q}(0) \gamma_5 b(0) \rangle, \quad C_{A_4}(t) = \sum_{\vec{x}} \langle \bar{b}(\vec{x}, t) \gamma_0 \gamma_5 q(\vec{x}, t) \bar{q}(0) \gamma_5 b(0) \rangle, \quad (3.1)$$

where the operator  $O$  is any  $Q^{s(d)}$  or  $Q_S^{s(d)}$  defined in Eqn. (2.1). The  $B$  meson operators are smeared at the sink with a 1S onium wavefunction. All the correlation functions in Eqn. (3.1) are calculated using the open meson propagator method described in [8].

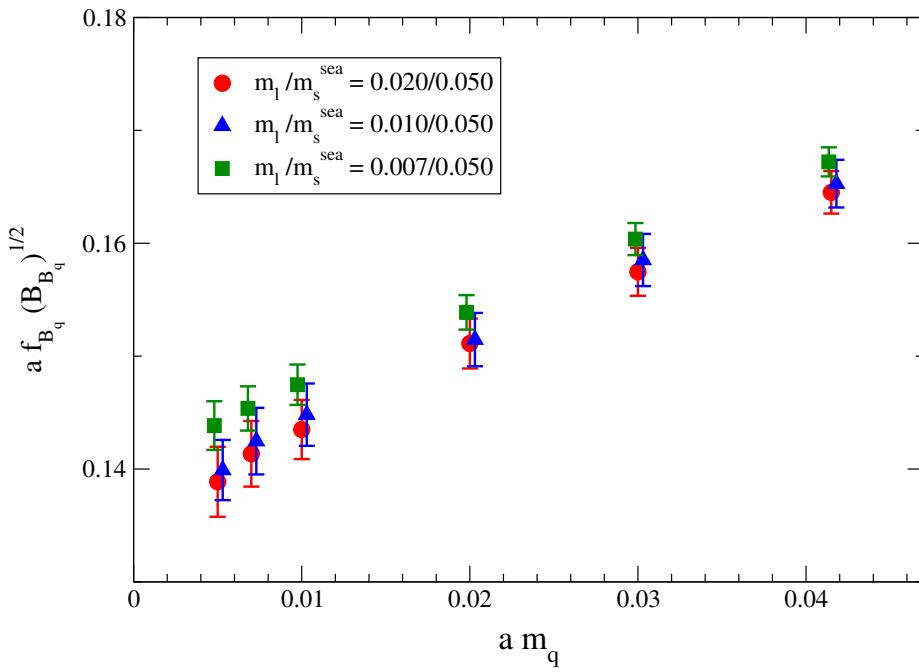
We have performed these calculations on the MILC coarse lattices ( $a = 0.12 \text{ fm}$ ) with 2+1 sea quarks and for three different sea light quark masses. The strange sea quark mass is always set to 0.050. The light sea masses,  $m_l \equiv m_{light}^{sea}$ , number of configurations and other simulation details are collected in Table 1. The mass of the bottom quark is fixed to its physical value, while for each sea quark mass we determine the different matrix elements for six different values of the light valence quark mass in a generic meson  $B_q^0$ ,  $m_q = 0.0415, 0.03, 0.020, 0.010, 0.007, 0.005$ . We use  $m_q = 0.0415$  for the valence strange mass in our simulations. It is close to the physical strange quark mass,  $m_s^{phys.} = 0.036$  [9]. The matrix elements of the operators are extracted from simultaneous fits of three-point and two-point functions using Bayesian statistics.

$m_l/m_s^{\text{sea}}$	Volume	$N_{\text{confs}}$	$a^{-1}(\text{GeV})$	$\alpha_s = \alpha_V(2/a)$	$N_{\text{sources}}$
0.020/0.050	$20^3 \times 64$	460	1.605(29)	0.31	4
0.010/0.050	$20^3 \times 64$	590	1.596(30)	0.31	4
0.007/0.050	$20^3 \times 64$	890	1.622(32)	0.31	4

**Table 1:** Simulation parameters and  $\alpha_s$  used in the matching with the continuum.  $m_l$  is the light sea quark mass.

#### 4. Results

The results in Fig. 1 show  $f_{B_q} \sqrt{B_{B_q}^{\overline{MS}}(m_b)}$  in lattice units as a function of the light valence mass  $a m_q$ . The errors shown are statistical errors only; the analysis of the systematic errors is not yet complete.



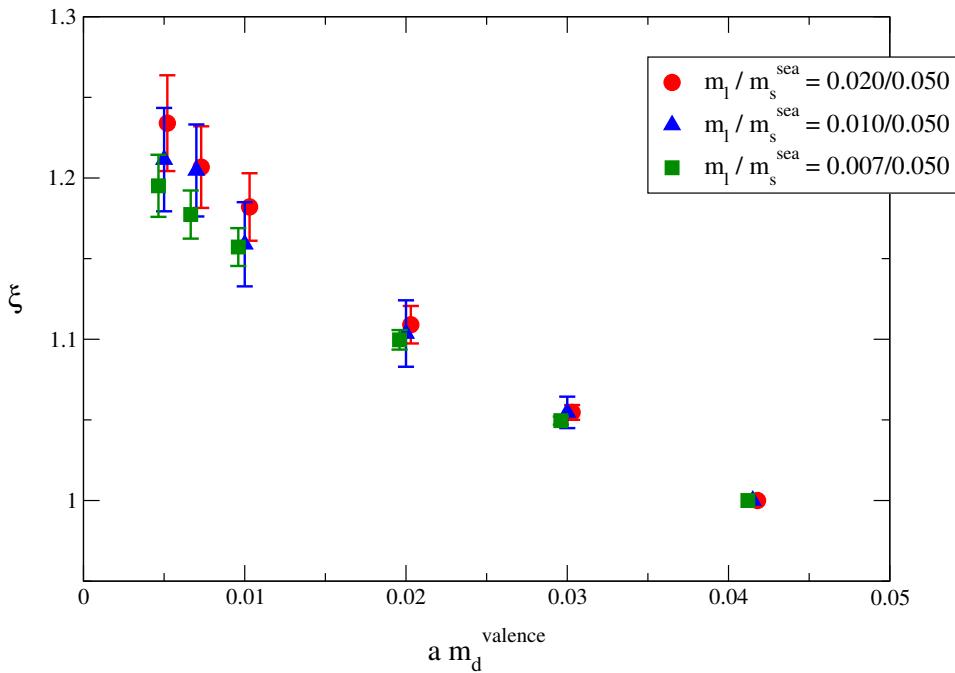
**Figure 1:**  $f_{B_q} \sqrt{B_{B_q}^{\overline{MS}}(m_b)}$  in lattice units. The different symbols and colors correspond to different values of the sea light quark masses,  $m_l$ .

The statistical errors range between 1 – 3%. Some conclusions can be already extracted from this plot. The light sea quark mass dependence of  $f_{B_q} \sqrt{B_{B_q}}$  is small compared to the statistical errors. The dependence on the light valence quark mass, however, is noticeable within statistics. In order to get a value for  $f_{B_s} \sqrt{B_{B_s}^{\overline{MS}}(m_b)}$ , since the  $s$  valence quark mass we are using is slightly larger than the physical one, we need an interpolation in the  $s$  valence quark mass together with a chiral

POS(LATTICE 2007) 354

extrapolation to the physical sea quark masses. To determine  $f_{B_d} \sqrt{B_{B_d}^{\overline{MS}}(m_b)}$  we need extrapolations in both the  $d$  valence quark mass and the sea quark masses. This is in progress.

In Fig. 2 we plot the ratio  $\xi = f_{B_s} \sqrt{B_{B_s}} / f_{B_d} \sqrt{B_{B_d}}$  as a function of the  $d$  valence quark mass in the denominator. Again, our results are preliminary for the same reasons as mentioned before and the errors are only statistical. Most of the systematic errors cancel in the ratio, but not those associated with the chiral extrapolation in the light valence quark mass.



**Figure 2:**  $\xi$  as a function of the valence  $d$  mass for three different values of the light sea quark masses.

#### 4.1 Chiral extrapolation

The continuum chiral expansion of the hadronic matrix element  $\langle \bar{B}_q | Q | B_q \rangle$  at NLO in (partially quenched) heavy meson chiral perturbation theory (HMChPT) is given by [10]

$$\langle \bar{B}_q | Q | B_q \rangle = \beta(1 + w(T_q + W_q + S_q)) + c_0 m_q + c_1(m_U + m_D + m_S) \quad (4.1)$$

where  $m_U, m_D, m_S$  are the sea quark masses and  $m_q$  the light valence quark mass.  $\beta, w, c_0$  and  $c_1$  are low energy constants (LECs) to be determined from the fits. The functions  $T_q, W_q$ , and  $S_q$  contain the chiral logs and correspond to tadpole-, wave-function, and sunset-type contributions, respectively.

The effects of  $O(a^2)$  taste changing interactions can be included in Eqn. (4.1) using staggered chiral perturbation theory (SChPT). In that case, the chiral log functions are modified to depend

	$f_{B_q} \sqrt{B_{B_q}}$	$\xi$
statistics	1 – 3	1 – 2
scale( $a^{-1}$ )	0.9	0
Higher order matching	~ 4.5	cancel to a large extent
Heavy quark discret.	2 – 3	< 0.5
Light quark discret. + $\chi$ PT fits	Work in progress	

**Table 2:** Error budget for  $f_{B_q} \sqrt{B_{B_q}}$  and  $\xi$  in percent.

on the masses of the different taste multiplets. Explicit expressions from SChPT for heavy-light bilinear quantities can be found in [11, 12]. Similar terms are expected to contribute to our four-quark operators. The modified chiral logs contain other fixed LEC's, most of which are already determined to a high degree of certainty [12]. The logs also contain constants from heavy quark effective theory, in particular the mass splitting between the vector and pseudoscalar heavy mesons,  $\Delta* = M_B* - M_B$ , and the mass splittings between the pseudoscalar heavy mesons containing different valence and sea light quarks,  $\delta_{qr} = M_{B_r} - M_{B_q}$ . These HQET constants can be determined directly from the two-point function fits and used as input into the chiral fits, with the experimental values then used in the extrapolation.

We are still in the process of determining the exact SChPT form of Eqn. (4.1). Once the functional SChPT form of Eqn. (4.1) is completely determined, we plan to use it to simultaneously fit it to our lattice data points for all sea and valence quark masses, and to determine the unknown LEC's in the process. For the systematic error analysis, we plan to study the effects of changing the SChPT form, for example by adding NNLO analytic terms, and the effects of allowing the more poorly known fixed parameters to vary. Our physical results will then be obtained by turning the taste-violations off, and extrapolating (interpolating) to the physical light (strange) sea and valence quark masses.

We will quote results for  $\xi$ ,  $f_{B_s} \sqrt{B_{B_s}}$ ,  $f_{B_d} \sqrt{B_{B_d}}$ ,  $B_{B_s}$ , and  $B_{B_d}$  when this step is completed. We expect light quark discretization effects to be an important source of uncertainty until we calculate the three-point correlators at several lattice spacings and use these in the chiral fits.

## 5. Summary and future work

We have presented preliminary results for  $f_{B_q} \sqrt{B_{B_q}}$  for six different values of  $m_q$  as well as for the ratio  $\xi$  with five different values of  $m_d^{valence}$ . Our analysis on three ensembles with different sea light quark masses gives statistical errors between 1 – 3% for  $f_{B_q} \sqrt{B_{B_q}}$  and 1 – 2% for the ratio  $\xi$ . The systematic error analysis is in progress, see Table 2.

Two important sources of error in the calculation of  $f_{B_q} \sqrt{B_{B_q}}$ , the matching uncertainties and heavy quark discretization errors, are expected to cancel to a large extent when taking the ratio. We have already checked that the difference between the tree level and the one-loop results for  $\xi$  with our preliminary results for the renormalization constants is less than 0.5%. The higher order matching errors in Table 2 have been naively estimated as being  $O(1 \times \alpha_s^2)$  for the coarse lattice. Heavy quark discretization effects in the table are estimated by power counting [13].

We are in the process of generating lattice data on the coarse lattice with a smaller light sea quark mass,  $m_l = 0.005$ , which will further constrain the chiral extrapolations. Better fitting approaches and different smearings that could reduce statistical errors further are also being investigated. Results from these improvements will be presented in a future publication [5]. We also plan to present results for the decay width differences  $\Delta\Gamma_s$  and  $\Delta\Gamma_d$ , for which we have already calculated all the hadronic matrix elements needed. Finally, we plan to improve this analysis by repeating this calculation at other lattice spacings to study the discretization errors in detail. Simulations on finer lattices will reduce both discretization and perturbative matching errors.

### Acknowledgments

The numerical simulations for this work were carried out on the Fermilab lattice QCD clusters, which are a computing resource of the USQCD collaboration and are funded by the DOE. We are grateful to the Fermilab Computing Divisions for operating and maintaining the clusters. This work was supported in part by the DOE under grant no. DE-FG02-91ER40677 and by the Junta de Andalucía [P05-FQM-437 and P06-TIC-02302].

### References

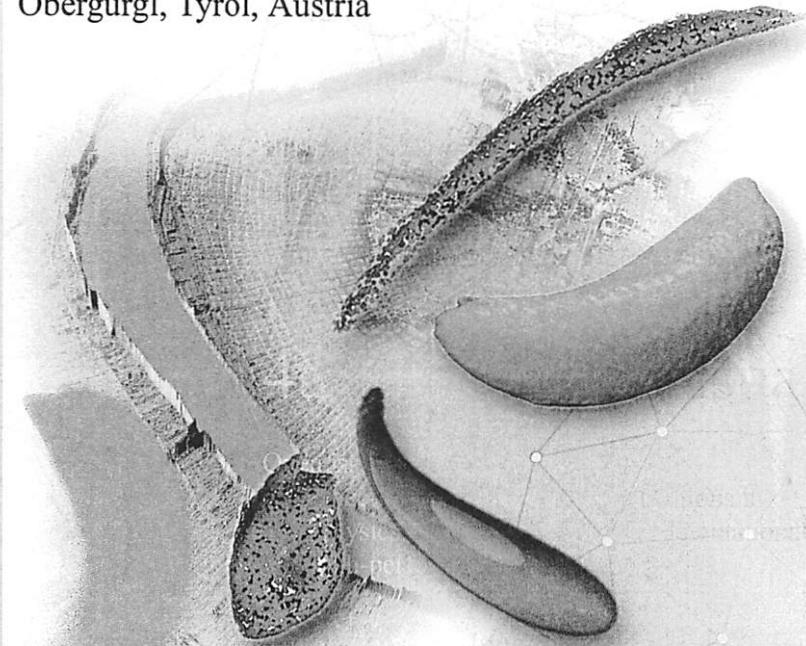
- [1] A. Abulencia *et al.* [CDF Collaboration], Phys. Rev. Lett. **97** (2006) 242003 [arXiv:hep-ex/0609040]; V. M. Abazov *et al.* [D0 Collaboration], Phys. Rev. Lett. **97** (2006) 021802 [arXiv:hep-ex/0603029].
- [2] A world average can be found in W. M. Yao *et al.* [Particle Data Group], J. Phys. G **33** (2006) 1.
- [3] A. J. Buras, M. Jamin and P. H. Weisz, Nucl. Phys. B **347** (1990) 491.
- [4] E. Dalgic *et al.*, Phys. Rev. D **76** (2007) 011501 [arXiv:hep-lat/0610104]; E. Gamiz *et al.*, PoS **LAT2007** (2007) 349; J. Wennekers, these proceedings.
- [5] R.T. Evans,*et al.* (Fermilab Lattice and MILC collaborations), in preparation.
- [6] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55** (1997) 3933 [arXiv:hep-lat/9604004].
- [7] S. Naik, Nucl. Phys. B **316** (1989) 238; G. P. Lepage, Phys. Rev. D **59** (1999) 074502 [hep-lat/9809157]; K. Orginos, D. Toussaint and R. L. Sugar [MILC Collaboration], Phys. Rev. D **60** (1999) 054503 [hep-lat/9903032].
- [8] R.T. Evans, A.X. El-Khadra and M. Di Pierro, PoS **LAT2006** (2006) 081.
- [9] C. Bernard, private communication.
- [10] W. Detmold and C. J. Lin, Phys. Rev. D **76**, 014501 (2007) [arXiv:hep-lat/0612028].
- [11] C. Aubin and C. Bernard, Phys. Rev. D **73**, 014515 (2006) [arXiv:hep-lat/0510088].
- [12] C. Aubin *et al.* [MILC Collaboration], Phys. Rev. D **70**, 114501 (2004) [arXiv:hep-lat/0407028].
- [13] C. Aubin *et al.*, Phys. Rev. Lett. **95**, 122002 (2005) [arXiv:hep-lat/0506030].

Werner Benger  
René Heinzl  
Wolfgang Kapferer  
Wolfram Schoor  
Mayank Tyagi  
Shalini Venkataraman  
Gunther H. Weber  
(Eds.)

## Proceedings of the 4th High-End Visualization Workshop

Open issues in visualization  
with special concentration on applications in  
astrophysics, numerical relativity, computational fluid dynamics  
and high-performance computing

June 18<sup>th</sup>- 22<sup>th</sup>, 2007  
Obergurgl, Tyrol, Austria



<http://www.vizworkshop.at>

Delaunay

# A Visualization Toolkit for Lattice Quantum Chromodynamics

Massimo Di Pierro

School of Computer Science, Telecommunications and Information Systems  
DePaul University, 243 S Wabash, Chicago, Illinois, USA  
`mdipierro@cs.depaul.edu`

## Abstract

Lattice QCD is an algorithmic formulation of QCD, the mathematical model that describes how quarks bind together to form composite particles such as proton and neutron. It has been successful in predicting properties of many newly discovered particles, including their mass and decay time. Unfortunately, lattice QCD is very computationally expensive and comprises of sophisticated algorithmic manipulations of large data-structures whose interpretation is purely statistical. In this paper we provide an overview of both lattice QCD and our work to develop a visualization toolkit to extract information from those data-structures. Our toolkit consists of a set of parallel algorithms for projecting the lattice QCD data structures into 3D scalar fields (for example the topological charge of the vacuum, the energy density, the wave function of the quarks, etc.) and uses VTK for the proper visualization.

## 1 Introduction

Quantum Chromodynamics [Marciano & Pagels, 1978] (QCD) is the mathematical model that best describes interactions among quarks, the basic constituents of most of ordinary matter. Lattice QCD is a formulation of QCD in terms of discretized space and time (lattice) that is suitable for numerical computation. Lattice QCD has been successful at explaining and predicting

properties of composite particles such as the mass and lifetime of protons, neutrons, and many other particles produced by modern particle accelerators such as the Tevatron [FERMILAB-Pub-01/197, ] at Fermilab, LEP and LHC at CERN, and Slac at Stanford.

For a compact introduction to lattice QCD see [Pierro, 2006] and references therein.

Lattice QCD algorithms comprise of massive parallel Monte Carlo simulations. Modern state of the art computations are performed on commercial supercomputers (such as Blue Gene [Bhanot et al., 2005] and the Earth Simulator), clusters of workstations (there are nearly 1000 dedicated nodes at Fermilab [Holmgren, 2005] and Jefferson Laboratory), and dedicated machines (Ape [apeNEXT Collaboration, 2003] in Rome, QCDOC [Gottlieb, 2006] at Columbia University and Brookhaven National Laboratory).

In many Lattice QCD computations, the only published output consists of one number with two or three significative digits. At the same time, hundreds of terabytes of data are generated in the intermediate steps of the computation and are not usually looked at because their interpretation is purely statistical: they are random points in a Monte Carlo ensemble.

The goal of our project is twofold: identifying a set of projection operators that would map this data into 3D scalar fields of physical significance for the purpose of extracting information in a visual format; incorporating these operators into a visualization toolkit that will interface with existing Lattice QCD software libraries such as MILC [Gottlieb, 2002], FermiQCD [Pierro, 2002] and SciDAC [Brower, 2006].

In the next section we will give a brief description of Lattice QCD from a computational point of view and we will discuss examples of projection operators of physical interest. In the third section we will present the high level design of our toolkit. Finally we will draw conclusions, show some images produced by our system and discuss the current status of the project.

## 2 Theoretical Foundations

The ingredients of any Lattice QCD computation are the following:

- In nature, there are 6 *flavors* of *Quarks*. They are the basic constituents of protons and neutrons, as well as of other composite particles that can be produced in particle accelerators and are occasionally produced naturally by cosmic rays. Each quark flavor can best be described

as a complex field  $q_{x,\alpha,i}$ . The index  $x = (\vec{x}, t)$  labels a point on the hypercubic lattice that corresponds to a position in space  $\vec{x}$  and time  $t$ . At point  $x$  in space-time, a quark exhibits a number of local degrees of freedom that are parameterized by the indices  $\alpha = 0, 1, 2, 3$  and  $i = 0, 1, 2$ .  $\alpha$  is referred to as spin index and  $i$  as color index.  $|q_{x,\alpha,i}|^2$  is the probability of finding the quark at a given position  $\vec{x}$  at time  $t$ , in a spin state  $\alpha$  and color state  $i$ .

- Quarks interact with each other by exchanging *gauge bosons*, also known as *gluons*. Gauge bosons can be best described as a complex field  $U_{x,\mu,i,j}$ . Similarly to the case of quarks, the index  $x$  labels a point in space-time, while the indices  $\mu, i$  and  $j$  parameterize the local degrees of freedom. If we define  $P_{x,\mu,\nu} = U_{x,\mu} U_{x+\mu,\nu} U_{x+\nu,\mu}^\dagger U_{x,\nu}^\dagger$  and  $U_{x,-\mu} = U_{x-\mu,\mu}^\dagger$  then<sup>1</sup>

$$F_{x,\mu,\nu}^a = \frac{1}{8} \operatorname{Re} \operatorname{Tr} [\lambda^a (P_{x,\mu,\nu} + P_{x,-\mu,\nu} + P_{x,-\mu,-\nu} + P_{x,-\mu,-\nu})] \quad (1)$$

$$- P_{x,\mu,\nu}^\dagger - P_{x,-\mu,\nu}^\dagger - P_{x,-\mu,-\nu}^\dagger - P_{x,-\mu,-\nu}^\dagger] \quad (2)$$

is the chromo-electro-magnetic tensor associated to gluons of type  $a$ . For each gluon type,  $E_{x,i} = F_{x,0,i}$  is the chromo-electric field and  $B_{x,k} = \sum_{i,j} \epsilon_{i,j,k} F_{x,i,j}$  is the chromo-magnetic field.

- QCD is a *Quantum Field Theory*. This means that there is no deterministic time-evolution for the above fields. In fact, the only meaningful physical observables in any Quantum Field Theory are the *correlations* between the degrees of freedom. Lattice QCD provides prescription rules on how to measure physical observables (for example the mass of a proton) using correlations among field variables. These correlations are computed numerically by averaging the corresponding operator over multiple *field configurations*, also known as *paths* or *histories*. Each configuration represents a possible evolution in time of a small portion of space of about  $(10^{-14}$  meters)<sup>3</sup> for about  $10^{-22}$  seconds.
- Field configurations are generated via a Markov Chain Monte Carlo (MCMC) algorithm using a transition probability that encodes the physical laws of QCD. The “Quantum” aspect of the theory is represented by the stochastic field fluctuations present in the gauge configurations and averaged over.

---

<sup>1</sup>here  $\lambda^a$  is any of the 8 generators of the  $SU(3)$  group

- For practical purposes any Lattice QCD computation is divided into three main steps: 1) gauge field configurations are generated using the MCMC; 2) for each gauge configuration one places the quarks in a certain state and let them evolve according to the Dirac equation in the background gauge field (the solution of the Dirac equation on a given gauge configuration is called a *quark propagator*); 3) the indices of a gauge configuration and its corresponding quark propagators are contracted together to compute the operator corresponding to a given physical observable, which is then averaged over all the gauge configurations.
- A typical gauge configuration  $U$  has a size of 96 points in time and  $64^3$  points in space, corresponding to 7 gigabytes of data. A typical quark propagator  $q$  for a single source on the above gauge configuration has a size of 2.5 gigabytes. Most Lattice QCD computations involve about 1000 gauge configurations and a minimum of 12 quark sources each, thus generating  $10 \div 100$  terabytes of data. Usually gauge configurations and quark propagators are stored and are reused for multiple observables in a semi-industrial fashion. Typical computations require 1-10 million hours of computing time for a Pentium 4 @ 3GHz equivalent CPU.
- One complication consists in the fact that some of the degrees of freedom in the gauge field  $U$  are redundant but cannot be eliminated in the computation. In fact  $U_{x,\mu,i,j}$  must be unitary matrices in the indices  $i, j$  and only operators invariant under the simultaneous transformations  $U_{x,\mu,i,j} \rightarrow \sum_{m,n} \Lambda_{x,i,m} \Lambda_{x+\mu,j,n}^* U_{x,\mu,m,n}$  and  $q_{x,\alpha,i} \rightarrow \sum_m \Lambda_{x,i,m} q_{x,\alpha,m}$  have a physical significance<sup>2</sup>. The above symmetry is called *gauge invariance* and it puts a major constraint on what can be visualized. The gauge invariance symmetry represents the principle of local indistinguishability among quarks of different colors (they are 3 but one cannot tell them apart). This symmetry, motivated by experiments, poses a strong constraint on the model and it almost completely determines the interaction term between quarks and gluons in the Dirac equation for QCD.
- The only input of a Lattice QCD computation are parameters that

---

<sup>2</sup>Here  $\Lambda_{x,i,j}$  is an arbitrary field of unitary matrices in the indices  $i, j$

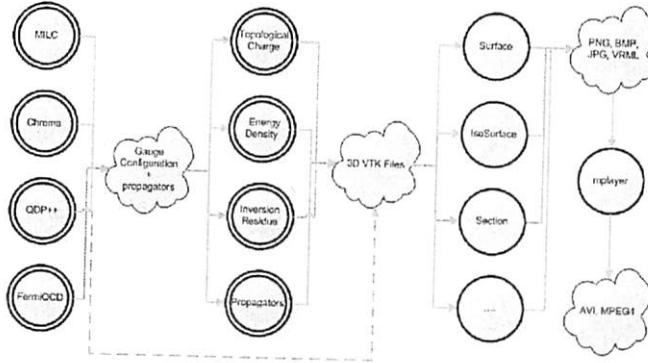


Figure 1: The figure shows the data flow in our visualization toolkit. The double circles represent parallel components including Lattice QCD algorithms and projection plug-ins. The single circle represent Mayavi visualization plug-ins.

loosely correspond to the masses of the quarks and to the lattice discretization scale. Physical observables have a dependence on the discretization scale that is known as *running*. For a sufficiently small discretization scale, the effect of the running can be corrected for and the results of the computation, in physical units, can be made independent on it. Physical observables also depend on the quark masses and, in fact, these are currently determined by a fine tuning the input parameters in Lattice QCD computations.

There are multiple reasons why extracting visual information from Lattice QCD data is important and here we list some of them:

- The gauge configurations have a non-trivial topological structure that can be isolated by removing short term fluctuations, a process called *cooling*. It is known that the total topological charge changes very slowly under MCMC steps, but it is not known how the local charge evolves.
- Some observables are themselves extended objects (for example the wave function of a quark inside a hadron or the energy density in pres-

ence of a quark-antiquark couple) but so far only 1D or 2D sections are usually visualized.

- The algorithms used for the MCMC and to invert the Dirac operator are both iterative. Therefore it is interesting to visualize how information propagates, step by step, across the lattice in order to understand the local convergence of these algorithms.
- Because of the size of the data structures involved, Lattice QCD algorithms are implemented as tightly-coupled parallel programs written in C/C++. Visualization can help monitor the communication patterns and identify bugs and network problems.
- Lattice QCD requires knowledge of multiple disciplines and therefore it has a very steep learning curve. The visualization of actual computations can serve a didactic purpose thus fostering a better intuitive understanding of QCD and pushing scientific progress.

### 3 Implementation

The main architecture of our system, fig. 1, is comprised of two main parts: a set of projection operators implemented in C/C++ as parallel MPI programs and a graphical interface based on the Enthought Workbench [wor, 2007] and Mayavi 2.0 [Ramachandran, 2001], which implements a Python interface to VTK [Schroeder et al., 2000]. We will refer to the former as a projection plug-in as opposed to the visualization plug-ins provided by Mayavi.

An example of a projection plug-in is a parallel algorithm that reads gauge configurations, cools it, computes the topological charge in 4 dimensions, takes a time slice and saves it as 3D scalar field in the VTK file format.

An example of a visualization plug-in is an algorithm that reads the above VTK file and generates iso-surfaces.

The Workbench, fig. 2, provides a GUI for the entire system and allows users to interactively manipulate the VTK files: display, rotate, edit, save them in some other standard graphical formats, including JPG, PNG, and VRML.

Mayavi is a Python interface to VTK and allows scripting of the above operations. A typical script would loop over a large set of fields, process each

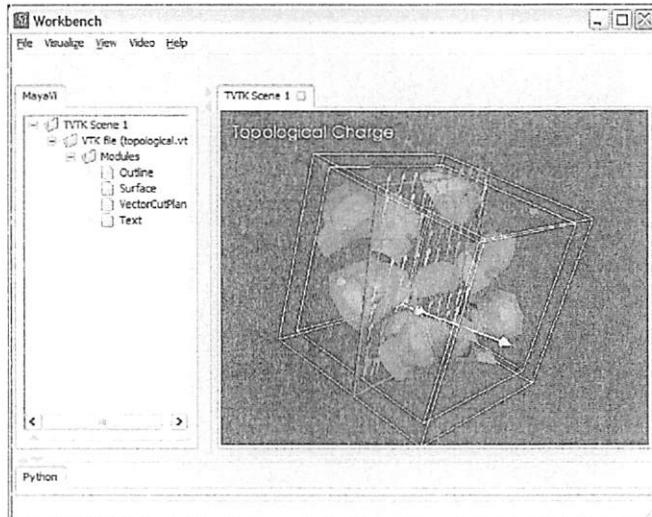


Figure 2: A screenshot of the Enthought Workbench showing the topological charge for a time-slice of a 4D gauge configuration.

of them using the same plug-ins and produce the individual frames as an animation.

Independently on the set of plug-ins used we identified three general recurrent patterns:

- Given one configuration, project and visualize the different time-slices.
- Given one set of configurations and one time-slice  $t$ , project and visualize the same time slice for each configuration.
- Given a set of configurations, for each time-slice, project the time-slice on each configuration, average over all configurations and visualize the average time-slice.

In order to remove visual unpleasant effects of high-frequency quantum fluctuations, when necessary, we adopted the following gauge-invariant smooth-

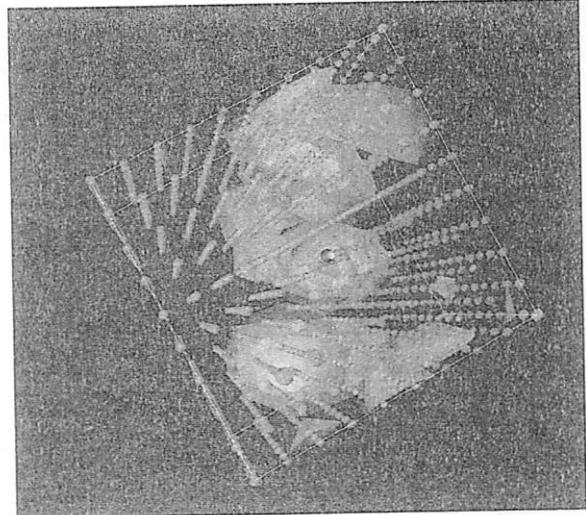


Figure 3: The figure shows iso-surfaces for the topological charge for a time-slice of small test gauge configuration. The spheres show the lattice geometry.

ing algorithm for the gauge configuration:

$$U_{x,\mu,i,j} = \mathcal{P}[\xi U_{x,\mu,i,j} + (1 - \xi) \sum_{n,m} U_{x,\nu,i,m} U_{x+\nu,\mu,m,n} U_{x+\mu,\nu,j,n}^*] \quad (3)$$

$$+ U_{x-\nu,\nu,m,i}^* U_{x-\nu,\mu,m,n} U_{x+\mu-\nu,\nu,n,j}] \quad (4)$$

Here  $x \pm \mu$  are the coordinates of a lattice point shifted from  $x$  in direction  $\pm \mu$ ,  $\xi$  is an arbitrary smoothing parameter, and  $\mathcal{P}$  is a projection operator into the  $SU(3)$  group.

#### 4 Examples and Conclusions

This project started about six months ago. We have successfully produced a set of prototype projection plug-ins that compute topological charge, wave functions, energy density, and density of heatbath hits. Examples of images are shown in figs. 3 and 4.

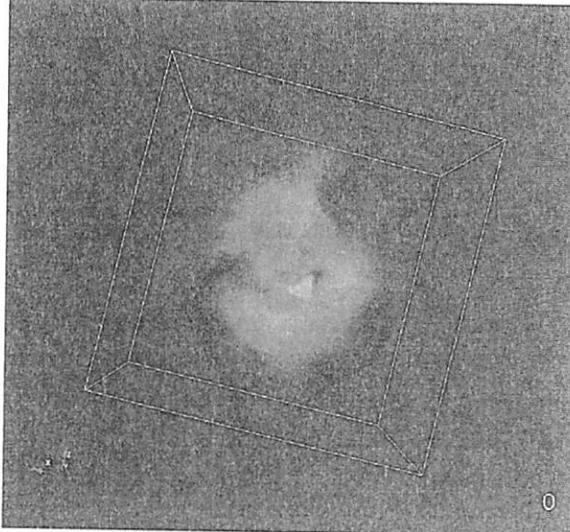


Figure 4: Wave function of a heavy-light meson computed on a single gauge configuration.

The field of visualization of Lattice QCD data is still in its infancy, but has a large potential impact for the physics community. It will help us better understand local properties of the algorithms and the spatial characteristics of extended physical objects such as mesons, hadrons and glue-balls. We also strongly believe that visualization is important to better explain what Lattice QCD is and to attract more students to this exciting area of research.

## 5 Acknowledgments

This project is supported by the Department of Energy under the Scientific Discovery Through Advanced Computing grant DE-FC02-06ER41441. We wish to thank the Fermilab theory group for continuing to share their data and their computing resources with the author. We also thank Dr. P. Ramachandran for developing Mayavi and his email support to this project. We finally thank the DePaul students involved in the project: Joel McGraw,

## References

- [wor, 2007] (2007). Workbench. Available from: <http://www.enthought.com>.
- [apeNEXT Collaboration, 2003] apeNEXT Collaboration (2003). The apenext project. *Nuclear Physics B - Proc. Suppl.*, 119, 1038–1040.
- [Bhanot et al., 2005] Bhanot, G. et al. (2005). Qcd on the bluegene/l supercomputer. *Nuclear Physics B - Proc. Suppl.*, 140, 823–825.
- [Brower, 2006] Brower, R. (2006). National software infrastructure for lattice quantum chromodynamics. *Journal of Physics - Conf. Series*, (pp. 142).
- [FERMILAB-Pub-01/197, ] FERMILAB-Pub-01/197. *B Physics at the Tevatron: Run II and Beyond*. Technical report.
- [Gottlieb, 2006] Gottlieb, S. (2006). Guest editor's introduction: Special purpose computers. *IEEE: Computing in Science and Engineering*, 8, 15.
- [Gottlieb, 2002] Gottlieb, S. (2002). *Nuclear Physics B - Proc. Suppl.*, 106-107, 1031–1033.
- [Holmgren, 2005] Holmgren, D. J. (2005). Pc clusters for lattice qcd. *Nuclear Physics B - Proc. Suppl.*, 140, 183–189.
- [Marciano & Pagels, 1978] Marciano, W. & Pagels, H. (1978). Quantum chromodynamics. *Physics Reports*, 36, 135.
- [Pierro, 2002] Pierro, M. D. (2002). Fermiqcd: A toolkit for parallel lattice qcd applications. *Nuclear Physics B - Proc. Suppl.*, 106-107, 1034–1036.
- [Pierro, 2006] Pierro, M. D. (2006). An algorithmic approach to quantm field theory. *International Journal of Modern Physics A*, 21(3).
- [Ramachandran, 2001] Ramachandran, P. (2001). Technical report.
- [Schroeder et al., 2000] Schroeder, W. J., Avila, L. S., & Hoffman, W. (2000). Visualizing with vtk: A tutorial. *IEEE Computer Graphics and Applications*, 20(5), 20–27.

THE JOURNAL OF

# PERFORMANCE MEASUREMENT®

THE PERFORMANCE MEASUREMENT RESOURCE

VOLUME 12 : NUMBER 1

INSIDE THIS ISSUE - FALL 2007

- PERFORMANCE MEASUREMENT FOR PENSION FUNDS
- MULTI-CURRENCY ATTRIBUTION - PART I  
THE REAL NATURE OF MULTI-CURRENCY RETURNS
- THE JOURNAL INTERVIEW - JONATHAN BOERSMA
- EDITORIAL VIEWPOINT -  
A REPORT ON SETTING PERFORMANCE PRESENTATION STANDARDS
- A HIERARCHY OF METHODS FOR CALCULATING RATES OF RETURN
- ANALYSIS OF RANKING FACTORS FOR A RISK AVERSE INVESTOR IN A  
NON-GAUSSIAN WORLD
- A BRINSON MODEL ALTERNATIVE:  
AN EQUITY ATTRIBUTION MODEL WITH ORTHOGONAL RISK CONTRIBUTIONS

# Analysis of Ranking Factors for a Risk Averse Investor in a Non-Gaussian World

*In this paper the authors discuss the relations between measures of risk, utility functions, and ranking factors for portfolio selection. They prove an exact equivalence between Sharpe, Sortino, Omega, Kappa, and Stutzer rankings in the case of Gaussian distributions. They also derive an exact “corrected” Levy ranking formula that applies to a portfolio with non-Gaussian distributions when the investment amount is predetermined. All the results presented in this paper have been proven by explicit analytical calculations. For brevity and clarity, details of those calculations are omitted from the paper.*

## Massimo Di Pierro, Ph.D.

*is an expert in Numerical Algorithms and applications to Scientific Computing and Computational Finance. He is the author of numerous articles in Physics and Finance and is currently an assistant professor in the School of Computer Science Telecommunications and Information Systems at DePaul University, and he is the director of the Master of Science in Computational Finance program (offered jointly by the School of Computer Science and the College of Commerce). Massimo is also a board member of MetaCryption LLC, a financial software development and consulting company. Massimo has a Ph.D. in Physics from the University of Southampton in U.K.*

## Jack Mosevich, Ph.D.

*joined Merrill Lynch in 1986 after several years as a professor of Mathematics and Computer Science. His main areas of expertise are quantitative finance, risk management, and derivatives analytics. Jack has held senior management positions at UBS Global Asset Management, Harris Investment Management, and Merrill Lynch, as well as being a partner in a hedge fund and fund-of-funds. He is currently a clinical professor of Finance at DePaul University and has been a part-time instructor in the University of Chicago Program on Financial Mathematics since its inception in 1997. Jack possesses a B.Sc. in Mathematics from the University of Illinois, an M.Sc. in Mathematics from Northern Illinois University, and a Ph.D. in Mathematics from the University of British Columbia. His current research is in areas associated with hedge fund risk analytics and risk budgeting.*

## INTRODUCTION

In this paper we discuss the relations between measures of risk, utility functions, and ranking factors for portfolio selection. Our focus is on portfolio selection and not on portfolio construction; therefore we identify a portfolio with its distribution of returns (Ortobelli *et.al.*, 2005). A portfolio may represent a single asset or other investment opportunity. We also distinguish two main cases: in Case #1 the investor makes a portfolio selection before deciding the quantity of money to allocate into the portfolio; in Case #2 the investor makes a portfolio selection after a fixed amount of money has been allocated for the investment. Although these two cases appear very similar they are not.

In Case #1 we conclude that, for portfolios with Gaussian distributed returns, the ranking schemes known as Sharpe, Sortino, Kappa, Omega, and Stutzer are all appropriate and equivalent because they produce

the same relative rankings. For non-Gaussian distributed returns, the above rankings are not equivalent and correspond to different (and subjective) definitions of risk.

In Case #2 we conclude that, for portfolios with Gaussian distributed returns, only the Levy ranking is correct, and it corresponds to being rational and risk averse. The use of Sortino, Kappa, Omega, and Stutzer, in Case #2, would correspond to not being risk averse. For non-Gaussian distributed returns we have derived a “corrected” Levy ranking formula (Levy and Markowitz, 1979) that incorporates the effects of skewness and kurtosis of the distribution in the ranking measure:

$$[return] + m/2[risk] - m/6[risk][skewness] + \\ m/720[risk][kurtosis] \\ (“corrected” Levy), \quad (1)$$

*[kurtosis]* here is the reduced kurtosis and it is zero for

a Gaussian,  $m$  is the risk aversion parameter of the CARA utility function).

### CASE #1

As pointed out by H. Markowitz (Markowitz 1952), in Case #1, the investor can make a selection without having to choose a utility function, although he must make a choice on how to measure risk. The choice of a utility function is necessary only to decide how much to allocate into the selected portfolio and how much to allocate into a risk-free asset (for example in the bank or in a U.S. Treasury bill). In Case #1 the investor would select the optimal portfolio by maximizing the ratio:

$$\frac{[\text{return}] - [\text{benchmark}]}{[\text{risk}]} \quad (2)$$

(Sharpe ranking)

Here  $[\text{return}]$  is the average return or the expected average return for the portfolio;  $[\text{benchmark}]$  is the risk-free rate. In a world where all portfolios are characterized by a Gaussian distribution of returns, the most common measure of  $[\text{risk}]$  is the standard deviation

$$[\text{risk}] = \text{mean of } (x[I] - [\text{return}]) \quad (3)$$

(Risk)

where  $x[I]$  are historical returns. In such a world the above ranking formula takes the name of Sharpe ratio (Sharpe 1964). The Sharpe ratio is the correct ranking scheme in Case #1 for Gaussian portfolios. After the investor has selected the optimal portfolio, the investor

is free to distribute the available funds between this portfolio and the risk-free asset. This requires a second choice. On a risk-return plane the set of available choices is represented by the Capital Allocation Line that passes through a point corresponding to the risk-free asset and the point corresponding to the optimal portfolio.

The figure shows a set of portfolios and a risk-free asset. In Case #1 the investor has to choose one portfolio and then combine it with the risk-free asset (dashed line).

This second choice is subjective and depends on the investor's own utility function, i.e., how to translate return (or wealth) into satisfaction. According to utility theory (Neumann 1947, Nash 1950) an investor who makes choices compatible with a utility function is defined as "rational." An investor who makes choices compatible with a monotonic increasing utility function is defined "rational" and "risk averse." A common monotonic increasing utility function is the Constant Absolute Risk Aversion (CARA) utility function.

### Ranking Factors

Various authors have proposed other measures of risk, for example downside risk

$$[\text{downside risk}] = \text{mean of } (x_i - [\text{benchmark}]) \text{ for } x_i < [\text{benchmark}] \quad (4)$$

(Downside Risk)

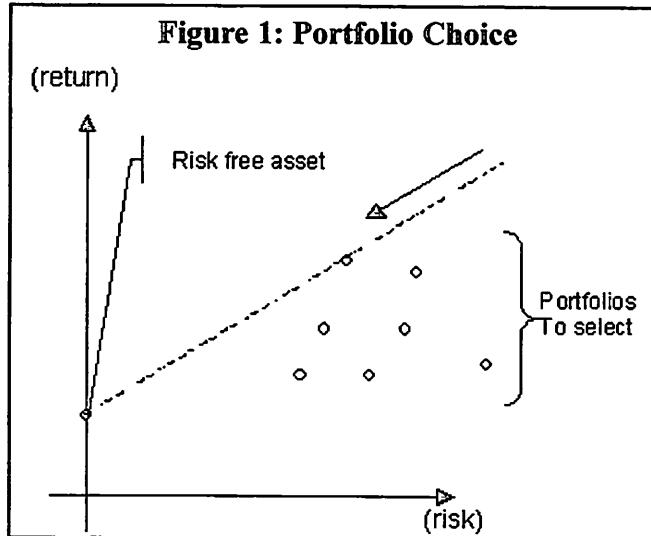
With this definition of risk the Sharpe ratio becomes the Sortino ratio (Sortino and von der Meer 1991).

We say that two ranking schemes are equivalent if and only if, for any set of portfolios, they generate the same relative ranking. In mathematical terms, two rankings,  $R_1$  and  $R_2$ , are equivalent (DiPietro and Mosevitch 2004) if there is a monotonic increasing function  $h$  such that for every portfolio  $A$  the following relation holds:

$$R_1(A) = h(R_2(A)) \quad (5)$$

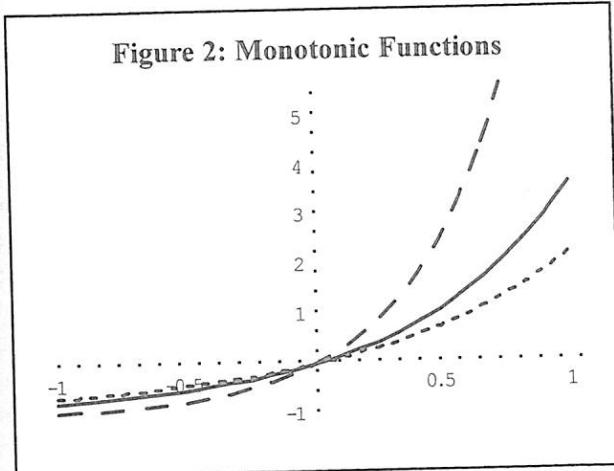
(Equivalence Relation)

The continuous line represents the value of the Sortino ratio as function of the Sharpe ratio for Gaussian portfolios; the large dashed line represents the value of the



Omega ratio as a function of the Sharpe ratio for Gaussian portfolio; the small dashed line represents the value of the Kappa ratio ( $n=3$ ) as function of the Sharpe ratio for Gaussian portfolios. Notice that Kappa at  $n=1$  is equivalent to Omega and Kappa at  $n=2$  is equivalent to Sortino by definition. Notice that all these functions are monotonic and therefore correspond to equivalent rankings in the Gaussian case.

We have proven that in a Gaussian world, Sharpe (Sharpe 1964), Sortino (Sortino and Price 1994) (Sortino and Forsey 1996), Kappa (Kaplan and Knowles 2004), Omega (Sortino 2001, Kazemi *et.al.*, 2003), and Stutzer (Stutzer 2000), are all equivalent ranking schemes. In practical terms, in this case, this implies that if portfolio A and portfolio B are characterized by two Gaussian distributions and if  $\text{Sharpe}(A) > \text{Sharpe}(B)$  then  $\text{Sortino}(A) > \text{Sortino}(B)$ ,  $\text{Kappa}(A) > \text{Kappa}(B)$ ,  $\text{Omega}(A) > \text{Omega}(B)$ , and  $\text{Stutzer}(A) > \text{Stutzer}(B)$ . This is not surprising since Gaussian distributions depend on a single parameter: the width of the Gaussian distribution. The equivalence is shown in Figure 2. In the case of non-Gaussian distributions the ranking schemes are no longer equivalent because they follow from different definitions of risk. For a review see (Artzner and Delbaen, 2000).



## CASE #2

Case #2 is different since a predetermined amount of money has to be invested in the selected portfolio. The two choices of Case #1 are now combined in a single choice and it requires a utility function. As pointed out by Levy and Markowitz, in a world where all portfolios are characterized by a Gaussian distribution, a rational risk averse investor using CARA would rank the portfolios using

$$[\text{return}] - m/2 [\text{risk}]^2 \quad (\text{Levy Ranking}) . \quad (6)$$

Here  $m$  is a subjective risk aversion factor of order one. The larger  $m$ , the more  $[\text{risk}]$  is penalized. In this paper we refer to the latter ranking formula as Levy ranking.

In mathematical terms, given a utility function  $U$ , a portfolio distribution  $P$ , and a ranking  $R$ , we say that  $R$  corresponds to (or is compatible with)  $U$  if and only if there is a monotonic increasing function  $h$  such that

$$\int U(x)P(x)dx = h(R(P)) \quad (\text{Correspondence Relation}) . \quad (7)$$

In other words, a ranking  $R$  corresponds to a utility function  $U$  if and only if the weighted average of the utility  $U$  of all possible returns of portfolio  $P$  generates the same relative ranking as  $R(P)$ .

We have computed the above integral for various utility functions and we have been able to establish the following relations (See Table 1).

We discuss each of these relations one by one.

### Theta Utility Function

An investor who uses the Sharpe ratio to rank portfolios

**Table 1: Portfolio/Utility Function Relations**

Portfolio	Ranking Scheme	Utility Function
Gaussian	Sharpe, Sortino, Kappa, Omega, Stutzer	$\theta$ -function
Gaussian	Levy	CARA
Non-Gaussian	“corrected” Levy	CARA
Non-Gaussian	Levy	CARA*
Non-Gaussian	Levy using [downside risk]	CARA**

in order to allocate a predetermined investment amount (our Case #2) is implicitly using the  $\theta$  utility function to make his selection. The  $\theta$ -function is always equal to +1 for positive returns and always equal to -1 for negative returns. This means that the investor values positive returns more than negative returns, but the investor does not value +100% more than +1%, nor -100% less than -1%. Therefore the investor who uses exclusively the Sharpe ratio to make his decision is rational but is not a risk averse investor.

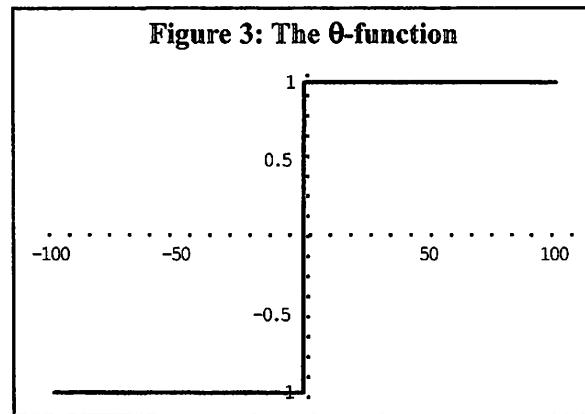
The same is true for Sortino, Kappa, Omega, and Stutzer by virtue of the Equivalence Relation discussed above for Gaussian portfolios.

The x-axis shows return in percent, and the y-axis shows the corresponding utility. The  $q$ -function associates to all positive returns the same utility +1 and all negative returns the same utility -1. Using the Sharpe ratio (as well as Omega, Sortino, Kappa, or Stutzer) for portfolio selection, in the Case #2 discussed in the paper, is equivalent to implicitly adopting the above utility function. Therefore such investor is not a risk averse inverse. The investor does not value a loss of -100% less than a loss of -1 percent.

#### The CARA Utility Function

For Gaussian portfolios, the use of the Levy ranking corresponds to the use of the CARA utility function. For non-Gaussian portfolios, this correspondence is broken by the presence of skewness and kurtosis in the portfolio distribution.

Interestingly it is possible to correct the Levy ranking to correct for skewness and kurtosis, and we obtain the



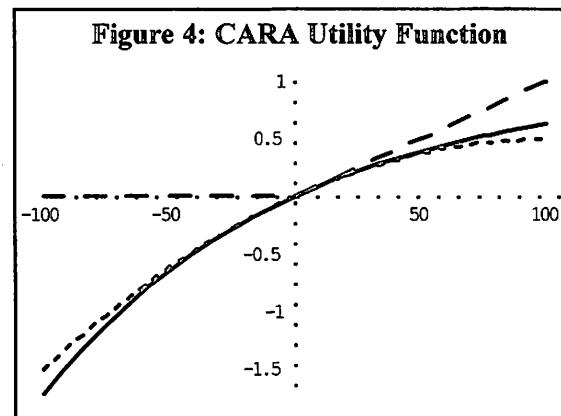
"corrected" Levy ranking formula

$$[return] + m/2[risk]^2 - m^2/6 [risk]^3 [skewness] + \\ m^3/720 [risk]^4 [kurtosis] \\ ("corrected" Levy). \quad (8)$$

([kurtosis] here is the reduced kurtosis, and it is zero for a Gaussian.) This formula follows from the Correspondence Relation by using CARA and expanding the integrand in Taylor series to the 4<sup>th</sup> order.

#### The CARA\* Utility Function

As an exercise we considered a new utility function that we called CARA\*. It consists of an approximation to CARA at second order. Figure 4 below shows that CARA\* approximates very well CARA for returns in range (-100%, +100%).



The continuous line shows the CARA utility function, the x-axis shows return in percent, and the y-axis shows the corresponding utility. The small dashed line shows the CARA\* utility function, defined as an approximation to CARA correct at the second order in Taylor. The large dashed line shows the CARA\*\* utility function, which is equivalent to CARA\* for negative returns but linear for positive returns. These three utility functions correspond to being risk averse. CARA and CARA\* are very similar for practical purposes, while CARA\*\* tends to overvalue positive returns, and thus undervalue risk, when compared with CARA or CARA\*. Counter intuitively CARA\*\* corresponds to using downside risk in place of standard deviation for [risk] in the Levy ranking formula - [return] - m/2 [risk].

The use of the Levy ranking without correction, in the

case of non-Gaussian distributions, corresponds to the implicit choice of the CARA\* utility function.

### The CARA\*\* Utility Function

As another exercise we asked which utility function would correspond to the use of the Levy ranking if the measure of [risk] were replaced by [downside risk]. We found that, counter intuitively, this ranking scheme corresponds to what we call CARA\*\* utility function, i.e., a utility function that is a quadratic approximation for CARA for negative returns and linear for positive returns. In other words CARA\*\* is very close to CARA for negative returns (losses) but weights positive returns (gains) more than CARA does. Therefore the substitution of [risk] with [downside risk] in the Levy ranking has the effect of giving a premium to large potential gains instead of penalizing large positive losses as naively expected.

### CONCLUSIONS

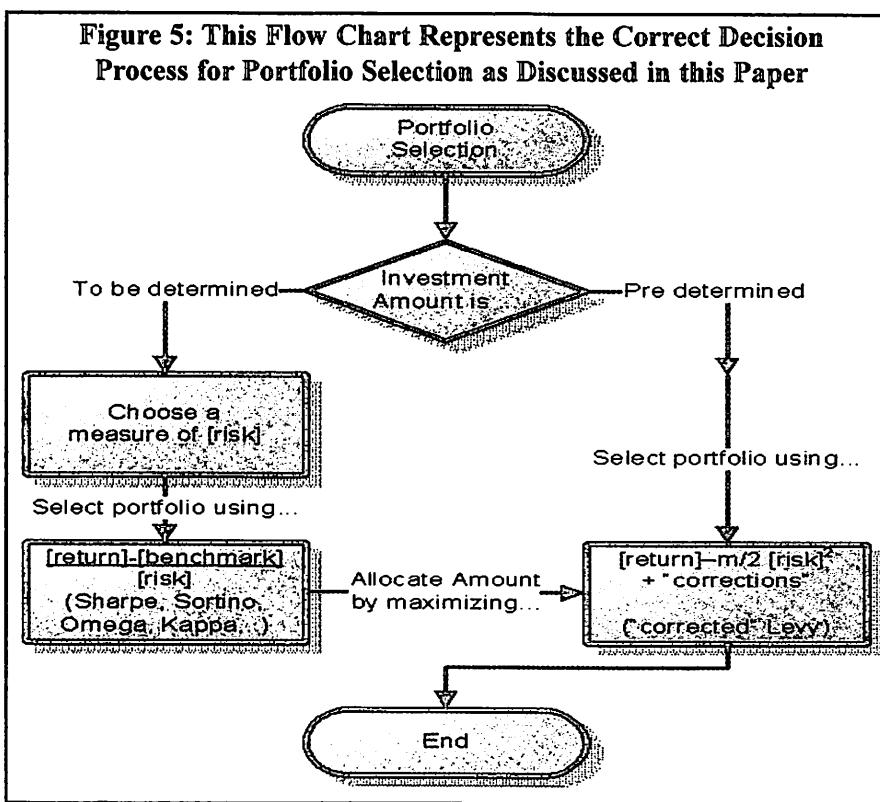
In this paper we have studied the issue of portfolio ranking for portfolio selection. We can summarize our conclusions in the block diagram of Figure 5.

We have distinguished two main cases: when the investment amount is to be determined and one is free to distribute funds between the selected portfolio and the risk-free asset; when there is a predetermined amount that has to be invested in the selected portfolio. In the former case, if the portfolios exhibit a Gaussian distribution, then it is correct to rank them using the Sharpe ratio (Sortino, Omega, Kappa or Stutzer would produce the same ranking). If portfolios do not have a Gaussian distribution, the above ranking schemes correspond to different definitions of risk. In the second case (for a predetermined amount), none of those ranking schemes are appropriate. For Gaussian distributions, a rational risk averse investor using the CARA utility function would rank portfolios using the Levy formula. For non-Gaussian distributions he should use our "corrected" Levy formula:

$$[return] + m/2[risk]^2 - m^2/6[risk]^3 [skewness] + \\ m^3/720[risk]^4 [kurtosis] \\ ("Corrected" Levy) \quad (9)$$

### REFERENCES

Artzner, Philippe, and Freddy Delbaen, Jean-Mark



- Eber, and David Heath, "Coherent Measures of Risk," *Mathematical Finance* 9, 2000, pp. 203-228.
- Di Pierro, Massimo, and Jack Mosevich, "On Ranking Schemes and Portfolio Selection," Hedge Funds and Investment Management, I. Nelken Ed., Butterworth-Heinemann 2004.
- Kaplan, Paul D., and James A. Knowles, "Kappa: A Generalized Downside Risk-Adjusted Performance Measure," *The Journal of Performance Measurement*, Spring 2004, Vol. 8 No. 3, pp. 42-54.
- Kazemi, Hossein, Thomas Schneeweis, and Raj Gupta, "Omega as Performance Measure," CISDM 2003 Proceedings.
- Levy, Haim, and Harry M. Markowitz, "Approximating Expected Utility by a Function of Mean and Variance," *American Economic Review* 69, 1979, pp. 308-317.
- Ortobelli, Sergio, Svetlozar T. Rachev, Stoyan Stoyanov, Frank Fabozzi, and Almira Biglova, "The Proper Use of Risk Measures in Portfolio Theory," *International Journal of Theoretical and Applied Finance*, Dec. 2005.
- Harry M. Markowitz, "Portfolio Selection," *Journal of Finance* 1952, 7 (1), pp.77-91.
- Nash Jr., John F., "The Bargaining Problem," Econometrica 18:155 1950.
- Sharpe, William F., "Capital Asset Prices: a Theory of Market Equilibrium Under Conditions of Risk," *Journal of Finance*, 1964, 19(3), pp. 425-442.
- Sortino, Frank A. and Robert Van Der Meer, "Downside Risk," *Journal of Portfolio Management*, 1991, v17(4), 27-32.
- Sortino, Frank A. and Lee N. Price, "Performance Measurement In a Downside Risk Framework," *Journal of Investing*, 1994, v3(3), pp. 59-64.
- Sortino, Frank A., and Hal J. Forsey, "On The Use and Misuse of Downside Risk," *Journal of Portfolio Management*, 1996, v22 (2,Winter), pp. 35-42.
- Sortino, Frank A., "From Alpha to Omega," in Managing Downside Risk in Financial Markets, Frank A. Sortino and Stephen E. Satchell, eds., Reed Educational and Professional Publishing Ltd., 2001.
- Stutzer, Michael, "A Portfolio Performance Index," *Financial Analysts Journal*, Vol. 56, May-June 2000.
- von Neumann, John, and Oskar Morgenstern, Theory of Games and Economic Behavior. Princeton, NJ. Princeton University Press. 1944 sec.ed. 1947.

# The decay constants $f_B$ and $f_{D^+}$ from three-flavor lattice QCD

C. Bernard<sup>a</sup>, C. DeTar<sup>b</sup>, M. Di Pierro<sup>c</sup>, A.X. El-Khadra<sup>d</sup>, R.T. Evans<sup>d</sup>, E. Freeland<sup>e,i</sup>, E. Gamiz<sup>d</sup>, Steven Gottlieb<sup>f</sup>, U.M. Heller<sup>g</sup>, J.E. Hetrick<sup>h</sup>, R. Jain<sup>d</sup>, A.S. Kronfeld<sup>i</sup>, J. Laiho<sup>ia</sup>, L. Levkova<sup>b</sup>, P.B. Mackenzie<sup>i</sup>, D. Renner<sup>j</sup>, J.N. Simone<sup>\*i</sup>, R. Sugar<sup>k</sup>, D. Toussaint<sup>j</sup>, and R.S. Van de Water<sup>i</sup>

<sup>a</sup>Department of Physics, Washington University, St. Louis, Missouri, USA

<sup>b</sup>Physics Department, University of Utah, Salt Lake City, Utah, USA

<sup>c</sup>School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, Illinois, USA

<sup>d</sup>Physics Department, University of Illinois, Urbana, Illinois, USA

<sup>e</sup>Liberal Arts Department, The School of the Art Institute of Chicago, Chicago, Illinois, USA

<sup>f</sup>Department of Physics, Indiana University, Bloomington, Indiana, USA

<sup>g</sup>American Physical Society, One Research Road, Box 9000, Ridge, New York, USA

<sup>h</sup>Physics Department, University of the Pacific, Stockton, California, USA

<sup>i</sup>Fermi National Accelerator Laboratory, Batavia, Illinois, USA

<sup>j</sup>Department of Physics, University of Arizona, Tucson, Arizona, USA

<sup>k</sup>Department of Physics, University of California, Santa Barbara, California, USA

E-mail: simone@fnal.gov

## Fermilab Lattice and MILC Collaborations

We present new preliminary results for the leptonic decay constants  $f_B$  and  $f_{D^+}$  determined in  $2 + 1$  flavor lattice QCD at lattice spacings  $a = 0.09, 0.12$  and  $0.15$  fm. Results are obtained using the MILC Collaboration gauge configuration ensembles, clover heavy quarks in the Fermilab interpretation and improved staggered light quarks. Decay constants, computed at partially quenched combinations of the valence and sea light quark masses, are used to determine the low-energy parameters of staggered chiral perturbation theory. The physical decay constants are found in an extrapolation using the parameterized chiral formula.

PoS(LATTICE 2007) 370

*The XXV International Symposium on Lattice Field Theory  
July 30-4 August 2007  
Regensburg, Germany*

---

\*Speaker.

## 1. Introduction

The  $D$  meson decay constants, when compared to precise experimental results, are a critical check of the lattice methods needed for  $f_B$ . In Ref. [1] we predicted  $f_{D^+} = 201 \pm 3 \pm 17$  MeV in good agreement with the CLEO-c measurement  $f_{D^+} = 223 \pm 17 \pm 3$  MeV revealed days later [2].

In this work we present new results for the  $D$  and  $B$  meson decay constants. Precise determinations of  $f_B$ ,  $f_{B_s}$  and the ratio  $f_{B_s}/f_B$  are needed to study the Standard Model picture of  $B$ - $\bar{B}$  and  $B_s$ - $\bar{B}_s$  mixing. A progress report for the mixing matrix element study is presented in Ref. [3].

## 2. Simulation details

We use the MILC Collaboration three-flavor asqtad ensembles [4]. Details are tabulated in Table 1. For these ensembles,  $m_l$  denotes the mass of the two degenerate lighter sea quarks. A single heavier sea quark has a mass  $m_h$  near the strange quark mass. Upsilon spectroscopy tells us the heavy quark potential scale  $r_1 = 0.318(7)$  fm [5]. The number of valence quark masses,  $\#m_q$ , used in this study is listed in the last column of the table.

The leptonic decay constant  $f_{H_q}$  for a meson  $H_q$  is defined by

$$\langle 0 | A_\mu | H_q(p) \rangle = i f_{H_q} p_\mu. \quad (2.1)$$

The combination  $\phi_{H_q} = f_{H_q} \sqrt{m_{H_q}}$  emerges from a combined fit to lattice 2-pt functions:

$$C_O(t) = \langle O_{H_q}^\dagger(t) O_{H_q}(0) \rangle \quad (2.2)$$

$$C_{A_4}(t) = \langle A_4(t) O_{H_q}(0) \rangle, \quad (2.3)$$

where  $O_{H_q}$  can be either a smeared or local operator.

The axial current renormalization is taken to be

$$Z_{A4}^{Qq} = \rho_{A4}^{Qq} \sqrt{Z_{V4}^{QQ} Z_{V4}^{qq}}. \quad (2.4)$$

$a$ [fm]	$am_h$	$am_l$	$\beta$	$r_1/a$	configs	$\# m_q$
0.09	0.031	0.0031	7.08	3.69	435	11
		0.0062	7.09	3.70	557	10
		0.0124	7.11	3.72	518	8
0.12	0.05	0.005	6.76	2.64	529	12
		0.007	6.76	2.63	833	12
		0.01	6.76	2.62	592	12
		0.02	6.79	2.65	460	12
		0.03	6.81	2.66	549	12
0.15	0.0484	0.0097	6.572	2.13	631	9
		0.0194	6.586	2.13	631	9
		0.029	6.600	2.13	440	9

**Table 1:** MILC three-flavor lattice parameters. The last column lists the number of valence light quarks used in this study.

Factors  $Z_{V_4}^{ff}$  are fixed nonperturbatively from scattering 3-pt functions and the known normalization of the vector current. Factors  $\rho_{A4}^{Qq}$  are known to one-loop order and are close to unity [6].

### 3. Staggered Chiral Perturbation Theory ( $S\chi$ PT)

With staggered quarks the (squared) taste-nonsinglet pseudoscalar meson masses are split:

$$M_{ab,\xi}^2 = (m_a + m_b)\mu + a^2 \Delta_\xi , \quad (3.1)$$

where  $m_a, m_b$  are quark masses and the (sixteen) mesons are labeled by their taste representation  $\xi = P, A, T, V, I$  with  $\Delta_P = 0$ .

At next-to-leading order (NLO) in  $\chi$ PT the expression for the decay constants is

$$\phi_{H_q} = \Phi_H [1 + \Delta f_H(m_q, m_l, m_h) + p_H(m_q, m_l, m_h)] \quad (3.2)$$

where  $\Delta f_H$  denotes the “chiral logs” and  $p_H$  denotes terms analytic in the meson masses.

With staggered quarks

$$\Delta f_H = -\frac{1+3g_{H^*H\pi}^2}{2(4\pi f_\pi)^2} [\bar{h}_q + h_q^I + a^2 (\delta'_A h_q^A + \delta'_V h_q^V)] . \quad (3.3)$$

Taste-breaking effects arise at finite  $a$  from the meson mass splittings and the  $\delta'_A$  and  $\delta'_V$  hair-pin terms [7]. Finite  $a$  effects reduce the chiral logarithm curvature, however, the expected QCD chiral logarithm is recovered in the continuum limit.

The NLO analytic terms are

$$p_H = \frac{1}{2(4\pi f_\pi)^2} [p_1(m_l, m_h) + p_2(m_q)] \quad (3.4)$$

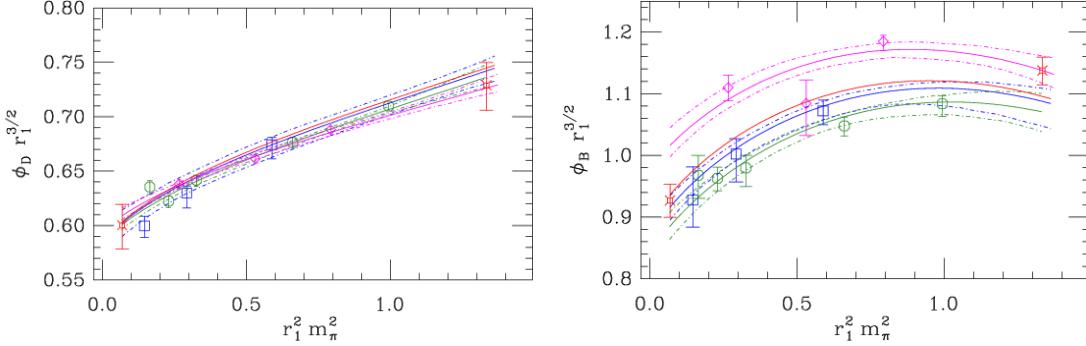
$$p_1 = f_1(\Lambda_\chi) \left[ \frac{11}{9}\mu(2m_l + m_h) + a^2 \left( \frac{3}{2}\bar{\Delta} + \frac{1}{3}\Delta_I \right) \right] \quad (3.5)$$

$$p_2 = f_2(\Lambda_\chi) \left[ \frac{5}{3}\mu m_q + a^2 \left( \frac{3}{2}\bar{\Delta} - \frac{2}{3}\Delta_I \right) \right] , \quad (3.6)$$

where  $\bar{\Delta}$  is the weighted average of taste splittings. The  $O(a^2)$  terms ensure that dependence upon the chiral logarithm scale,  $\Lambda_\chi$ , in  $f_1$  and  $f_2$  cancels that of  $\Delta f_q$ .

Equation (3.2) with the addition of four NNLO analytic terms parameterizes our chiral extrapolations. We fit  $\phi_{H_q}$  to determine the parameters. Constraints (value and width) for  $\mu$ ,  $\Delta_\xi$ ,  $f_\pi$ ,  $\delta'_A$  and  $\delta'_V$  come from  $\chi$ PT for lattice pions and kaons [8]. The coupling  $g_{D^*D\pi}^2 = 0.35 \pm 0.14$  is likewise constrained by the CLEO measurement [9]. From heavy quark symmetry we expect  $g_{B^*B\pi}^2 \approx g_{D^*D\pi}^2$ . The remaining parameters  $\Phi_H$ ,  $f_1$  and  $f_2$  and the NNLO analytic parameters are determined in the fit.

In order to extrapolate to the physical results we set  $\Delta_\xi = \delta'_{A,V} = 0$ ,  $m_h \rightarrow m_s$  and  $m_l \rightarrow (m_u + m_d)/2$ . Then  $\phi_{H_d}$  ( $\phi_{H_s}$ ) is found in the limit  $m_q \rightarrow m_d$  ( $m_s$ ).



**Figure 1:** Chiral fits for the  $D$  (left) and  $B$  (right) mesons. Each fit is viewed along the direction in which  $m_q = m_l$ . Each fit is shown as a set of solid curves with the 68% confidence limits denoted by broken curves. Only statistical errors are shown. The  $a = 0.09$  fm curve and data points are shown in blue, 0.12 fm in green and 0.15 fm in magenta. The  $a^2 \rightarrow 0$  extrapolation curve and the  $\phi_{H_{d,s}}$  points at physical values of  $m_u$ ,  $m_d$  and  $m_s$  are shown in red. The statistical errors on the  $D$  and  $B$  physical points are of comparable size.

#### 4. The Fit and Extrapolation for $D$ and $B$

We determine both  $\phi_{D^+}$  and  $\phi_{B_s}$  from a single fit of  $\phi_{D_q}$  simulation results using the expression in Eqn. (3.2), adding the four NNLO analytic terms and allowing for an explicit  $O(a^2)$  term. We combine simulation results from 11 gauge ensembles at lattice spacings of  $a = 0.09$ , 0.12 and 0.15 fm in the fit. A total of 116 points are included in the fit. A bootstrap procedure propagates errors and correlations among the simulated results through to the statistical errors on our results. An analogous fit procedure for the  $B$  meson simulation results yields  $\phi_{B_d}$  and  $\phi_{B_s}$ .

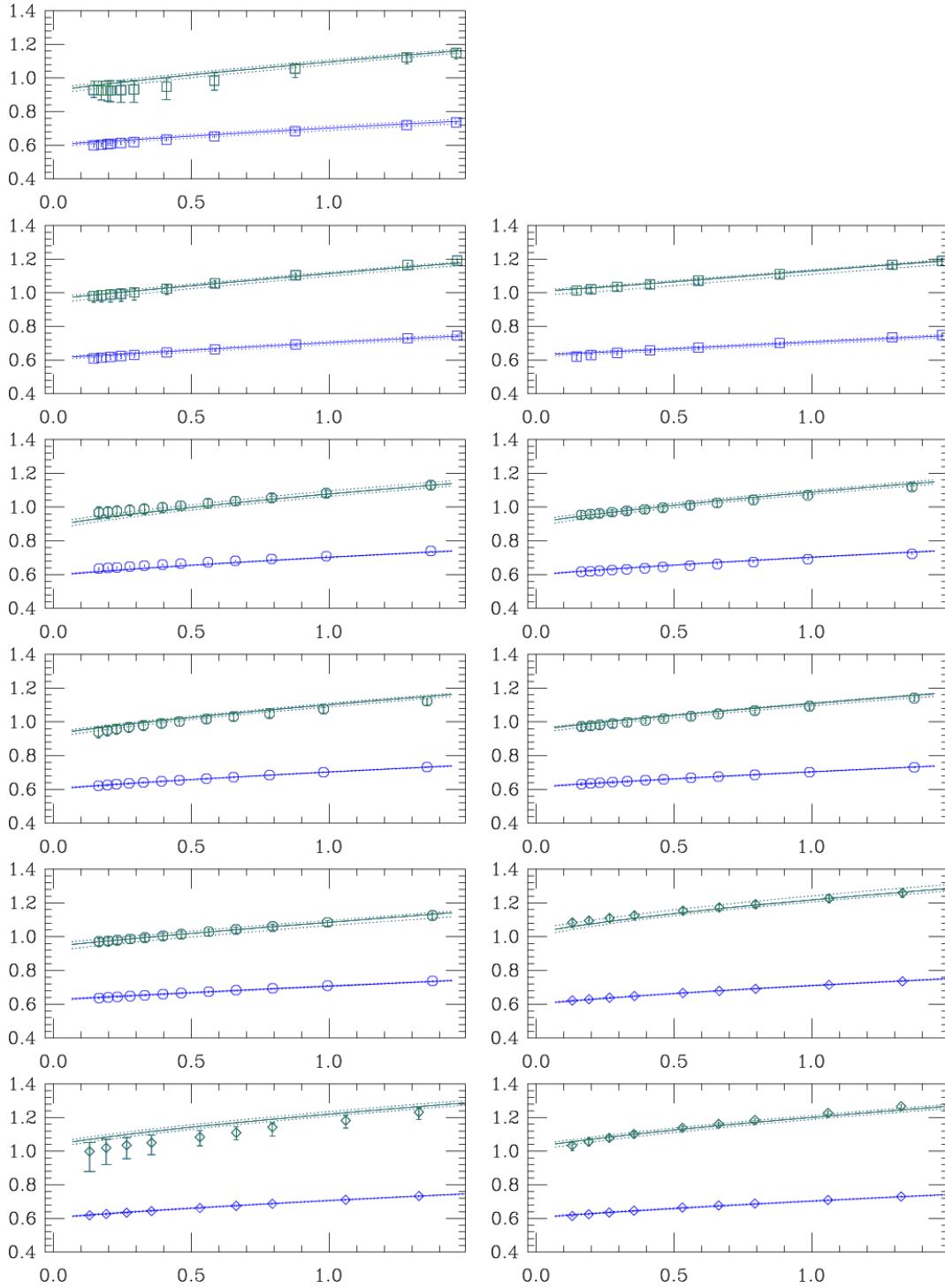
The  $D$  and  $B$  meson fits combining the three lattice spacings are shown in Figs. 1 and 2. Figure 1 shows each fit and the data points along the  $m_q = m_l$  direction while Fig. 2 shows the valence mass dependence of the fit at fixed values of the sea quark mass. All of the fit points are visible in Fig. 2 while only the subset of points with  $m_q = m_l$  is visible in Fig. 1.

Figure 1 shows the  $D$  system on the left and the  $B$  system on the right. In each plot, the solid blue, green and magenta curves are the fit to the lattice data for lattice spacings 0.09, 0.12 and 0.15 respectively. These curves include the  $a^2$  effects described by the chiral fit function. The 68% confidence limits for each curve are indicated by dotted contours of the corresponding color.

In Fig. 2 we show the valence mass dependence the  $D$  and  $B$  systems. In each plot the  $D$  system (blue points and curves) is shown together with  $B$  system (green points and curves). Each plot in the figure corresponds to a single combination of lattice spacing and  $m_l$  from Table 1. Together, the points represent all of the simulation results for  $\phi_{D_q}$  used in this study. The fit curves include the  $a^2$  effects described by the chiral fit function. Each curve is shown with its 68% confidence contours. The expression in Eqn. (3.2) predicts a divergent logarithmic rise as  $m_q \rightarrow 0$  with  $m_l$  fixed. Our fits detect these logarithms even though taste breaking effects obscure them, so they are not immediately obvious in the plots.

The extrapolation,  $a \rightarrow 0$ ,  $m_h \rightarrow m_s$  and  $m_l = m_q$  is shown in Fig. 1 as a solid red curve for each of the  $D$  and  $B$  systems. Our result for  $\phi_{D^+}$  ( $\phi_{B_d}$ ) is found from the extrapolation by setting  $m_l = \hat{m} = (m_u + m_d)/2$  and  $m_q = m_d$ . Likewise,  $\phi_{D_d}$  ( $\phi_{B_s}$ ) is found by setting  $m_q = m_s$ . The physical

POS (LATTICE 2007) 370



**Figure 2:** The  $m_q$  dependence of the single  $D$  fit (in blue) and the single  $B$  fit (in green) at fixed  $m_l$ . The figures ordered from left-to-right and top-to-bottom, have  $am_q$  equal to 0.0031, 0.0062, 0.0124 ( $a = 0.09\text{ fm}$ ), 0.005, 0.007, 0.01, 0.02, 0.03 ( $0.12\text{ fm}$ ), 0.0097, 0.0194 and 0.029 ( $0.15\text{ fm}$ ) respectively. On the y-axis is  $r_1^{3/2} \phi_{Hq}$  and on the x-axis is  $r_1^2 m_\pi^2$ . All 116 points in each fit are shown. The  $\chi^2 = 98.6$  for the  $D$  fit and 48.5 for the  $B$  fit.

quantity	value
$\phi_{D_s}$	0.356(11) GeV $^{3/2}$
$\phi_{D_d}$	0.293(11) GeV $^{3/2}$
$R_{D_{d/s}}$	0.824(8)
$\phi_{B_s}$	0.556(12) GeV $^{3/2}$
$\phi_{B_d}$	0.453(13) GeV $^{3/2}$
$R_{B_{d/s}}$	0.815(15)

**Table 2:** The main results of this preliminary study.

results are indicated with the red burst symbols. Since these points are projected into the  $m_q = m_l$  plane of each figure, the central values do not lie on the red curve. The statistical errors for the physical  $\phi_{H_q}$  values are shown in each figure. We find statistical errors comparable in magnitude for the physical  $\phi_{D_q}$  and  $\phi_{B_q}$  results.

## 5. Results and Outlook

Our preliminary results for the physical  $\phi_{H_q}$  values and ratios with their statistical errors are in Table 2. We tabulate the major sources of uncertainty in Table 3. We omit listing uncertainties arising from terms of order  $1/m_H$  in the chiral extrapolations since adding such terms changes the final results by less than the statistical errors. Such effects are still under investigation. Uncertainties from the input parameters  $r_1$  and the light quark masses are found by propagating the uncertainties found in the MILC  $f_\pi$  and  $f_K$  determinations [5]. We estimate a 3.8% uncertainty in the bare charm mass and a 6.8% uncertainty in the bare bottom mass from variations in tuning procedures for the 0.09 fm lattice. Using simulation results for two heavy-quark masses near both charm and bottom, we estimate the uncertainties in  $\phi_{H_q}$ . The uncertainties in  $Z_V^{ff}$  are statistical. Errors from unknown higher orders in  $\rho_{A_4}$  are estimated by considering higher orders effects to be as large as the 1-loop terms. Heavy quark discretization effects are estimated by power counting arguments. The dominant uncertainty in  $\phi_{H_q}$  comes from effects of order  $\alpha_s \Lambda a \times h(am)$  and  $a^2 \Lambda^2$ , where  $h(am)$  is some mild function of the heavy quark mass. The uncertainties in the ratios are smaller by a factor of  $m_s/\Lambda$ . Light quark discretization effects are estimated by varying the extrapolation procedure. Finite volume effects are estimated by comparing theories at finite volume to the continuum.

From Table 2 and the experimental  $D^+$ ,  $D_s$ ,  $B^0$  and  $B_s$  masses we compute the decay constants:

$$f_{D_s} = 254 \pm 8 \pm 11 \text{ MeV} \quad (5.1)$$

$$f_{D^+} = 215 \pm 8 \pm 11 \text{ MeV} \quad (5.2)$$

$$f_{B_s} = 240 \pm 5 \pm 11 \text{ MeV} \quad (5.3)$$

$$f_{B_d} = 197 \pm 6 \pm 12 \text{ MeV} \quad (5.4)$$

where each of the first errors is statistical. The second error is the systematic error combined in quadrature from Table 3.

We also consider ratios of  $B$  to  $D$  decay constants where statistical and systematic errors are expected to be reduced due to cancellations. Statistical errors in the ratios are from a bootstrap

source	$\phi_{D_s}$	$\phi_{D_d}$	$R_{d/s}$	$\phi_{B_s}$	$\phi_{B_d}$	$R_{d/s}$
statistics	3.1	3.8	1.0	2.1	3.1	1.8
inputs $r_1$ , $m_s$ , $m_d$ and $m_u$	1.4	2.0	0.5	3.1	3.8	0.6
input $m_c$ or $m_b$	2.7	2.7	<0.1	1.1	1.1	<0.1
$Z_V^{QQ}$ and $Z_V^{qq}$	1.4	1.4	0	1.4	1.4	0
higher-order $\rho_{A_4}$	0.3	0.3	<0.2	1.3	1.1	<0.2
heavy quark discretization	2.7	2.7	0.3	1.9	1.9	0.2
light quark discretization	1.0	2.7	1.8	2.0	3.8	1.8
finite volume	0.2	0.6	0.6	0.2	0.6	0.6
total systematic	4.4	5.3	2.0	4.7	6.1	2.0

**Table 3:** The error budget for the decay constants and their ratios. Uncertainties are quoted as a percentage. The total combines systematic errors in quadrature.

procedure in order to preserve statistical correlations.

$$f_{D^+}/f_{D_s} = 0.845 \pm 0.008 \pm 0.017 \quad (5.5)$$

$$f_{B_d}/f_{B_s} = 0.821 \pm 0.015 \pm 0.017 \quad (5.6)$$

$$f_{B_d}/f_{D^+} = 0.919 \pm 0.051 \pm 0.056 \quad (5.7)$$

$$f_{B_s}/f_{D_s} = 0.945 \pm 0.043 \pm 0.043 \quad (5.8)$$

The overall systematic errors for the first two ratios come from Table 3. Systematic errors for the last two ratios also come from combining errors in quadrature. These errors may be overestimates since we have not studied possible correlations at present.

We will extend this study to include a lattice spacing of  $a = 0.06$  fm. We will improve statistics at  $a = 0.09$  and  $0.12$  fm and add another ensemble (sea quark mass combination) at  $0.09$  fm. A finer lattice spacing, more sea quark combinations and better statistics will help control light- and heavy-quark discretization effects and improve statistical errors. The new gauge configurations will be used by MILC to refine  $r_1$  and the light quark masses inputs used in this study.

## References

- [1] C. Aubin *et al.*, Phys. Rev. Lett. **95**, 122002 (2005) [arXiv:hep-lat/0506030].
- [2] M. Artuso *et al.* [CLEO Collaboration], Phys. Rev. Lett. **95**, 251801 (2005) [arXiv:hep-ex/0508057].
- [3] R. T. Evans, E. Gamiz, A. X. El-Khadra and M. Di Pierro, arXiv:0710.2880 [hep-lat].
- [4] C. Aubin *et al.*, Phys. Rev. D **70**, 094505 (2004) [arXiv:hep-lat/0402030].
- [5] C. Bernard *et al.* [MILC Collaboration], PoS **LAT2005**, 025 (2006) [arXiv:hep-lat/0509137].
- [6] A. X. El-Khadra, E. Gamiz, A. S. Kronfeld and M. A. Nobes, arXiv:0710.1437 [hep-lat].
- [7] C. Aubin and C. Bernard, Phys. Rev. D **73**, 014515 (2006) [arXiv:hep-lat/0510088].
- [8] C. Aubin *et al.* [MILC Collaboration], Phys. Rev. D **70**, 114501 (2004) [arXiv:hep-lat/0407028].
- [9] A. Anastassov *et al.* [CLEO Collaboration], Phys. Rev. D **65**, 032003 (2002) [arXiv:hep-ex/0108043].

## Visualization for Lattice QCD

---

**Massimo Di Pierro**

*School of Computer Science, Telecommunications and Information Systems  
DePaul University, 243 S Wabash Ave, Chicago, IL 60604, USA  
E-mail: mdipierro@cs.depaul.edu*

We present a prototype visualization toolkit for Lattice Quantum Chromodynamics. The toolkit consists of a set of parallel algorithms for the computation of the topological charge, energy density, density of heat-bath hits, two and three point correlation functions, that are interfaced with a Graphical User Interface for an interactive visualization of the fields. The toolkit allows both real-time and off-line visualization, scripting for automation, and the ability to combine individual frames into animations (to produce, for example, an animation of the topological charge as function of the MCMC step). The toolkit also includes analysis and plotting tools to automate the typical workflow of Lattice QCD computations.

PoS(LATTICE 2007)031

*The XXV International Symposium on Lattice Field Theory  
July 30-4 August 2007  
Regensburg, Germany*

## 1. Introduction

Typical Lattice QCD simulations generate Tera-bytes of data in the intermediate steps of the computation but the results of each simulation consist of few numbers with a couple of digits of precision. We believe that the ability to visualize the data produced in the intermediate steps of the computations can provide additional value to current and future simulations. On the one hand, data visualization can facilitate the communication of physical concepts and strengthen the outreach effort of the lattice community. On the other hand, it will improve our ability to identify bugs in the algorithms and potential sources of bias in the computations, it will help find pathologies in the convergence properties of the inverters and, perhaps, improve our understanding of Lattice QCD.

Other authors in the past have generated excellent visualizations from Lattice QCD computations [1, 2]. Our project differs from those since it aims to build a comprehensive set of user friendly tools, as opposed to making specific visualizations.

Our toolkit consists mainly of a set of parallel algorithms for Lattice QCD that read typical data files, such as gauge configurations and propagators, and produce various types of projections into 4D gauge invariant scalar fields, for example the topological charge density. The 4D scalar fields are partitioned by time slice and stored together and in parallel in binary VTK files which can be read by most visualization applications including MayaVi, Paraview, VisIt and OpenDX. We have modified the MayaVi2 interface to allow real-time image update and animations. Visualizations and animations can be scripted in Python.

The toolkit also includes an analysis and plotting tool to extract information form log files and to study, both off-line and in real-time, the results of the computation (plot partial averages, sample distributions, moving averages, bootstrap errors, etc.).

Our toolkit is a work in progress and the long term goal is that of developing a comprehensive set of components for automating the workflow of Lattice QCD computations and extract visual information from them.

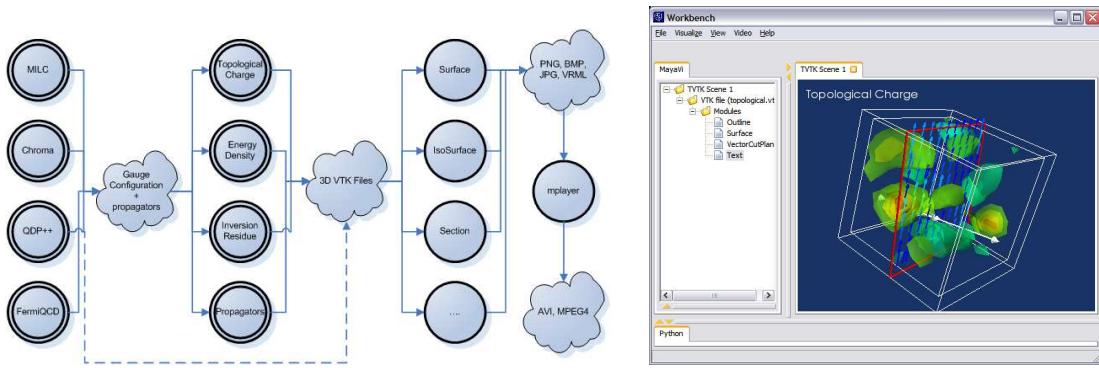
Many of the images shown in this paper, although they are the result of real Lattice QCD computations, should be considered qualitative in nature because their purpose is to explain the capabilities of the toolkit and not present new physics results. For this reason they have been computed on small lattices, relatively to modern state of the art computations; moreover the parameters (quark masses and gauge coupling) have not been tuned to match the physical ones.

The current version of the toolkit, additional images and some movies can be downloaded from <http://mdp.cti.depaul.edu/vqcd>.

## 2. Visualization Tools

The main component of our visuation toolkit is a library of parallel lattice QCD algorithms. Programs using the library are generated using meta-programming techniques and they do not require programming by the user. These programs can read gauge configurations generated by FermiQCD, UKQCD, MILC, Chroma and QDP++ code and apply algorithms in the desired order. For example 1) perform cooling, 2) compute the topological charge; 3) slice and save as VTK file.

VTK, or Visualization Tool Kit, is a standard format for storing large 3D data sets. It supports both structured and unstructured meshes, scalar, vectors and tensor fields. We designed a VTK



**Figure 1:** The figure represents the workflow diagram for our toolkit. The figure on the right is a screenshot of the modified MayaVi2 workbench. Via the workbench it is possible to interact with the visualization in real-time.

based file format to store the most general type of 4-dimensional field. All local field components are stored together and different time-slices are separated but stored in the same file.

In this way it is possible to use any standard visualization application to read the field and use the GUI to browse through the time-slices and the field components interactively. Most visualization applications have a pipeline design that allows to send the data through various filters, smoothing and rendering algorithms. The most common ways to render a 3d scalar field is by means of volume density plots or by iso-surfaces (fig. 1-left).

Different fields and algorithms can then be overlapped, rotated, scaled, projected and sliced interactively. For our toolkit we choose to use the MayaVi2 Workbench (fig. 1-right). This is an open source application written in Python that interacts with VTK via the MayaVi2 API. We modified the workbench and included the ability to loop over multiple LQCD files in order to handle large numbers of files at once (for example loop over all files in a folder, produce one image for each file and then encode the frames into an MPEG animation) and to monitor files for changes (for example to see how a field evolves while a LQCD computation is in progress).

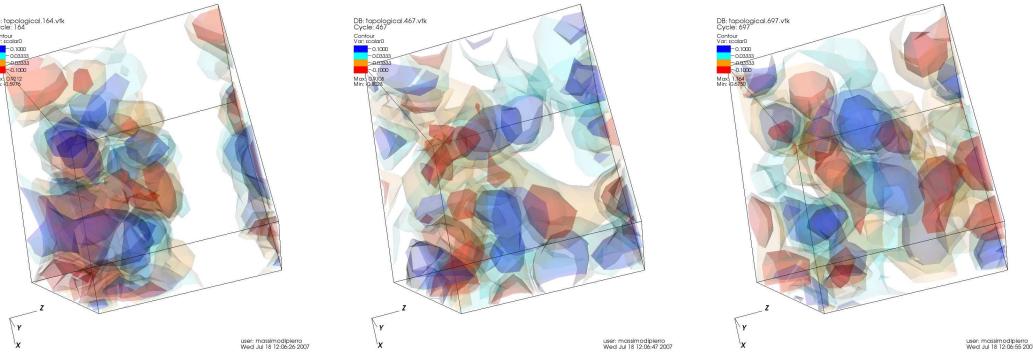
In this paper we reproduce, as examples, screenshots of animations generated by our toolkit.

Fig. 2 shows isosurfaces for the topological charge (instantons) at different steps of the heat-bath Markov Chain. The initial gauge configuration is characterized by half of the lattice in a cold state and half in a hot state. A qualitative but interesting result of this visualization is that instantons do not appear to diffuse from the hot to the cold part of the lattice, instead their density(size) decreases(increases) in the hot half of the lattice and increases(decreases) in the cold half. We adopted the following definition of local topological charge in accord with ref. [3]

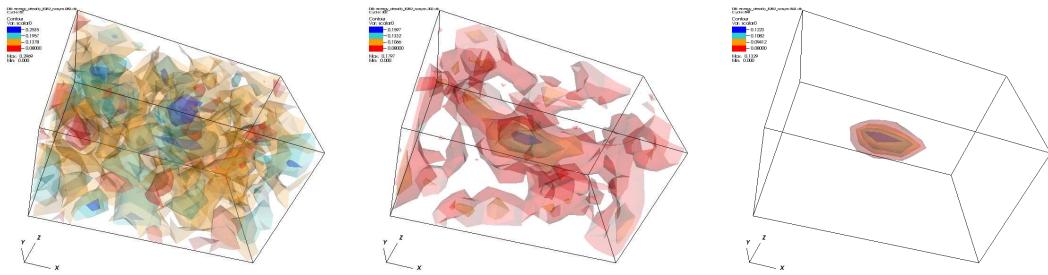
$$Q(x) = \text{Re} \langle F_{\mu\nu}(x) \tilde{F}_{\mu\nu}(x) \rangle \quad (2.1)$$

$F_{\mu\nu}$  is the smeared chromo-electro-magnetic field.

Fig. 3 shows the energy density for a  $SU(2)$  gauge field in presence of a static quark and a static anti-quark as more gauge configurations are averaged. The initial noise (in the first image) dissipates and coherence causes the emergence of the flux tube connecting the quark and the anti-quark (in the third image). Our computation is performed without abelian projection using 1000



**Figure 2:** The figures show isosurfaces for the topological charge (instantons) at different steps of the heathbath Markov Chain. The initial gauge configuration is characterized by half of the lattice in a cold state and half in a hot state. A qualitative but interesting result of this visualization is that instantons do not diffuse from the hot to the cold part of the lattice, instead their denity(size) decreases(increases) in the hot half of the lattice and increases(decreases) in the cold half.



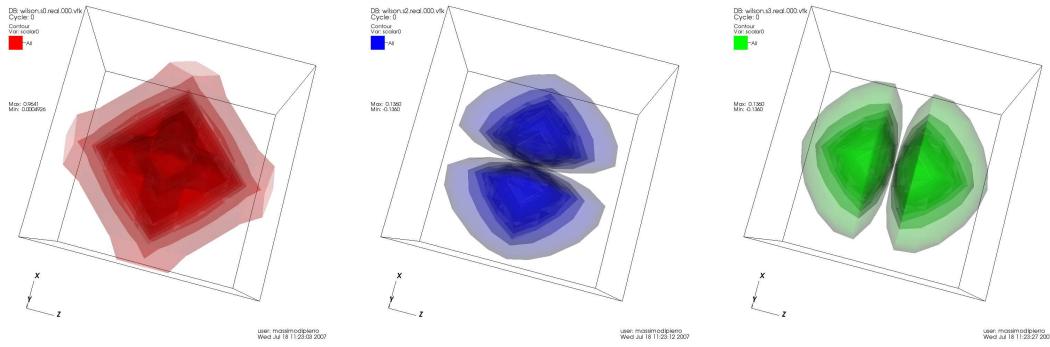
**Figure 3:** The figures show the energy density for a  $SU(2)$  gauge field in presence of a static quark and a static anti-quark as more gauge configurations are averaged. The initial noise (in the first image) dissipates and coherence causes the appearance of the a flux tube connecting the quark and the anti-quark (in the third image).

gauge configurations and the result is qualitatively different from those computations with abelian projection. In fact, our case the peak in the energy density is at the center of the flux tube, while in the abelian projected case there are two peaks where the quarks are localized. We adopted the following definition of energy density, in accord with ref. [4]

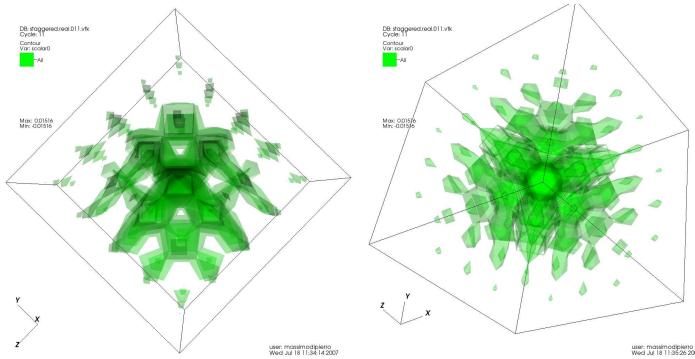
$$V_{\mu\nu}(\vec{x}) = \frac{\langle WP_{\mu\nu}(T/2, \vec{x}) \rangle}{\langle W \rangle} - \langle P_{\mu\nu}(T/2, \vec{x}) \rangle \quad (2.2)$$

$V_{0i} \equiv E_i^2$ ,  $V_{ij} \equiv \epsilon_{ijk} B_k^2$  and  $x = (T/2, \vec{x})$ .  $P$  is a  $SU(2)$  plaquette.  $W$  is the Wilson loop corresponding the static quark and anti-quark.

Fig. 4 shows the only non-zero components of a cold Wilson propagator, respectively  $\psi_{00}$ ,  $\psi_{20}$  and  $\psi_{30}$ , i.e. solution of the Dirac equation  $(\not{D} - m)\psi = \delta$  in absence of background field.  $\delta$  is a delta function localized at the center of the lattice where only spin component 0 and color component 0 are different from zero.



**Figure 4:** The only non-zero components of a cold Wilson propagator, respectively the real parts of  $\psi_{00}$ ,  $\psi_{20}$  and  $\psi_{30}$ , i.e. solution of the Dirac equation  $(\not{D} - m)\psi = \delta$  in absence of background field.



**Figure 5:** A cold staggered propagator, viewed from a face and viewed from corner of the lattice respectively..

Fig. 5 shows a cold staggered propagator, viewed from a face and viewed from corner of the lattice respectively. Because of symmetries of the action, the view is identical from and face and any corner.

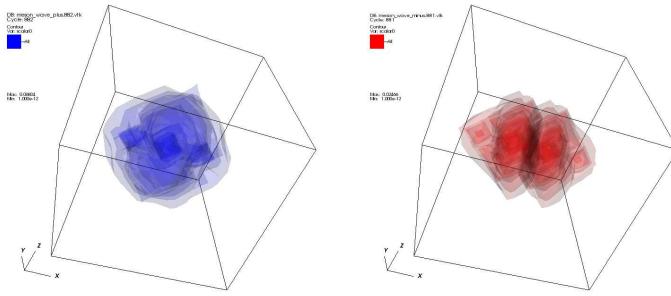
Fig. 6 shows symmetric and anti-symmetric components of the light (valence) quark in a heavy-light meson. The resulting wave function is the QCD analogous of the chemical orbital for the hydrogen atom. They are computed by inserting a  $\bar{q}q$  operator, at different locations in space, in between creation and annihilation operators for a B meson

$$f_\alpha(\vec{x}) = \langle B(+t) | \bar{q}_\alpha(\vec{x}) q_\alpha(\vec{x}) | B(-t) \rangle \quad (2.3)$$

Here  $B(t) = \bar{h}(t, \vec{0}) \gamma^5 q(t, \vec{0})$ .  $h$  is a static quark.

### 3. Analysis Tools

Some of the critical components of many Lattice QCD computations are the Monte Carlo average, the study of convergence of the average, the estimation of the bootstrap error and analysis of the distribution of bootstrap samples. This process is repeated for many quantities, then final



**Figure 6:** Symmetric and anti-symmetric components of the light quark in a B meson.

results are aggregated, fitted, and various types of plots are produced. Our analysis tool automates this workflow.

First, our system is provided with log (output) files containing various quantities measured on the individual gauge configurations. As an example, we consider log files that look like this:

```
2pt[0]=1.01232634
2pt[1]=0.33123125
...
3pt[0][0]=4.23452345
3pt[0][1]=5.52348454
...
```

for various combinations of the indices and for each gauge configurations. (The log files can also contain additional text, such as comments, that are of no interest here.)

Second, our system is provided with an expression like<sup>1</sup>

```
"3pt[<t1>][<t2>]" / ("2pt[<t1>]" * "2pt[<t2>]" )
```

The sub-expressions in quotes are the patterns to look for in the log files, the variables between  $<\dots>$  are defined by the expression itself and are matched with the patterns. The rest is just a mathematical formula, function of the sub-expressions in quotes.)

Third, by clicking one button, the system does everything else:

- Matches all occurrences of “2pt[...]” and “3pt[...]” in the logfiles as function of  $t_1$  and  $t_2$ .
- For each occurrence of “2pt[...]” and “3pt[...]” it computes autocorrelations, distributions, moving averages, and generates the corresponding plots.
- It builds all possible instances of the above expression, for example  $3pt[3][4]/(2pt[3] * 2pt[4])$ . For each of them it computes moving average, mean, bootstrap error, distribution of bootstrap samples, and generates the corresponding plots (including bar plots as function of  $t_1$  and  $t_2$ ).

<sup>1</sup>this is a typical operator that appears, for example, when computing the expectation value of an operator sandwiched between two creation/annihilation operators, as in eq.2.3.

- 
- All plots (typically hundreds or thousands of them for each expression) and numerical results can be browsed via a Graphical User Interface
  - Plots are interactive. They can be zoomed over and customized for printing.
  - All bar plots allow fitting and extrapolations, including correlated, constrained, non-linear and bayesian fits using an easy to understand and natural syntax.

#### 4. Conlcusions and Outlook

In this paper we have presented the general design and the current capability of a visualization toolkit for Lattice QCD. It comprises of two parts: a set of parallel algorithms for computing quantities of physical interest and generating corresponding VTK files, and an analysis and plotting tool to automate the workflow of typical computations.

Although all the components shown here are fully functional, this project is still in its infancy. Our aim is to extend its functionality by adding components that may be of interest to physicists. Moreover the toolkit is in the process of being integrated with the broader USQCD software effort.

#### Acknowledgements

This project is supported by a Scientific Discovery through Advanced Computing (SciDAC) grant DE-FC02-06ER41441 from the Department of Energy.

We wish to acknowledge the Fermilab Theory Group for collaborating on this and other related projects. We also wish to thank Vincent Harvey for his substantial and indenspensible contribution in developing this toolkit.

#### References

- [1] D. Leinweber, Proceedings of "Workshop on Light-Cone QCD and Nonperturbative Hadron Physics", Adelaide, Australia, 13-22 Dec 1999. [hep-lat/0004025]
- [2] M. Feurstein *et al.* Nucl. Phys. B (Proc. Supp.) Vol. 53, 1-3 (1997) pp.553-556
- [3] F. Bonnet *et al.* Phys. Rev. D 62, 094509 (2000)
- [4] D. G. Caldi and T. Sterling, Phys. Rev. Lett. 60, 24 (1988) pp.2454

# Analysis and Visualization Tools for Lattice QCD

PoS(LAT2009)038

**Massimo Di Pierro\***

*School of Computing - DePaul University - Chicago, IL - USA*

*E-mail:* mdipierro@cs.depaul.edu

**Yaoqian Zhong**

*School of Computing - DePaul University - Chicago, IL - USA*

*E-mail:* ati\_zhong@hotmail.com

We developed a toolkit for analysis and visualization of QCD data. The analysis tools include a web based application for extracting data in real time from log files, computing autocorrelation, moving averages, bootstrap samples and bootstrap errors from user provided functions, non-linear Bayesian correlated fits, make the relative plots, and facilitate collaborative work. The visualization tools include standard QCD algorithms with the additional ability to save intermediate steps of the computation in parallel in the VTK file format for real time visualization of on-going computations. To demonstrate, we made visualizations showing the convergence of the MinRes inverter for different sources, evolution of the topological charge under different algorithms for dynamical fermions, energy density in space in the presence of static quark-antiquark, and heavy-light wave functions. Our toolkit is self contained and works in cooperation with many existing QCD code.

*The XXVII International Symposium on Lattice Field Theory - LAT2009*

*July 26-31 2009*

*Peking University, Beijing, China*

---

\*Speaker.

## 1. Introduction

Our work consists of building computer programs to facilitate collaboration, analysis and presentation of data for Lattice QCD physicists. We mainly distinguish two classes of such tools:

- A web based application (also available as a self-standing GUI tool) for collaboration, Monte Carlo analysis and 2D visualization of analysis results.
- A collection of parallel algorithms for generating 3D images and animations from Lattice QCD gauge configurations.

We briefly describe each of these tools in the following two sections.

## 2. Analysis and 2D Visualization Tools

In a typical Monte Carlo computation [1] one can distinguish two main phases. In Phase I, a computationally intensive Markov Chain Monte Carlo (MCMC) program generates gauge configurations and, for each configuration, measures multiple correlation functions. In Phase II, the output of the above program is analyzed to produce averages and plots. While Phase II requires less computation than I, it is not less expensive in human time, and the one that requires the most collaboration among people. For this reason we have created a web based tool called *mc4qcd* that performs the following steps:

- It **acquires** data from the users, in the form of log files from phase I. The system does not dictate a file format for these log files. It just requires that they contain multiple measurements for the same quantity (once per gauge configuration) and that each quantity is labeled using a prefix. The prefix may contain an index. For example, the log file may contain lines similar to “ $2pt[4] = 0.121235$ ” or “2-point correlation function at 4: 0.121235”. The log file can contain additional text that will be ignored in the parsing.
- It **parses** the data using regular expressions to extract quantitative information. The regular expressions are provided by the user, can be different for each log file, and may be applied to multiple runs. An example could be: “Extract all 2pt and 3pt correlation functions.”
- It **filters** the data based on user preferences. For example “consider only 2pt correlation functions separated by less than 12 time-slices”.
- It **mines** the data for expressions of interest. For example “compute all possible ratios of  $3pt[< t_1 >][< t_2 >]/(2pt[< t_1 >] \times 2pt[< t_2 >])$ ”.
- It **represents** the data in multiple formats such as text based, plots and histograms. For example “show an histogram of the bootstrap samples for  $3pt/2pt^2$ ”. The various plot types include: raw data, autocorrelations, moving averages, bootstrap samples, averages with bootstrap errors.
- It allows one to further **refine** the data and **interact** with the data by fitting the bootstrap results using Bayesian correlated fits with arbitrary non-linear functions.

The entire process is executed via a browser, it requires login, and enforces access control on all data at every step. This facilitates collaboration between members of a group by allowing them to easily locate and search data and plots. Users can also comment on each other's data (if they have access to it) and the comments can include Latex expressions. In this way, the thought process of the physicist working on the data is stored together with the data in the web application. The underlying libraries can be used to batch script all the functions provided by the web application.

Figure 1 shows some screenshots from the *mc4qcd* program. The top-left screenshot lists the data sets available to a user, links possible analysis algorithms, and analysis results. The plots represent an example of autocorrelation functions for the  $2pt$  at 3 different time slices, the histograms of the bootstrap samples, and a plot of  $2pt[t]$  with an exponential fit as function of t.

Although this program was designed specifically for lattice QCD in mind, it can be used for the analysis and plotting of MCMC data in other areas of Physics.

The program is packaged together with a web server and a file based relational database into a single application that runs on Windows, Mac and Linux. Its only dependencies are the Python interpreter and the matplotlib library.

### 3. 3D Visualization Tools

Typical Lattice QCD computations generate a large quantity of data that is normally only looked at in aggregated form (numbers with a precision of a few digits and simple 2D plots). A lot of the information is discarded. While this is the nature of MCMC computations, we believe there may be a value in looking in more detail at the data being generated in order to identify patterns that may lead to better understanding of the algorithms and/or that may be symptomatic of errors in the computations. Moreover, it will better present scientific results and educate the public about Lattice QCD.

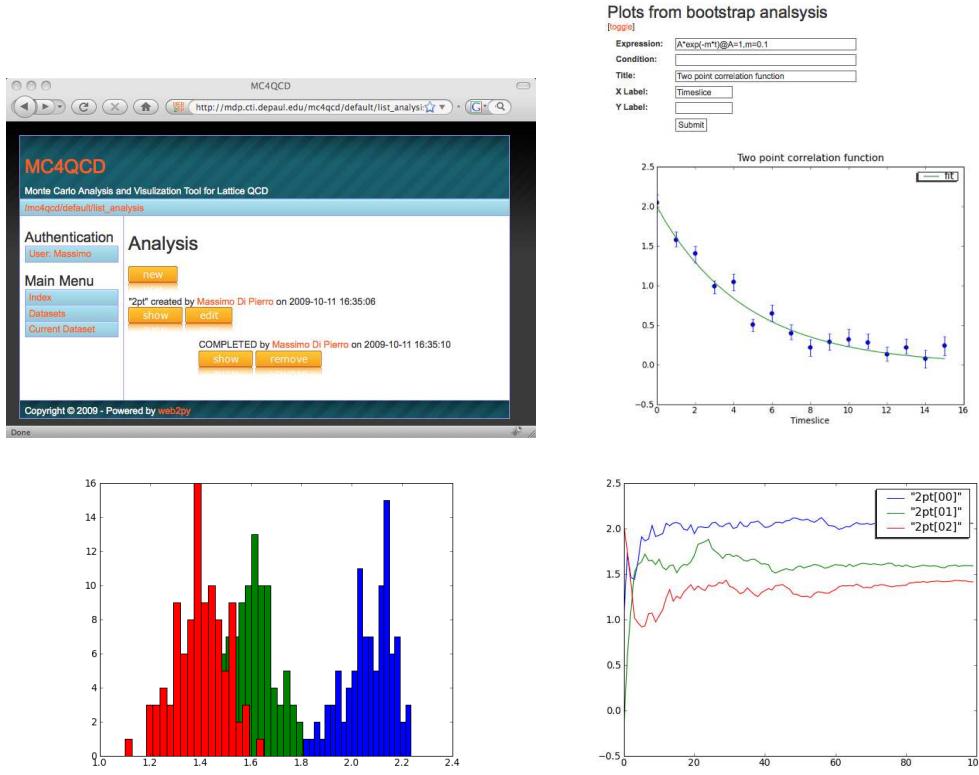
Our toolkit comprises of a set of Lattice QCD libraries for implementing actual computations. These libraries have been modified to be able to save their intermediate state, in parallel, into fields (gauge, fermions, scalars) which can be visualized in 3D interactively (volume plots, iso-surfaces, slices).

Our toolkit can read multiple gauge configurations formats including ILDG, Lime, Nersc (3x2 and 3x3), MILC and FermiQCD [2]. The file format is automatically detected and each file is converted into a FermiQCD format for further analysis.

The output fields (for example the energy density, the topological charge, the component of a propagator) are saved as scalars and VTK files. Each time-slice is saved as a separate scalar field within the same VTK file. The VTK files can be combined using an mpeg-encoder to make movies.

The Visualization Toolkit (VTK) is an open source, platform independent, graphics engine with libraries for parallel rendering of 3D visualizations. VTK is being developed by a large collaboration of universities, laboratories (Sandia, Los Alamos, and Lawrence Livermore), and private companies. VTK provides core functionality for many existing visualization packages such as VisIT (developed by LLNL), MayaVi and Paraview. All these packages are compatible and can interoperate with our software.

Here we present two case studies of usage of our toolkit.

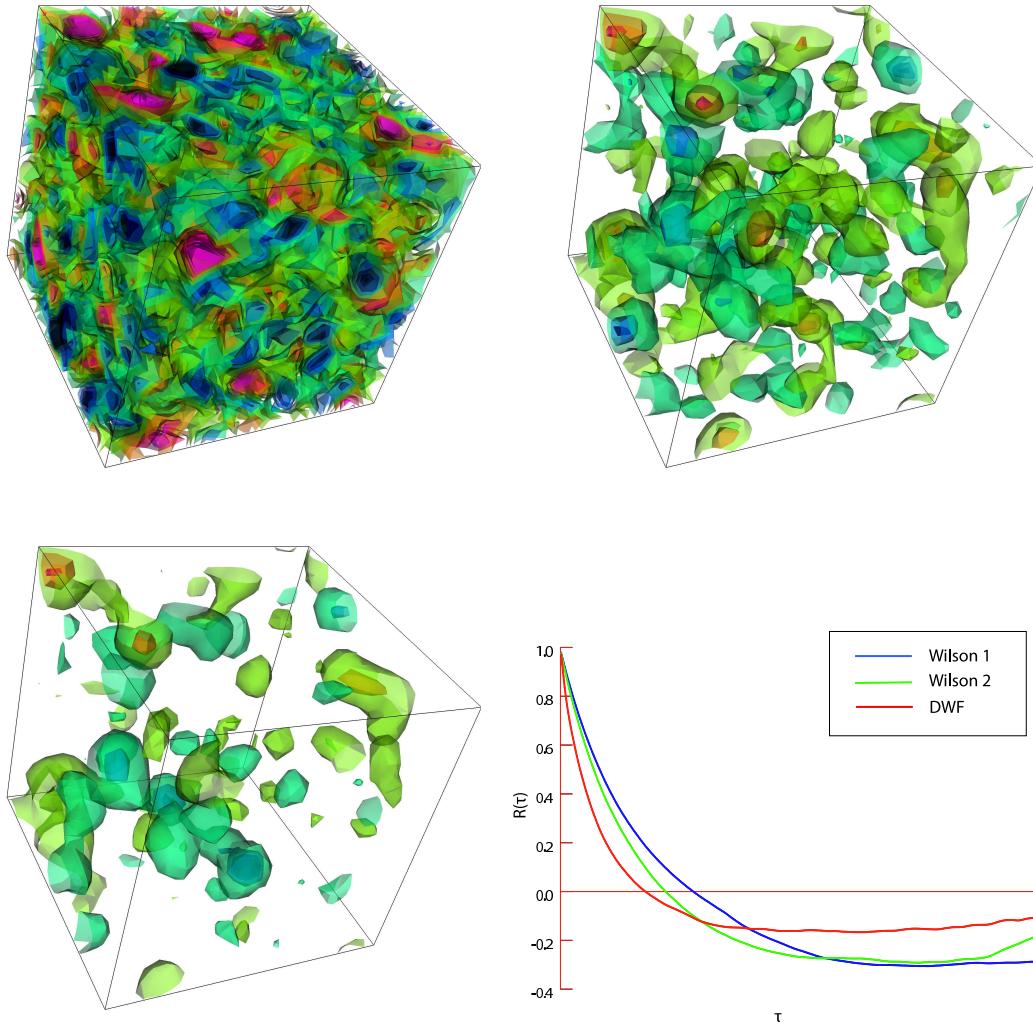


**Figure 1:** The top-left image shows a page of the mc4qcd program. This page allows a user to select a data-set and run one or more analysis algorithms on that data-set. The top-right image shows a typical plot generated by the system for a 2 point correlation function analysis on the database. The images at the bottom show, from left to right respectively, the distribution of bootstrap samples and moving averages for  $2pt[t]$  at three different values of  $t$ . These plots and more are generated automatically by the web application upon upload of a data-set.

### 3.1 Autocorrelation length

In this case study [3] we visualize topological charge density and measure how fast or slow it evolves under the Markov Chain Monte Carlo (MCMC) for different algorithms and different simulation parameters (the dynamic quark mass in particular). This would give us strong confidence that current Lattice QCD computation are adequately sampling the space of all possible configurations as opposed to getting stuck in one topological sector, and provide an effective tool for developing algorithms that maximize equilibration of the topological charge. We analyzed ad hoc gauge configurations generated with small trajectories by Michael Clark and Chulwoo Jung respectively:

- Two ensembles of  $32 \times 24^3$  dynamical Wilson gauge configurations with  $\beta = 5.6$  and quark masses of 66MeV and 33MeV respectively. ( $dt = 0.1$ )
- One ensemble of  $64 \times 24^3$  dynamical Domain Wall gauge configurations with beta=2.13, two degenerate light quarks of 14MeV, and one strange quark of 74MeV. ( $dt = \frac{1}{6}$ ).



**Figure 2:** The images show topological charge density at different cooling steps for a sample gauge configuration. The plot at the bottom-right shows the autocorrelation length for the computed topological change density for the three ensembles of production quality gauge configurations discussed in the paper.

We performed 20 gauge invariant APE smearing steps for cooling, shown in figure 2, computed the topological change, and computed the average autocorrelation of the topological change density.

The result is shown in bottom-right plot. The autocorrelation crosses 0 after about 100 steps. This indicates that the local topological charge becomes de-correlated after about 10 MCMC steps at  $dt = 1$  which is typical for production grade Lattice QCD computations.

### 3.1.1 Inversion Residue

Visualization can be used to represent various types of fields; for example, wave functions, form factors, and/or bare quark propagators. Visualizing the latter can be useful to understand the rate of convergence of the inverters at a local level as opposed to at a global level (as measured by

the total residue). In this example we show how we have modified the MinRes inverter to visualize a quark propagator on a typical production quality gauge configuration:

```
class MinResVtk {
public: static inversion_stats inverter(...) {
    ...
    psi_in.update();
    mul_Q(r,psi_in,U,coeff);
    r*=-1;
    r+=psi_in;
    psi_out=psi_in;
    do {
        r.update(); // sync parallel buffers
        mul_Q(q,r,U,coeff);
        alpha=q*r;
        alpha/=norm_square(q);
        mdp_add_scaled_field(psi_out, alpha, r);
        mdp_add_scaled_field(r, -alpha, q);
        residue=sqrt(norm_square(r)/r.global_size());
        forallsites(x) // project into a scalar
        for(int a=0, s(x)=0; a<4; a++)
            for(int k=0; k<psi_in.nc; k++)
                s(x)+=sqrt(real(psi_out(x,a,k)*conj(psi_out(x,a,k))));
        s.save_vtk(filename,...); // save as VTK file
    } while (...);
    return stats;
}
}
```

Figure 3 show the absolute value of a quark propagator for a planar source (top) and a solid source (bottom), after a few inversion steps (left) and after convergence (right), for the same gauge configuration. The visualization indicates that the inversion algorithm converges uniformly everywhere.

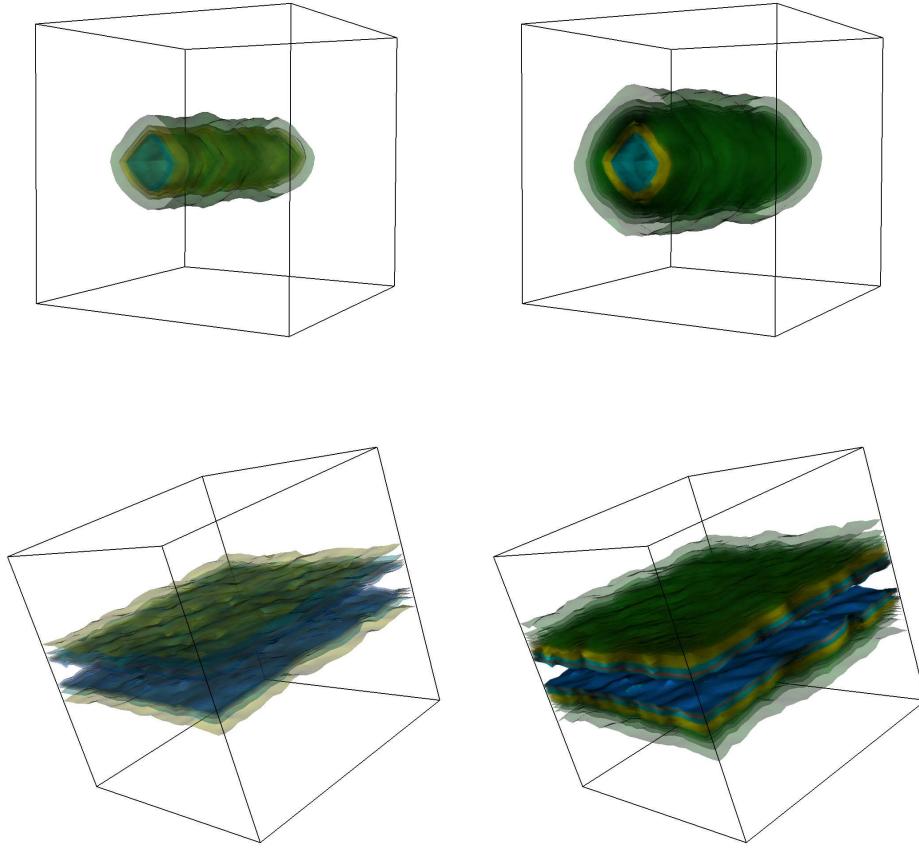
#### 4. Conclusions

We believe visualization will prove to be an important tool for understanding Lattice QCD algorithms and the MCMC evolution. For example the exact nature of the topological objects responsible for confinement is yet to be understood satisfactorily. We also believe visualization can play an important role in education and outreach outside the core of the Lattice QCD community. Our tools are Open Source (GPL2) and can be downloaded from:

<https://launchpad.net/qcdmc>  
<https://launchpad.net/qcd>

#### Acknowledgements

We thank Argonne National Laboratory for proving computing resources Michael Clark (BU) and Chulwoo Jung (BNL) for proving the USQCD gauge configurations used in some of our im-



**Figure 3:** The first three images show the topological charge for a typical gauge configuration at different cooling steps. The plot at the bottom-right shows auto-correlation for the three different ensembles of gauge configurations discussed in the paper.

ages. We also thank James Osborn (ANL), John Negele (MIT), Richard Brower (BU), and Steven Gottlieb (Indiana) for various comments and advice.

This project is funded by the US Department of Energy under grant DEFC02-06ER41441.

## References

- [1] M. Di Pierro, J. Mod. Physics A21 (2006)
- [2] M. Di Pierro *et al.*, Lattice 2003, Nucl. Phys. B Proc. Suppl. (2004)
- [3] M. Di Pierro *et al.*, J. Phys. Conf. Ser. 180 (2009) 012068

# Progress on charm semileptonic form factors from 2+1 flavor lattice QCD

**Jon A. Bailey<sup>\*a†</sup>, A. Bazavov<sup>b</sup>, C. Bernard<sup>c</sup>, C. Bouchard<sup>e</sup>, C. DeTar<sup>d</sup>, A.X. El-Khadra<sup>e</sup>, E.D. Freeland<sup>c</sup>, W. Freeman<sup>b</sup>, E. Gamiz<sup>a,e</sup>, Steven Gottlieb<sup>e,f,g</sup>, U.M. Heller<sup>h</sup>, J.E. Hetrick<sup>i</sup>, A.S. Kronfeld<sup>a</sup>, J. Laiho<sup>c</sup>, L. Levkova<sup>d</sup>, P.B. Mackenzie<sup>a</sup>, M.B. Oktay<sup>d</sup>, M. Di Pierro<sup>j</sup>, J.N. Simone<sup>a</sup>, R. Sugar<sup>k</sup>, D. Toussaint<sup>b</sup>, and R.S. Van de Water<sup>l</sup>**

<sup>a</sup>Theoretical Physics Department, Fermilab, Batavia, IL 60510, USA

<sup>b</sup>Department of Physics, University of Arizona, Tucson, AZ 85721, USA

<sup>c</sup>Department of Physics, Washington University, St. Louis, MO 63130, USA

<sup>d</sup>Physics Department, University of Utah, Salt Lake City, UT 84112, USA

<sup>e</sup>Physics Department, University of Illinois, Urbana, IL 61801, USA

<sup>f</sup>Department of Physics, Indiana University, Bloomington, IN 47405, USA

<sup>g</sup>National Center for Supercomputing Applications, University of Illinois, Urbana, IL 61801, USA

<sup>h</sup>American Physical Society, One Research Road, Ridge, NY 11961, USA

<sup>i</sup>Physics Department, University of the Pacific, Stockton, CA 95211, USA

<sup>j</sup>School of Computing, DePaul University, Chicago, IL 60604, USA

<sup>k</sup>Department of Physics, University of California, Santa Barbara, CA 93106, USA

<sup>l</sup>Department of Physics, Brookhaven National Laboratory, Upton, NY 11973, USA

## The Fermilab Lattice and MILC Collaborations

Lattice calculations of the form factors for the charm semileptonic decays  $D \rightarrow K l \bar{\nu}$  and  $D \rightarrow \pi l \bar{\nu}$  provide inputs to direct determinations of the CKM matrix elements  $|V_{cs}|$  and  $|V_{cd}|$  and can be designed to validate calculations of the form factors for the bottom semileptonic decays  $B \rightarrow \pi l \bar{\nu}$  and  $B \rightarrow K l \bar{\nu}$ . We are using Fermilab charm (bottom) quarks and asqtad staggered light quarks on the 2+1 flavor asqtad MILC ensembles to calculate the charm (bottom) form factors. We outline improvements to the previous calculation of the charm form factors and detail our progress. We expect our current round of data production to allow us to reduce the theoretical uncertainties in  $|V_{cs}|$  and  $|V_{cd}|$  from 10.5% and 11%, respectively, to about 7%.

*The XXVII International Symposium on Lattice Field Theory*

*July 26-31, 2009*

*Peking University, Beijing, China*

---

<sup>\*</sup>Speaker.

<sup>†</sup>jabaily@fnal.gov

## 1. Introduction

The CKM matrix elements  $|V_{cs}|$  and  $|V_{cd}|$  can be extracted to greatest precision (currently to 0.02% and 0.4%, respectively) by assuming CKM unitarity and performing a fit to all data [1]. However, the simplest tests of unitarity require direct determinations of the CKM matrix elements.

The decay rate for  $D \rightarrow K(\pi)l\nu$  is proportional to a form factor and  $|V_{cs}|$  ( $|V_{cd}|$ ). Experiments can measure the decay rates and the form factor shapes, but nonperturbative calculations of the strong force are required to fix the form factor normalizations and extract  $|V_{cs(d)}|$ . Therefore these decays allow direct determinations of  $|V_{cs(d)}|$  and consistency checks between lattice QCD and unitarity. Such consistency increases our confidence in both.

In June CLEO-c published the results of an analysis of  $818 \text{ pb}^{-1}$  collected at charm threshold [2]. Combining the CLEO-c results with the first 2+1 flavor lattice calculations of the  $D \rightarrow K(\pi)l\nu$  form factors [3, 4] yields  $|V_{cs(d)}|$  [2]:

$$|V_{cs}| = 0.985(1 \pm 0.9\% \pm 0.6\% \pm 10.5\%), \quad (1.1)$$

$$|V_{cd}| = 0.234(1 \pm 3\% \pm 0.9\% \pm 11\%). \quad (1.2)$$

The first errors are experimental statistical errors, and the second are experimental systematics. The third errors are due to uncertainties in the lattice QCD calculations. The theory errors dominate the uncertainties.

Discretization effects are the dominant source of the theory errors [3]. Other uncertainties enter because of incomplete suppression of oscillations due to opposite-parity states, truncation effects in fits to staggered chiral perturbation theory ( $S\chi PT$ ), and model-dependence implicit in the Becirevic-Kaidalov (BK) parameterization [3, 5].

These sources of uncertainty were addressed in work on  $B \rightarrow \pi l\nu$  decays [6]. By calculating the  $D \rightarrow K(\pi)l\nu$  form factors using the same methods, we may be able to validate their application to calculations of the form factors for  $B \rightarrow \pi l\nu$  and  $B \rightarrow K l\bar{l}$ . The former decay allows a precise determination of  $|V_{ub}|$  and a stringent test of unitarity. The latter is a rare decay and a prime candidate for new physics. Below we describe our progress in reducing the uncertainties in the charm form factors and anticipate the reduction of the uncertainties in  $|V_{cs(d)}|$ .

## 2. Ensembles and quark masses

To decrease discretization effects and improve our control of the chiral extrapolation, we are generating full QCD and partially quenched data on each of the ensembles shown in Table 1. These ensembles include the four most chiral coarse ensembles used in the calculations of Ref. [3], the two fine ( $a \approx 0.09 \text{ fm}$ ) ensembles included in our recent calculation of the form factor for  $B \rightarrow \pi l\nu$  [6], two additional fine ensembles, three superfine ( $a \approx 0.06 \text{ fm}$ ) ensembles, and one ultrafine ( $a \approx 0.045 \text{ fm}$ ) ensemble [7]. The MILC Collaboration has increased the number of configurations in each of the previously used coarse and fine ensembles by a factor of four, and we expect a corresponding decrease in all statistics-dominated uncertainties by a factor of two.

We have found that randomizing the spatial location of the sources significantly decreases autocorrelations in 2-point functions, which suggests that we may be able to increase our statistics further by increasing the number of source times on each configuration. We have nearly completed

	$\approx a$ (fm)	$am_l/am_s$	Volume	$N_{conf}$	$am_{\text{valence}}$
coarse	0.12	0.02/0.05	$20^3 \times 64$	2052	0.005, 0.007, 0.01,
		0.01/0.05	$20^3 \times 64$	2259	0.02, 0.03, 0.0415,
		0.007/0.05	$20^3 \times 64$	2110	0.05; 0.0349
		0.005/0.05	$24^3 \times 64$	2099	
fine	0.09	0.0124/0.031	$28^3 \times 96$	1996	0.0031, 0.0047, 0.0062,
		0.0062/0.031	$28^3 \times 96$	1946	0.0093, 0.0124, 0.031;
		0.00465/0.031	$32^3 \times 96$	983	0.0261
		0.0031/0.031	$40^3 \times 96$	1015	
superfine	0.06	0.0036/0.018	$48^3 \times 144$	668	0.0036, 0.0072, 0.0018,
		0.0025/0.018	$56^3 \times 144$	800	0.0025, 0.0054, 0.0160;
		0.0018/0.018	$64^3 \times 144$	826	0.0188
ultrafine	0.045	0.0028/0.014	$64^3 \times 192$	861	TBD

**Table 1:** Asqtad staggered quark ensembles generated by the MILC Collaboration [7, 8, 9] and slated for upcoming heavy-light analyses, together with the valence quark masses being used at each lattice spacing. The last valence mass listed at each lattice spacing (after the semicolon) is the tuned strange quark mass. We are presently generating correlators at four source times on each ensemble and investigating the possibility of adding more source times to further increase the total number of source-configurations.

data generation at four source times on the coarse ensembles, the fine ensembles with  $m_l = 0.4m_s$ ,  $0.2m_s$ , and  $0.1m_s$ , and the superfine ensemble with  $m_l = 0.2m_s$ .

Power counting arguments [3, 6] indicate that including these ensembles will effectively eliminate discretization effects due to light quarks and gluons, while heavy-quark discretization effects will be reduced but remain significant. To improve our estimates of heavy-quark discretization effects, we are investigating including them in chiral-continuum expansions [10]. This approach incorporates the information from power counting while more systematically fixing the appropriate hadronic scales.

### 3. Correlators and correlator ratios

The form factors parameterize the hadronic matrix elements of the flavor-changing vector currents,

$$\langle K(\pi)|V_\mu|D\rangle = \sqrt{2m_D} \left[ v_\mu f_{||}^{D \rightarrow K(\pi)}(q^2) + p_{\perp\mu} f_{\perp}^{D \rightarrow K(\pi)}(q^2) \right], \quad (3.1)$$

where  $V_\mu$  is the lattice current corresponding to  $i\bar{s}\gamma_\mu c$  ( $i\bar{d}\gamma_\mu c$ ),  $v = p_D/m_D$  is the four-velocity of the  $D$  meson,  $p_\perp = p_{K(\pi)} - (p_{K(\pi)} \cdot v)v$  is the component of kaon (pion) momentum perpendicular to  $v$ , and  $q^2 \equiv (p_D - p_{K(\pi)})^2$  is the invariant mass of the leptons. We work in the  $D$ -meson rest frame, in which the form factors are proportional to the temporal and spatial components of the hadronic matrix elements, and  $q^2 = m_D^2 + m_{K(\pi)}^2 - 2m_D E_{K(\pi)}$ .

One way to extract the hadronic matrix elements is by considering simple ratios of 3-point to 2-point correlators [3],

$$\frac{C_{3,\mu}^{D \rightarrow K(\pi)}(t, T; \mathbf{p}_{K(\pi)})}{C_2^{K(\pi)}(t; \mathbf{p}_{K(\pi)}) C_2^D(T-t)}, \quad (3.2)$$

where  $T$  is the separation between source and sink in the 3-point functions, and

$$C_{3,\mu}^{D \rightarrow K(\pi)}(t, T; \mathbf{p}_{K(\pi)}) = \sum_{\mathbf{x}, \mathbf{y}} e^{i\mathbf{p}_{K(\pi)} \cdot \mathbf{y}} \langle \mathcal{O}_{K(\pi)}(t_i, \mathbf{0}) V_\mu(t, \mathbf{y}) \mathcal{O}_D^\dagger(t_f, \mathbf{x}) \rangle, \\ t \in [t_i, t_f = (t_i + T) \bmod n_t], \quad (3.3)$$

$$C_2^{K(\pi)}(t; \mathbf{p}_{K(\pi)}) = \sum_{\mathbf{x}} e^{i\mathbf{p}_{K(\pi)} \cdot \mathbf{x}} \langle \mathcal{O}_{K(\pi)}(t_i, \mathbf{0}) \mathcal{O}_{K(\pi)}^\dagger(t, \mathbf{x}) \rangle, \\ t \in [t_i, t_f = (t_i + n_t) \bmod n_t], \quad (3.4)$$

$$C_2^D(t) = \sum_{\mathbf{x}} \langle \mathcal{O}_D(t_i, \mathbf{0}) \mathcal{O}_D^\dagger(t, \mathbf{x}) \rangle, \quad t \in [t_i, t_f = (t_i + n_t) \bmod n_t]. \quad (3.5)$$

where  $n_t$  is the temporal extent of the lattice, and  $\mathbf{p}_{K(\pi)}$  is the momentum of the outgoing kaon (pion). We calculate the correlators for momenta  $\mathbf{p}_{K(\pi)} = (0, 0, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)$ , and  $(2, 0, 0)$  (in units of  $2\pi/L$ , where  $L$  is the spatial extent of the lattice) and all times  $t$  in the ranges shown. We increase statistics by averaging correlators with source times  $t_i = 0, n_t/4, n_t/2, 3n_t/4$ . The  $D$ -meson interpolating operators  $\mathcal{O}$  are smeared with a charmonium wavefunction to suppress coupling to excited states.

$C_3$  is calculated with insertions of the current operator at all times  $t$  between the source and sink. At sufficiently large source-sink separations  $T$  and times  $t$  sufficiently far from both source and sink ( $0 \ll t \ll T$ ), a plateau emerges in the ratio (3.2). This plateau is directly proportional to the desired hadronic matrix element.

In practice we find that oscillations from opposite-parity excited states contaminate the entire plateau region [3, 6]. We therefore consider the more carefully constructed correlator ratios introduced in Ref. [6]:

$$\bar{R}_{3,\mu}^{D \rightarrow K(\pi)}(t, T; q^2) \equiv \frac{1}{\phi_{K(\pi)\mu}} \frac{\bar{C}_{3,\mu}^{D \rightarrow K(\pi)}(t, T; \mathbf{p}_{K(\pi)})}{\sqrt{\bar{C}_2^{K(\pi)}(t; \mathbf{p}_{K(\pi)}) \bar{C}_2^D(T-t)}} \sqrt{\frac{2E_{K(\pi)}}{e^{-E_{K(\pi)}t} e^{-m_D(T-t)}}}, \quad (3.6)$$

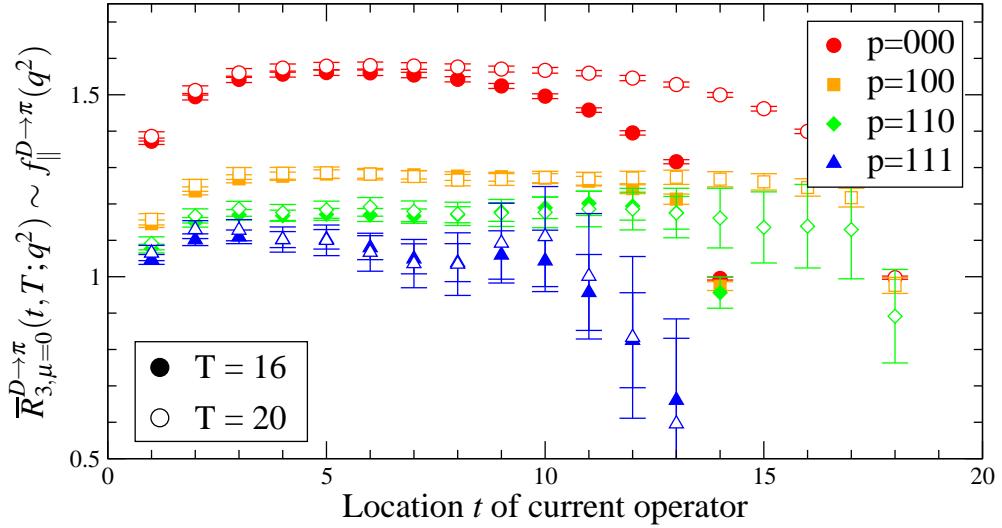
where  $\phi_{K(\pi)\mu} \equiv (1, \mathbf{p}_{K(\pi)})$  and the correlators  $\bar{C}_3, \bar{C}_2$  are constructed from the correlators  $C_3, C_2$  to eliminate oscillations from opposite-parity states:

$$\bar{C}_3(t, T) \equiv \frac{1}{8} \left[ C_3(t, T) + C_3(t, T+1) e^{m_D} + 2C_3(t+1, T) e^{E_{K(\pi)} - m_D} + 2C_3(t+1, T+1) e^{E_{K(\pi)}} \right. \\ \left. + C_3(t+2, T) e^{2(E_{K(\pi)} - m_D)} + C_3(t+2, T+1) e^{2E_{K(\pi)} - m_D} \right], \quad (3.7)$$

$$\bar{C}_2(t) \equiv \frac{1}{4} [C_2(t) + 2C_2(t+1) e^{m_D} + C_2(t+2) e^{2m_D}]. \quad (3.8)$$

Experience suggests that the errors in direct fits to the oscillating states can be larger than errors in simpler fits. The construction of (3.6) and (3.7, 3.8) allows us to fit the ratios to constants without introducing systematic errors. In the plateau region ( $0 \ll t \ll T$ ), the ratios  $\bar{R}_{3,\mu}^{D \rightarrow K(\pi)}$  for  $\mu = 0$  ( $\mu = i$ ) approach the form factors  $f_{||}^{D \rightarrow K(\pi)}$  ( $f_{\perp}^{D \rightarrow K(\pi)}$ ).

For source-sink separations  $T = 16$  and  $T = 20$ , examples of the plateaus are shown in Figs. 1 and 2, where the features leading to the choice of these  $T$ -values can also be seen. As the source-sink separation increases, signal-to-noise decreases. As the source-sink separation decreases, the



**Figure 1:** Ratios of correlators for extracting the form factor  $f_{||}^{D \rightarrow \pi}(q^2)$ . The correlators were calculated on 2110 configurations of the coarse ensemble with  $m_l = 0.14m_s$ .  $T = 16, 20$  are the source-sink separations, and the three-momenta  $p$  of the pions are given in units of  $2\pi/L$ , where  $L$  is the spatial extent of the lattice. Note the excited state contamination in the zero momentum data with  $T = 16$ .

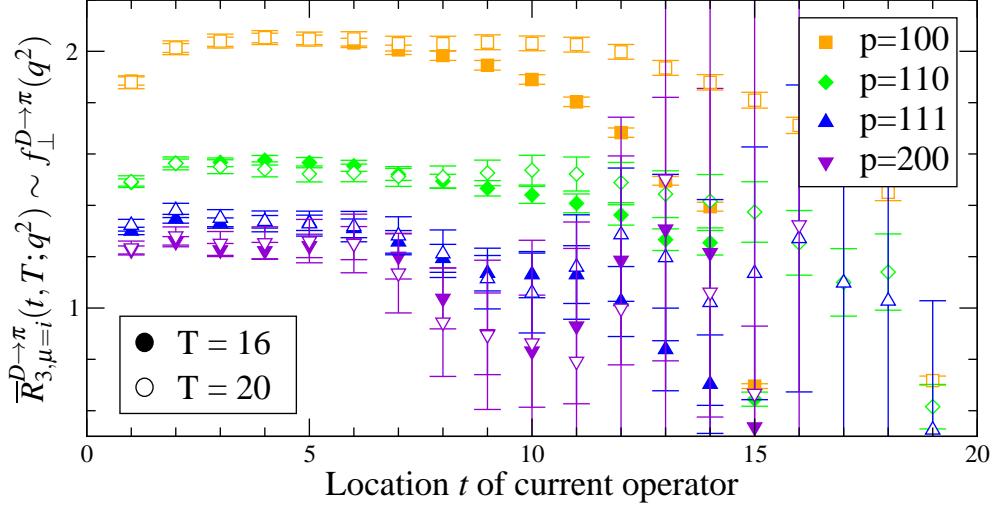
plateau region shrinks and eventually disappears. The optimal  $T$ -value is the smallest for which a plateau exists. For this  $T$ , signal-to-noise is maximized without sacrificing the plateau to excited state contamination. The statistical errors increase with momentum, so the optimal  $T$  is momentum dependent.

To optimize  $T$  we generated data with  $T = 16, 18$ , and  $20$  on the coarse  $m_l = 0.14m_s$  ensemble. As shown in Figs. 1 and 2, for  $T = 20$  plateaus exist for all momenta. At zero momentum, comparing the  $T = 16$  data with the  $T = 20$  data reveals the effects of excited state contamination in the  $T = 16$  data for all  $t$ ; the plateau has essentially vanished. At nonzero momentum, comparing the  $T = 16$  data with the  $T = 20$  data reveals smaller statistical errors in the  $T = 16$  data with intact plateau regions. The larger  $T$  allows checks for excited state contamination at smaller momenta, and the smaller  $T$  allows us to minimize statistical errors at larger momenta. On the remaining ensembles, we expect the optimal  $T$ -value in physical units to be similar. We are therefore generating data on each ensemble with two  $aT$ -values of approximately  $0.12 \text{ fm} \times 16$  and  $0.12 \text{ fm} \times 20$ .

#### 4. Renormalization and chiral-continuum-energy extrapolation-interpolation

Lattice form factors obtained from the plateaus in Figs. 1 and 2 must be renormalized and extrapolated to zero lattice spacing and the physical light quark masses. The renormalization factors can be written as products of non-perturbatively calculable factors  $Z_V$  and perturbatively calculable factors  $\rho$ . The uncertainties in these renormalization factors contribute to the uncertainties in the form factors and CKM matrix elements.

To perform simultaneous chiral-continuum extrapolations and the kaon (pion) energy interpolation, we can use staggered heavy meson partially quenched chiral perturbation theory ( $\chi$ PT)



**Figure 2:** Ratios of correlators for extracting the form factor  $f_\perp^{D\rightarrow\pi}(q^2)$ . The correlators were calculated on 2110 configurations of the coarse ensemble with  $m_l = 0.14m_s$ .  $T = 16, 20$  are the source-sink separations, and the three-momenta  $p$  of the pions are given in units of  $2\pi/L$ , where  $L$  is the spatial extent of the lattice. The consistency of the results for  $T = 16$  and  $T = 20$  indicates that the smaller source-sink separation can be used to minimize statistical errors without introducing significant excited state contamination.

with constrained curve fitting [6, 11, 12]. This approach incorporates the energy-dependence of the form factors and yields a model-independent result while accounting for the systematic error due to truncating the expansion.

To extract  $|V_{cs(d)}|$ , one can divide the experimental results [2] by the lattice form factors evaluated at  $q^2 = 0$ . However, minimizing the uncertainty in  $|V_{cs(d)}|$  requires a simultaneous fit to all (experimental and lattice) data. The analyticity-based parameterization described in Ref. [13] captures the energy-dependence of the form factors throughout the kinematic domains, so using it to fit the data and extract CKM matrix elements does not introduce model-dependent systematic errors.

For  $D \rightarrow K(\pi)l\nu$ , the energy-domains of the lattice and experimental data overlap significantly, allowing a stringent test of the consistency of the shapes of the form factors as determined independently by the lattice and experiment. This test provides important validation for applying the analyticity-based parameterization to the extraction of  $|V_{ub}|$  from  $B \rightarrow \pi l\nu$ , in which the overlap of the lattice data and experimental data is smaller and this self-consistency check, less powerful.

## 5. Expected uncertainties

A projected error budget for the form factors at  $q^2 = 0$  is shown in Table 2. The expected uncertainties reflect previous experience with  $B \rightarrow \pi l\nu$  [6], including the use of improved correlator ratios,  $\chi$ PT with constrained curve fitting, and the analyticity-based parameterization to eliminate systematic errors due to incomplete cancellation of oscillating state contributions, truncation of the chiral expansion, and model-dependence in the BK parameterization. The projections also reflect the four-fold increase in statistics on the coarse ensembles and the addition of the two largest

Stat. + $\chi$ PT	$g_{D^*D\pi}$	$r_1$	$\hat{m}$	$m_s$	$\kappa_c$	$p_\pi$	HQ	$Z_V$	$\rho$	$L^3 < \infty$	Sys.	Total
4.9	2.9	1.4	0.3	1.3	0.2	0.1	3.9	0.7	0.7	0.5	5.4	7.3

**Table 2:** Contributions to the relative uncertainties in the form factors at  $q^2 = 0$  assuming data with four source times on the four extended coarse ensembles, two largest fine ensembles, and the superfine  $m_l = 0.2m_s$  ensemble. The errors are due to limited statistics and the truncation of chiral perturbation theory; uncertainties in the  $D^*D\pi$  coupling, scale, average up-down quark mass, strange quark mass, and charm hopping parameter; momentum-dependent discretization effects from the light quarks and gluons; heavy-quark discretization effects; uncertainties in the renormalization factors  $Z_V$  and  $\rho$ ; and finite volume effects. The last two entries are the total systematics and the total error, both added in quadrature.

fine ensembles and the superfine  $m_l = 0.2m_s$  ensemble. The increase in statistics decreases our statistical uncertainties by a factor of two, while the addition of the superfine ensemble reduces systematic errors due to heavy-quark discretization effects.

Heavy-quark discretization effects and the uncertainty in the  $D^*D\pi$  coupling dominate the systematic uncertainties, while statistics and  $\chi$ PT truncation error are alone comparable to the entire remaining systematic error. Heavy-quark discretization effects are sensitive to the smallest lattice spacings included, so they will decrease further with the addition of the ultrafine ensemble in Table 1. The error due to the  $D^*D\pi$  coupling may respond to the increased statistics. From Table 2 and Eqs. (1.1) and (1.2), we expect to reduce the theoretical uncertainties in the CKM matrix elements from about 11% to about 7%.

Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

## References

- [1] C. Amsler *et al.* [Particle Data Group], Phys. Lett. B **667**, 1 (2008).
- [2] D. Besson *et al.* [CLEO Collaboration], Phys. Rev. D **80**, 032005 (2009) [arXiv:0906.2983 [hep-ex]].
- [3] C. Aubin *et al.* [Fermilab Lattice and MILC Collaborations], Phys. Rev. Lett. **94**, 011601 (2005) [arXiv:hep-ph/0408306].
- [4] C. Bernard *et al.* [Fermilab Lattice and MILC Collaborations], Phys. Rev. D **80**, 034026 (2009) [arXiv:0906.2498 [hep-lat]].
- [5] D. Becirevic and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000) [arXiv:hep-ph/9904490].
- [6] J. A. Bailey *et al.* [Fermilab Lattice and MILC Collaborations], Phys. Rev. D **79**, 054507 (2009) [arXiv:0811.3640 [hep-lat]].
- [7] A. Bazavov *et al.*, arXiv:0903.3598 [hep-lat].
- [8] C. W. Bernard *et al.* [MILC Collaboration], Phys. Rev. D **64**, 054506 (2001) [arXiv:hep-lat/0104002].
- [9] C. Aubin *et al.* [MILC Collaboration], Phys. Rev. D **70**, 094505 (2004) [arXiv:hep-lat/0402030].
- [10] A. Bazavov *et al.* [Fermilab Lattice and MILC Collaborations], PoS(LAT2009)249.
- [11] C. Aubin and C. Bernard, Phys. Rev. D **73**, 014515 (2006) [arXiv:hep-lat/0510088].
- [12] C. Aubin and C. Bernard, Phys. Rev. D **76**, 014002 (2007) [arXiv:0704.0795 [hep-lat]].
- [13] T. Becher and R. J. Hill, Phys. Lett. B **633**, 61 (2006) [arXiv:hep-ph/0509090].

# Quarkonium mass splittings with Fermilab heavy quarks and 2+1 flavors of improved staggered sea quarks

---

**T. Burch,\* C.E. DeTar and L. Levkova<sup>†</sup>**

*Physics Department, University of Utah, Salt Lake City, UT 84112, USA  
E-mail: ludmila@physics.utah.edu*

**M. Di Pierro**

*School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, Illinois, USA*

**A.X. El-Khadra**

*Physics Department, University of Illinois, Urbana, Illinois, USA*

**Steven Gottlieb**

*Department of Physics, Indiana University, Bloomington, IN 47405, USA*

**A.S. Kronfeld, P.B. Mackenzie, and J.N. Simone**

*Fermi National Accelerator Laboratory, Batavia, IL 60510, USA*

We present results from an ongoing lattice study of the lowest lying charmonium and bottomonium level splittings using the Fermilab heavy quark formalism. Our objective is to test the performance of this action on MILC-collaboration ensembles of (2 + 1) flavors of light improved staggered (asqtad) quarks. Measurements are done on 16 ensembles with degenerate up and down quarks of various masses, thus permitting a chiral extrapolation, and over lattice spacings ranging from 0.09 fm to 0.18 fm, thus permitting study of lattice-spacing dependence. We examine combinations of the mass splittings that are sensitive to components of the effective quarkonium potential.

*The XXVII International Symposium on Lattice Field Theory - LAT2009  
July 26-31 2009  
Peking University, Beijing, China*

---

\*Present address: Institut für Theoretische Physik, Universität Regensburg, 93040 Regensburg, Germany.

<sup>†</sup>Speaker.

ensemble	$a$ (approx) (fm)	sea quark ratio $m_{ud}/m_s$
Extra coarse	0.18	0.6, 0.4, 0.2, 0.1
Medium coarse	0.15	0.6, 0.4, 0.2, 0.1
Coarse	0.12	0.6, 0.4, 0.2, 0.15, 0.1
Fine	0.09	0.4, 0.2, 0.1

**Table 1:** Light quark mass ratios and lattice spacings for the ensembles used in this study. The strange quark mass is set to approximately its physical value.

## 1. Introduction

The well-studied charmonium and bottomonium systems have long been used as a test bed for phenomenological models and lattice methods. It is well known that including light sea quarks is essential for obtaining good agreement with experiment. Few studies have carried out a systematic treatment that includes both the chiral (sea quark) and continuum limit. The present study describes progress to date in such an ongoing study [1]. It is based on charm and bottom masses that were determined in previous studies [2]. We limit our attention to lattices with spacing  $a \geq 0.09$  fm. Table 1 lists the 16 ensembles used in this study [3–5].

We simulate the heavy charm and bottom quarks with the Fermilab action [6],

$$\begin{aligned} S = & \sum_n \bar{\psi}_n \psi_n - \kappa \sum_n \left[ \bar{\psi}_n (1 - \gamma_4) U_{n,4} \psi_{n+4} + \bar{\psi}_{n+4} (1 + \gamma_4) U_{n,4}^\dagger \psi_n \right] \\ & - \kappa \zeta \sum_{n,i} \left[ \bar{\psi}_n (r_s - \gamma_i) U_{n,i} \psi_{n+i} + \bar{\psi}_{n+i} (r_s + \gamma_i) U_{n,i}^\dagger \psi_n \right] \\ & - c_B \kappa \zeta \sum_n \bar{\psi}_n i \boldsymbol{\Sigma} \cdot \mathbf{B}_n \psi_n - c_E \kappa \zeta \sum_{n;i} \bar{\psi}_n \boldsymbol{\alpha} \cdot \mathbf{E}_n \psi_n. \end{aligned}$$

The energy of a single quark of spatial momentum  $\mathbf{p}$  in nonrelativistic approximation is

$$E(\mathbf{p}) = m_1 + \frac{\mathbf{p}^2}{2m_2} + O(p^4),$$

where  $m_1$  is the rest mass and  $m_2$  is the “kinetic” mass. They can be made equal if we tune the temporal anisotropy  $\zeta$ . Instead, we set  $\zeta = 1$  and limit our attention to mass splittings for which the additive mass renormalization cancels. We also take  $c_B = c_E = 1/u_0^3$ , where  $u_0$  is the tadpole factor. These choices are explained in greater detail in [7]. The resulting action is just the standard clover action with the clover coefficient set according to the Fermilab interpretation.

## 2. Tuning the heavy quark masses

There are a variety of possible ways to determine the masses ( $\kappa$ 's) of the charm and bottom quarks. Since we know the lattice scale from other measurements, determining the heavy quark mass involves matching a lattice mass with an experimentally observed mass. Tuning to the rest mass  $M_1$  of quarkonium is clearly inaccurate, since it inherits the large additive renormalization

ensemble	$a$	$\kappa_c$	$\kappa_b$
Extra coarse	0.18 fm	0.120	—
Medium coarse	0.15 fm	0.122	0.076
Coarse	0.12 fm	0.122	0.086
Fine	0.09 fm	0.127	0.0923

**Table 2:** Tuned charm and bottom  $\kappa$ 's.

of the quark mass  $m_1$ . Tuning the kinetic mass  $M_2$  of quarkonium is a possibility, but that mass includes a strong binding energy that we would like to study independently of the tuning [8]. So a cleaner approach tunes to the spin-averaged kinetic masses of the  $\bar{D}_s = \frac{1}{4}m_{D_s} + \frac{3}{4}m_{D_s^*}$  and the corresponding  $\bar{B}_s$  multiplet [2, 7]. The heavy-light system has only a mild binding contribution. In this way our study of quarkonium binding is more predictive. Results of tuning are shown in Table 2. Tuning errors are discussed in detail in Ref. [7].

All tuning methods should agree in the continuum limit. Discrepancies at nonzero lattice spacing come from discretization artifacts that grow with  $ma$ , *i.e.*, the quark mass in lattice units. So, for example, at  $a = 0.15$  fm we find that the tuned charm mass is approximately the same whether obtained from the kinetic mass of the  $D_s$  multiplet or the kinetic charmonium mass, but the tuned bottom mass differs significantly:  $\kappa_b = 0.94$  from tuning the kinetic bottomonium mass and 0.76 from tuning the  $B_s$  multiplet.

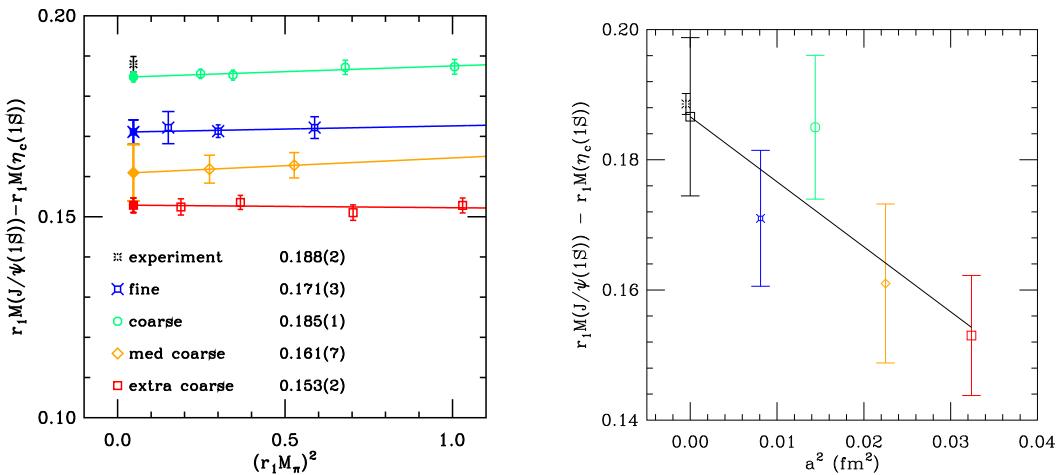
Nonetheless, there are situations that require tuning the quarkonium rest mass. For our companion study of charmonium annihilation effects, mixing between quarkonium states and glueball states could be important [9]. In this case it is important to arrange for a correct placement of the unmixed charmonium and glueball eigenenergies of the lattice hamiltonian, *i.e.*, the unmixed rest masses [10]. However, in that study we hope for at best 15% accuracy in computing the tiny mass shifts coming from annihilation, so we tolerate a mistuning of the kinetic quark mass.

### 3. Results

We measure quarkonium correlators with smeared relativistic and nonrelativistic S-wave and P-wave sources and sinks. To extract masses, we use a multistate fit model with loose Bayesian priors, and we determine statistical errors in mass splittings from a bootstrap analysis. We present a sampling of results. More are given in [7]. We examine them in terms of a traditional nonrelativistic decomposition of the effective heavy quark potential, namely, central, spin-spin, spin-orbit, and tensor contributions.

#### 3.1 Charmonium hyperfine splitting

Hyperfine splitting provides a direct measure of the strength of the spin-spin chromomagnetic interaction. In Fig. 1 we show our results for charmonium hyperfine splitting. Here only the quark line “connected” diagrams are included. The dependence on sea quark mass is evidently quite weak. The continuum extrapolation, shown with kappa tuning errors included, gives 116(5) MeV compared with 117(1) MeV from experiment. It would clearly be good to reduce  $\kappa$ -tuning errors.



**Figure 1:** Results for charmonium hyperfine splitting. Splittings are in  $r_1 = 0.318$  fm units. ( $1/r_1 = 620$  MeV). The left panel shows the chiral extrapolation with only statistical errors shown. The right panel shows the continuum extrapolation in  $a^2$  with kappa tuning errors of 6% included.

superfine	0.06 fm	-3.4(3) MeV
fine	0.09 fm	-5.5(8) MeV

**Table 3:** Contribution from charm annihilation to the charmonium hyperfine splitting

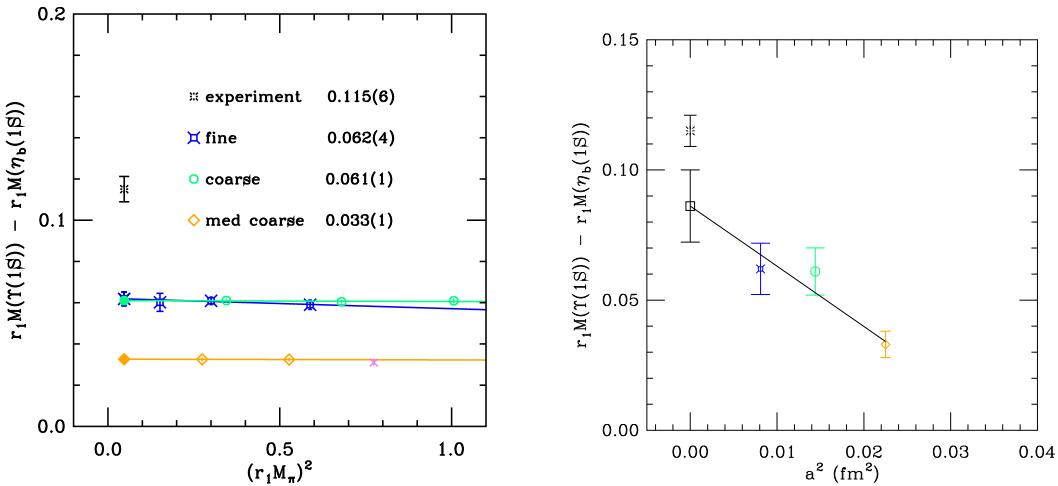
The contribution to the charmonium correlator and mass from quark line disconnected diagrams is expected to be small, so they are usually ignored. Because it is so small, it is a challenge to calculate it [9]. Our most recent results are given in Table 3.1. We find that annihilation processes actually decrease the magnitude of the splitting. The effect is smaller than or comparable to our current errors in the connected contribution.

### 3.2 Bottomonium hyperfine splitting

In Fig. 2 we show results for hyperfine splitting of the bottomonium ground state. The continuum extrapolation gives 53(8) MeV. The  $\eta_b$  was recently found [11, 12] with a splitting of 71(4) MeV from the  $\Upsilon(1S)$ . The HPQCD collaboration reports 61(4)(13) [13] using an NRQCD method with a chromomagnetic interaction of a quality comparable to ours.

### 3.3 $2S - 1S$ level splitting

In Fig. 3 we show results for the splitting of the spin-averaged  $\bar{2S}$  and  $\bar{1S}$  levels. This quantity tests the “central” part of the quarkonium effective potential. We see that agreement with experiment in the charmonium case is not good. It is better in the bottomonium case. Our fit model does not include open charm states. So the  $2S$  charmonium state could be confused with the nearby open charm threshold that comes closer as the light sea quark mass decreases. The dashed line locates



**Figure 2:** The left panel shows the chiral extrapolation with only statistical errors shown. The right panel shows the continuum extrapolation in  $a^2$  with kappa tuning errors of 15% included, resulting in 53(9) MeV.

the physical open charm threshold. For the bottomonium case the open bottom threshold is safely off scale.

### 3.4 $1P - 1S$ splitting

The spin-averaged  $\overline{1P} - \overline{1S}$  splitting, shown in Fig. 4, also tests the central part of the potential. Within errors, our results seem to approach the experimental value.

### 3.5 Spin-orbit and tensor components

The contribution to the  $J = 0, 1$ , and 2 P-wave masses from the spin-orbit term in the quarkonium effective potential can be isolated with the combination

$$m_{1P_{\text{spin-orbit}}} = \frac{1}{9} (5m_{c2} - 2m_{c0} - 3m_{c1})$$

Our result is shown in Fig. 5. This term tests the strength of the chromoelectric interaction. In both cases the results seem to approach the experimental value in the chiral and continuum limits.

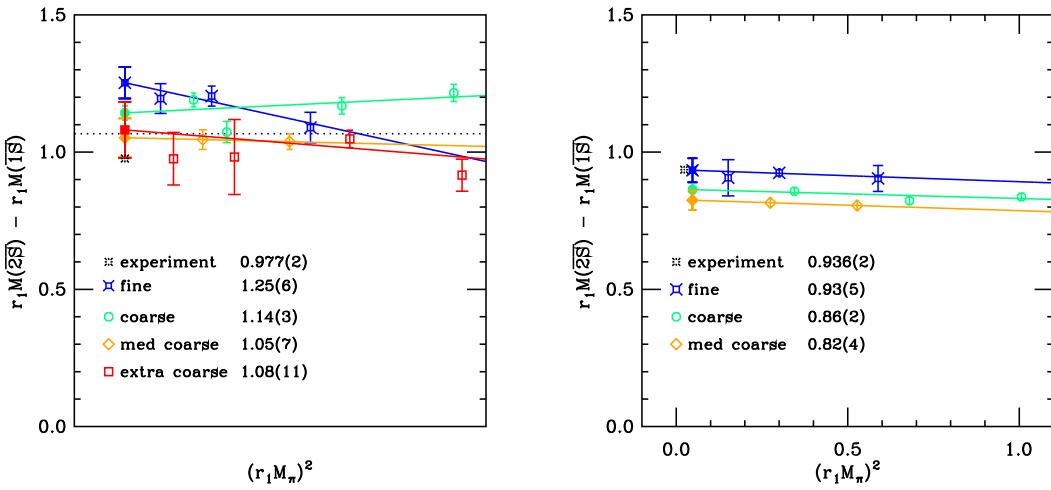
Similarly, the contribution to the P-wave levels from the tensor component is proportional to the combination

$$m_{1P_{\text{tensor}}} = \frac{1}{9} (3m_{c1} - m_{c2} - 2m_{c0}),$$

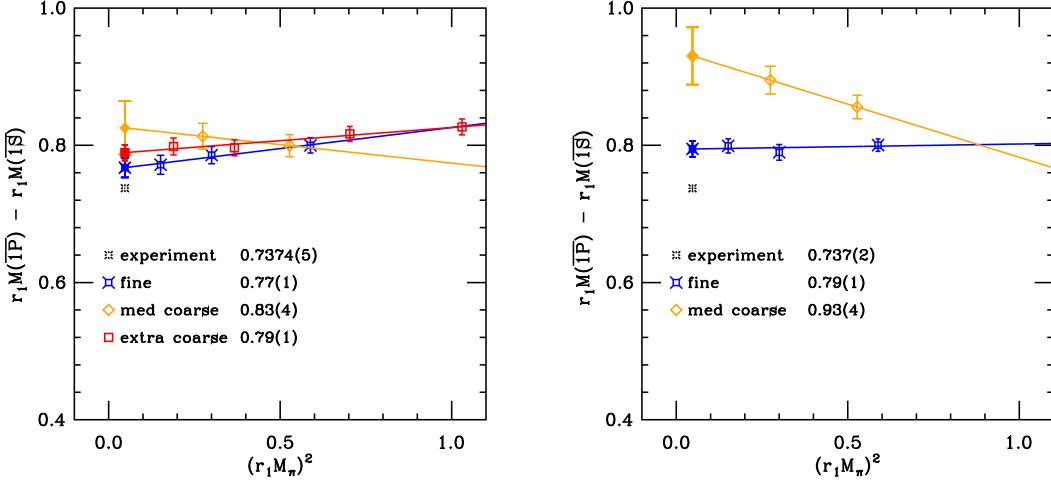
shown in Fig. 6. Since the tensor and spin-spin components both measure the strength of the chromomagnetic interaction, here we divide by the 1S hyperfine splitting to see whether they are proportional. It appears that they are not. Still the results seem to approach the experimental values in the chiral and continuum limits.

### 3.6 Full spectrum

In Fig. 7 we reconstruct the low-lying quarkonium spectrum from splittings, starting from the experimental value for the spin-averaged 1S level.



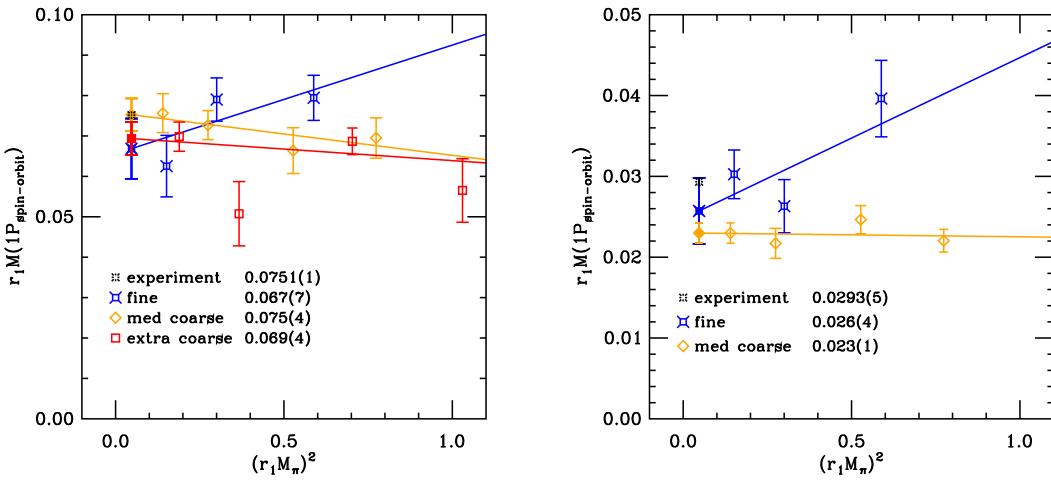
**Figure 3:** Splitting of the spin-averaged  $2S$  and  $1S$  levels in charmonium (left) and bottomonium (right). The dashed line indicates the physical open charm threshold. Since the  $\eta'_b$  has not been observed the “experimental” point uses only the  $\Upsilon(2S)$  in the splitting.



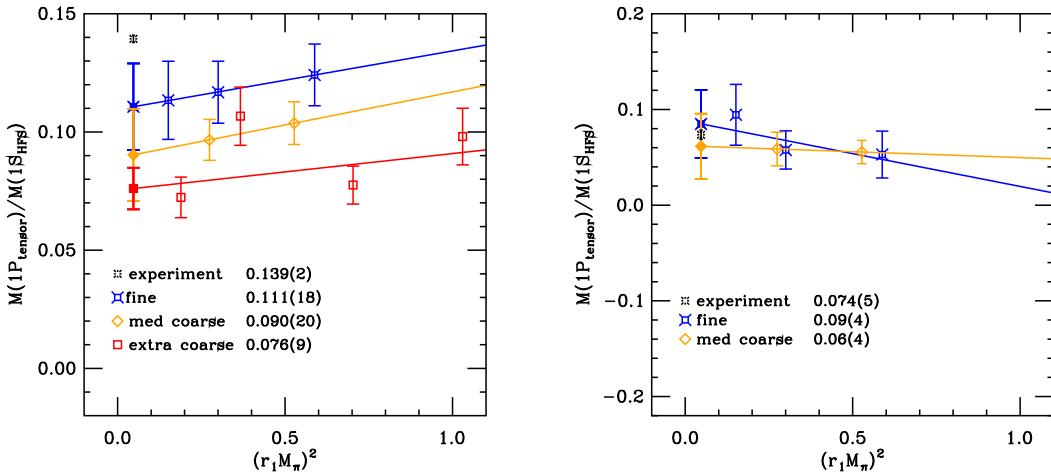
**Figure 4:** Splitting of the spin-averaged  $1P$  and  $1S$  levels in charmonium (left) and bottomonium (right).

#### 4. Conclusion and Outlook

We have seen that in most cases quarkonium level splittings are quite insensitive to the light sea quark masses. Systematic uncertainties in tuning the quark masses are much larger than our statistical errors. With the present set of lattice spacings and the present level of precision, the Fermilab action seems to perform well in the charmonium system, but there are indications that lattice discretization artifacts affect some of our bottomonium splittings. Work currently underway



**Figure 5:** Spin-orbit combination from the  $1P$  levels for charmonium (left) and bottomonium (right).

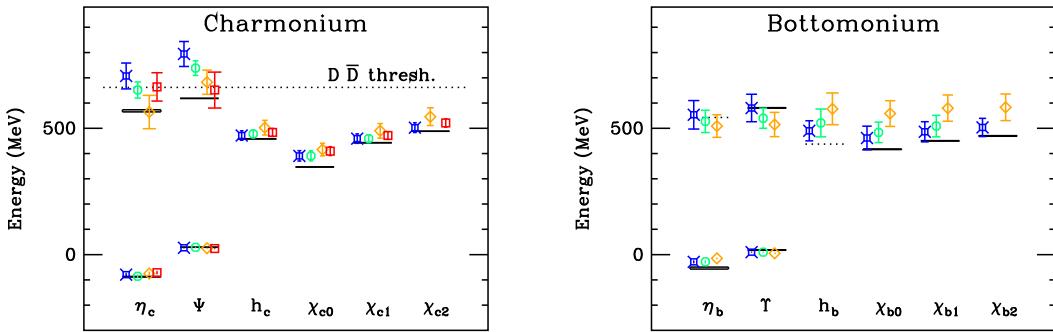


**Figure 6:** The  $1P$  tensor combination, divided by the  $1S$  hyperfine splitting for charmonium (left) and bottomonium (right).

seeks a more precise determination of the charm and bottom masses and will use the still finer MILC-collaboration 0.06 fm lattices.

### Acknowledgments

Work is supported by grants from the US Department of Energy and US National Science Foundation. The lattice ensembles used in this study were generated by the MILC collaboration. Computations for this work were carried out on facilities of the USQCD Collaboration, which are



**Figure 7:** All quarkonium levels in this study, constructed from splittings from the physical  $\bar{D}D$  level for charmonium (left) and bottomonium (right). Error bars include kappa tuning errors. Symbol colors distinguish the lattice spacings: 0.18 fm (red), 0.15 fm (orange), 0.12 fm (green), 0.09 fm (blue)

funded by the Office of Science of the U.S. Department of Energy. T.B. acknowledges current support by the DFG (SFB/TR55).

## References

- [1] M. Di Pierro *et al.*, *Nucl. Phys. Proc. Suppl.* **119**, 586 (2003) [[arXiv:hep-lat/0210051](#)]; *Nucl. Phys. Proc. Suppl.* **129**, 340 (2004) [[arXiv:hep-lat/0310042](#)]; S. Gottlieb *et al.*, PoS(LAT2005)203 (2006) [[arXiv:hep-lat/0510072](#)]; PoS(LAT2006)175 (2006) [[arXiv:0910.0048 \[hep-lat\]](#)].
- [2] C. Bernard *et al.* [Fermilab Lattice and MILC Collaborations], in preparation (2009).
- [3] C. W. Bernard *et al.*, *Phys. Rev. D* **64**, 054506 (2001) [[arXiv:hep-lat/0104002](#)].
- [4] C. Aubin *et al.*, *Phys. Rev. D* **70**, 094505 (2004) [[arXiv:hep-lat/0402030](#)].
- [5] A. Bazavov *et al.* [MILC collaboration], *Rev. Mod. Phys.* (to be published) [[arXiv:0903.3598](#)].
- [6] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, *Phys. Rev. D* **55**, 3933 (1997) [[arXiv:hep-lat/9604004](#)].
- [7] T. Burch *et al.* [Fermilab Lattice and MILC Collaborations], in preparation (2009).
- [8] A. S. Kronfeld, *Nucl. Phys. Proc. Suppl.* **53**, 401 (1997) [[arXiv:hep-lat/9608139](#)].
- [9] L. Levkova and C. E. DeTar, PoS(LATTICE 2008)133 [[arXiv:0809.5086](#)] and work in preparation (2009).
- [10] Y. Chen *et al.*, *Phys. Rev. D* **73**, 014516 (2006) [[arXiv:hep-lat/0510074](#)].
- [11] B. Aubert *et al.* [BaBar Collaboration], *Phys. Rev. Lett.* **101**, 071801 (2008), [[arXiv:0807.1086](#)].
- [12] B. Aubert *et al.* [BaBar Collaboration], [[arXiv:0903.1124](#)].
- [13] A. Gray *et al.* [HPQCD Collaboration], *Phys. Rev. D* **72**, 094507 (2005) [[arXiv:hep-lat/0507013](#)].

# $B \rightarrow \pi l \nu$ semileptonic form factor from three-flavor lattice QCD: A model-independent determination of $|V_{ub}|$

Jon A. Bailey,<sup>1</sup> C. Bernard,<sup>2</sup> C. DeTar,<sup>3</sup> M. Di Pierro,<sup>4</sup> A. X. El-Khadra,<sup>5</sup> R. T. Evans,<sup>5</sup> E. D. Freeland,<sup>6</sup> E. Gamiz,<sup>5</sup> Steven Gottlieb,<sup>7</sup> U. M. Heller,<sup>8</sup> J. E. Hetrick,<sup>9</sup> A. S. Kronfeld,<sup>1</sup> J. Laiho,<sup>2</sup> L. Levkova,<sup>3</sup> P. B. Mackenzie,<sup>1</sup> M. Okamoto,<sup>1</sup> J. N. Simone,<sup>1</sup> R. Sugar,<sup>10</sup> D. Toussaint,<sup>11</sup> and R. S. Van de Water<sup>1,\*</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>Fermi National Accelerator Laboratory, Batavia, Illinois, USA

<sup>2</sup>Department of Physics, Washington University, St. Louis, Missouri, USA

<sup>3</sup>Physics Department, University of Utah, Salt Lake City, Utah, USA

<sup>4</sup>School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, Illinois, USA

<sup>5</sup>Physics Department, University of Illinois, Urbana, Illinois, USA

<sup>6</sup>Liberal Arts Department, The School of the Art Institute of Chicago, Chicago, Illinois, USA

<sup>7</sup>Department of Physics, Indiana University, Bloomington, Indiana, USA

<sup>8</sup>American Physical Society, One Research Road, Ridge, New York, USA

<sup>9</sup>Physics Department, University of the Pacific, Stockton, California, USA

<sup>10</sup>Department of Physics, University of California, Santa Barbara, California, USA

<sup>11</sup>Department of Physics, University of Arizona, Tucson, Arizona, USA

(Received 20 December 2008; published 30 March 2009)

We calculate the form factor  $f_+(q^2)$  for  $B$ -meson semileptonic decay in unquenched lattice QCD with  $2 + 1$  flavors of light sea quarks. We use Asqtad-improved staggered light quarks and a Fermilab bottom quark on gauge configurations generated by the MILC Collaboration. We simulate with several light-quark masses and at two lattice spacings, and extrapolate to the physical quark mass and continuum limit using heavy-light meson staggered chiral perturbation theory. We then fit the lattice result for  $f_+(q^2)$  simultaneously with that measured by the *BABAR* experiment using a parameterization of the form-factor shape in  $q^2$ , which relies only on analyticity and unitarity in order to determine the Cabibbo-Kobayashi-Maskawa matrix element  $|V_{ub}|$ . This approach reduces the total uncertainty in  $|V_{ub}|$  by combining the lattice and experimental information in an optimal, model-independent manner. We find a value of  $|V_{ub}| \times 10^3 = 3.38 \pm 0.36$ .

DOI: 10.1103/PhysRevD.79.054507

PACS numbers: 12.38.Gc, 12.15.Hh, 13.20.He

## I. INTRODUCTION

The semileptonic decay  $B \rightarrow \pi l \nu$  is a sensitive probe of the heavy-to-light-quark-flavor-changing  $b \rightarrow u$  transition. When combined with an experimental measurement of the differential decay rate, a precise QCD determination of the  $B \rightarrow \pi l \nu$  form-factor allows a clean determination of the Cabibbo-Kobayashi-Maskawa (CKM) matrix element  $|V_{ub}|$  with all sources of systematic uncertainty under control. In the standard model, the differential decay rate for this process is

$$\frac{d\Gamma(B \rightarrow \pi l \nu)}{dq^2} = \frac{G_F^2 |V_{ub}|^2}{192 \pi^3 m_B^3} [(m_B^2 + m_\pi^2 - q^2)^2 - 4m_B^2 m_\pi^2]^{3/2} \times |f_+(q^2)|^2, \quad (1)$$

where  $q \equiv p_B - p_\pi$  is the momentum transferred from the  $B$  meson to the outgoing leptons. The form factor  $f_+(q^2)$  parameterizes the hadronic contribution to the weak decay,

and must be calculated nonperturbatively from first principles using lattice QCD.

A precise knowledge of CKM matrix elements such as  $|V_{ub}|$  is important not only because they are fundamental parameters of the standard model, but because inconsistencies between independent determinations of the CKM matrix elements and  $CP$ -violating phase would provide evidence for new physics. Although the standard model has been amazingly successful in describing the outcome of most particle physics experiments to date, it cannot account for gravity, dark matter, and dark energy, or the large matter-antimatter asymmetry of the Universe. Thus, we know that it is incomplete, and expect new physics to affect the quark-flavor sector to some degree, although we do not know *a priori* what experimental and theoretical precision will be needed to observe it.

The determination of  $|V_{ub}|$  from  $B \rightarrow \pi l \nu$  semileptonic decay relies upon the assumption that, because the leading standard model decay amplitude is mediated by tree-level  $W$ -boson exchange, it will not be significantly affected by new physics at the current level of achievable precision. Recently, however, hints of new physics have appeared in

\*ruthv@bnl.gov

various regions of the heavy-quark-flavor sector such as  $CP$  asymmetries in  $B \rightarrow K\pi$  [1], constraints on  $\sin(2\beta)$  from  $\Delta F = 2$  neutral meson mixing and 1-loop penguin-induced decays [2], and the phase of the  $B_s$ -mixing amplitude [3–5]. The unexpected inconsistency most relevant to our new lattice QCD calculation of the  $B \rightarrow \pi\ell\nu$  form factor and  $|V_{ub}|$  is the current “ $f_{D_s}$  puzzle” [6]. The HPQCD Collaboration’s lattice QCD calculation of the  $D_s$ -meson leptonic decay constant  $f_{D_s}$  [7] disagrees with the latest results from the Belle, *BABAR*, and CLEO experiments [8–12] at the  $3\sigma$  level, although HPQCD’s determinations of the masses  $m_{D^+}$  and  $m_{D_s}$  and the decay constants  $f_\pi$ ,  $f_K$ , and  $f_{D^+}$  all agree quite well with experimental measurements [13,14]. Furthermore, because the significance of the discrepancy is dominated by the statistical experimental uncertainties, it cannot easily be explained by an underestimate of the theoretical uncertainties. Additional lattice QCD calculations of  $f_{D_s}$  are needed to either confirm or reduce the inconsistency. If the disagreement holds up, however, it is evidence for a large new physics contribution to a tree-level standard model process at the few percent level. Therefore, although  $B \rightarrow \pi\ell\nu$  semileptonic decay provides a theoretically clean method for determining  $|V_{ub}|$  within the framework of the standard model, we should keep in mind that new physics could appear in  $b \rightarrow u$  transitions.

Understanding and controlling all sources of systematic uncertainty in lattice QCD calculations of hadronic weak matrix elements is essential in order to allow accurate determinations of standard model parameters and reliable searches for new physics. The hadronic amplitudes for  $B \rightarrow \pi\ell\nu$ , in particular, can be calculated accurately using current lattice QCD methods because the decay process is “gold plated,” i.e., there is only a single stable hadron in both the initial and final states. Lattice calculations with staggered quarks allow for realistic QCD simulations with dynamical quarks as light as  $m_s/10$ , multiple lattice spacings, large physical volumes, and high statistics. The resulting simulations of many light-light and heavy-light meson quantities with dynamical staggered quarks are in excellent numerical agreement with experimental results [15]. This includes both postdictions, such as the pion decay constant [16], and predictions, as in the case of the  $B_c$  meson mass [17]. Such successes show that the systematic uncertainties in these lattice QCD calculations are under control, and give confidence that additional calculations using the same methods are reliable.

The publicly available MILC gauge configurations with three flavors of improved staggered quarks [18] that have enabled these precise lattice calculations make use of the “fourth-root” procedure for removing the undesired four-fold degeneracy of staggered lattice fermions. Although this procedure has not been rigorously proven correct, Shamir uses plausible assumptions to argue that the continuum limit of the rooted theory is in the same universality

class as QCD [19,20]. The rooting procedure leads to violations of unitarity that vanish in the continuum limit; both theoretical arguments [21,22] and numerical simulations [23–25], however, show that the unitarity-violating lattice artifacts in the pseudo-Goldstone boson sector can be described and hence removed using rooted staggered chiral perturbation theory (rS $\chi$ PT), the low-energy effective description of the rooted staggered lattice theory [26–28]. Given the wealth of numerical and analytical evidence supporting the validity of the rooting procedure, most of which is reviewed in Refs. [29–31], we work under the plausible assumption that the continuum limit of the rooted staggered theory is QCD. We note, however, that it is important to have cross-checks of lattice calculations of phenomenologically important quantities using a variety of fermion formulations, since they all have different sources of systematic uncertainty.

Both existing unquenched lattice calculations of the  $B \rightarrow \pi\ell\nu$  form factor use the MILC configurations. When combined with the Heavy Flavor Averaging Group’s latest determination of the experimental decay rate from ICHEP 2008 [32], they yield the following values for  $|V_{ub}|$ :

$$|V_{ub}| \times 10^3 = 3.40 \pm 0.20^{+0.59}_{-0.39} \quad \text{HPQCD [33]}, \quad (2)$$

$$|V_{ub}| \times 10^3 = 3.62 \pm 0.22^{+0.63}_{-0.41} \quad \text{Fermilab-MILC [34]}, \quad (3)$$

where the errors are experimental and theoretical, respectively. Both analyses primarily rely upon data generated at a “coarse” lattice spacing of  $a \approx 0.12$  fm, and use a smaller amount of “fine” data at  $a \approx 0.09$  fm to check the estimate of discretization errors. Neither is therefore able to extrapolate the  $B \rightarrow \pi\ell\nu$  form factor to the continuum ( $a \rightarrow 0$ ). The most significant difference in the two calculations is their use of different lattice formulations for the bottom quarks. The HPQCD Collaboration [33] uses nonrelativistic heavy quarks [34], whereas we use relativistic clover quarks with the Fermilab interpretation [35] via heavy-quark effective theory (HQET) [36–38]. Both methods work quite well for heavy bottom quarks. The Fermilab treatment, however, has the advantage that it can also be applied to charm quarks; we can therefore use the same method for other semileptonic form factors such as  $D \rightarrow \pi\ell\nu$ ,  $D \rightarrow K\ell\nu$ , and  $B \rightarrow D^*\ell\nu$  [39,40]. The two unquenched lattice calculations of the  $B \rightarrow \pi\ell\nu$  form factor, which have largely independent sources of systematic uncertainty, nevertheless lead to consistent values of  $|V_{ub}|$  with similar total errors of  $\sim 15\%$ .

In this paper we present a new *model-independent* unquenched lattice QCD calculation of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor and  $|V_{ub}|$ . Our work builds upon the previous Fermilab-MILC calculation and improves upon it in several ways. We now include data on both the coarse and fine MILC lattices, and can therefore take the  $a \rightarrow 0$

limit of our data, which is generated at nonzero lattice spacing. We also have additional statistics on a subset of the coarse ensembles. The most important improvements, however, are in the analysis procedures.

We have removed all model-dependent assumptions about the shape in  $q^2$  of the form factor from the current analysis. Our result is therefore theoretically cleaner and more reliable than those of previous lattice QCD calculations. The first refinement over previous unquenched lattice  $B \rightarrow \pi\ell\nu$  form-factor calculations is in the treatment of the chiral and continuum extrapolations. We simultaneously extrapolate to physical quark masses and zero lattice spacing and interpolate in the momentum transfer  $q^2$  by performing a single fit to our entire data set (all values of  $m_q$ ,  $a$ , and  $q^2$ ) guided by functional forms derived in heavy-light meson staggered chiral perturbation theory (HMS $\chi$ PT) [41]. We thereby extract the physical form factor  $f_+(q^2)$  in a controlled manner without introducing a particular ansatz for the form factor's  $q^2$  dependence. The second refinement over previous unquenched  $B \rightarrow \pi\ell\nu$  lattice form-factor calculations is in the combination of the lattice form-factor result and experimental data for the decay rate to determine the CKM matrix element  $|V_{ub}|$ . We fit our lattice numerical Monte Carlo data and the 12-bin *BABAR* experimental data [42] together to the model-independent “z expansion” of the form factor given in Ref. [43], in which the form factor is described by a power series in a small quantity  $z$  with the sum of the squares of the series coefficients bounded by unitarity constraints. We leave the relative normalization factor  $|V_{ub}|$  as a free parameter to be determined by the fit, thereby extracting  $|V_{ub}|$  in an optimal, model-independent way. Others have also fit lattice and experimental results together using different, equally valid, parameterizations [44,45]. This work, however, is the first to use the full correlation matrices, derived directly from the data, for both the lattice calculation and experimental measurement.

This paper is organized as follows: In Sec. II, we describe the details of our numerical simulations. We discuss the gluon, light-quark, and heavy-quark lattice actions, and present the parameters used, such as the quark masses and lattice spacings. We then define the matrix elements needed to calculate the semileptonic form factors and discuss the method for matching the lattice heavy-light current to the continuum. Next, we describe our analysis for determining the form factors in Sec. III. This is a three-step procedure. We first fit pion and  $B$ -meson 2-point correlation functions to extract the meson masses. We then fit the  $B \rightarrow \pi$  3-point function, using the masses and amplitudes from the 2-point fits as input, to extract the lattice form factors at each value of the light-quark mass and lattice spacing. Finally, we extrapolate the results at unphysical quark masses and nonzero lattice spacing to the physical light-quark masses and zero lattice spacing using HMS $\chi$ PT. In Sec. IV, we estimate the contributions

of the various systematic uncertainties to the form factors, discussing each item in the error budget separately. We then present the final result for  $f_+(q^2)$  with a detailed breakdown of the error by source in each  $q^2$  bin. We combine our result for the form factor with experimental data from the *BABAR* Collaboration to determine the CKM matrix element  $|V_{ub}|$  in Sec. V. We also define the model-independent description of the form-factor shape that we use in the fit and discuss alternative parameterizations of the form factor. Finally, in Sec. VI, we compare our results with those of previous unquenched lattice calculations. We also compare our determination of  $|V_{ub}|$  with inclusive determinations and to the preferred values from the global CKM unitarity triangle analysis. We conclude by discussing the prospects for improvements in our calculation and its impact on searches for new physics in the quark-flavor sector.

## II. LATTICE CALCULATION

In this section we describe the details of our numerical lattice simulations. We first present the actions and parameters used for the light (up, down, strange) and heavy (bottom) quarks in Sec. II A. We then define the procedure for constructing lattice correlation functions with both staggered light and Wilson heavy quarks in Sec. II B. Finally, in Sec. II C, we show how to match the lattice heavy-light vector currents to the continuum with a mostly nonperturbative method, so that lattice perturbation theory is only needed to estimate a small correction.

### A. Actions and parameters

We use the ensembles of lattice gauge fields generated by the MILC Collaboration and described in Ref. [18] at two lattice spacings,  $a \approx 0.12$  and  $0.09$  fm, in our numerical lattice simulations of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor. The ensembles include the effects of three dynamical staggered quarks—two degenerate light quarks with masses ranging from  $m_s/8 - m_s/2$  and one heavier quark tuned to within 10–30% of the physical strange quark mass. This allows us to perform a controlled extrapolation to both the continuum and the physical average  $u$ - $d$  quark mass. The physical lattice volumes are all sufficiently large ( $m_\pi L \geq 4$ ) to ensure that effects due to the finite spatial extent remain small.

For each independent ensemble we compute the light valence quark in the 2-point and 3-point correlation functions only at the same mass,  $m_l$ , as the light quark in the sea sector. Thus all of our simulations are at the “full QCD” point. Note, however, that we still have many correlated data points on each ensemble because of the multiple pion energies. Table I shows the combinations of lattice spacings, lattice volumes, and quark masses used in our calculation.

For bottom quarks in 2-point and 3-point correlation functions we use the Sheikholeslami-Wohlert (SW) “clo-

TABLE I. Lattice simulation parameters. The columns from left to right are the approximate lattice spacing in fm, the bare light-quark masses  $am_l/am_s$ , the linear spatial dimension of the lattice in fm, the dimensionless factor  $m_\pi L$  (corresponding to the taste-pseudoscalar pion composed of light sea quarks), the dimensions of the lattice in lattice units, the number of configurations used for this analysis, the clover term  $c_{\text{SW}}$  and bare  $\kappa$  value used to generate the bottom quark, and the improvement coefficient used to rotate the bottom quark field in the  $b \rightarrow u$  vector current.

$a$ (fm)	$am_l/am_s$	$L$ (fm)	$m_\pi L$	Volume	# Configs.	$c_{\text{SW}}$	$\kappa_b$	$d_1$
0.09	0.0062/0.031	2.4	4.1	$28^3 \times 96$	557	1.476	0.0923	0.09474
0.09	0.0124/0.031	2.4	5.8	$28^3 \times 96$	518	1.476	0.0923	0.09469
0.12	0.005/0.05	2.9	3.8	$24^3 \times 64$	529	1.72	0.086	0.09372
0.12	0.007/0.05	2.4	3.8	$20^3 \times 64$	836	1.72	0.086	0.09372
0.12	0.01/0.05	2.4	4.5	$20^3 \times 64$	592	1.72	0.086	0.09384
0.12	0.02/0.05	2.4	6.2	$20^3 \times 64$	460	1.72	0.086	0.09368

ver” action [46] with the Fermilab interpretation via HQET [35,36], which is well suited for heavy quarks, even when  $am_Q \gtrsim 1$ . Because the spin-flavor symmetry of heavy-quark systems is respected by the lattice regulator, the expansion in  $1/m_Q$  of the heavy-quark lattice action has the same form as the  $1/m_Q$  expansion of the heavy-quark part of the QCD action. Discretization effects in the lattice heavy-quark action are therefore parameterized order by order in the heavy-quark expansion by deviations of effective operator coefficients from their values in continuum QCD. Thus, in principle, the lattice heavy-quark action can be improved to arbitrarily high orders in HQET by tuning a sufficiently large number of parameters in the lattice action. In practice, we tune the hopping parameter  $\kappa$ , and the clover coefficient  $c_{\text{SW}}$ , of the SW action, to remove discretization effects through next-to-leading order (NLO),  $\mathcal{O}(1/m_Q)$ , in the heavy-quark expansion.

The SW action includes a dimension-five interaction with a coupling  $c_{\text{SW}}$  that must be adjusted to normalize the heavy quark’s chromomagnetic moment correctly [35]. In our calculation we set the value of  $c_{\text{SW}} = u_0^{-3}$ , as suggested by tadpole-improved, tree-level perturbation theory [47]. We determine the value of  $u_0$  either from the plaquette ( $a \approx 0.09$  fm) or from the Landau link ( $a \approx 0.12$  fm). The tadpole-improved bare quark mass for SW quarks is given by

$$am_0 = \frac{1}{u_0} \left( \frac{1}{2\kappa} - \frac{1}{2\kappa_{\text{crit}}} \right), \quad (4)$$

such that tuning the parameter  $\kappa$  to the critical quark hopping parameter  $\kappa_{\text{crit}}$  leads to a massless pion. Before generating the correlation functions needed for the  $B \rightarrow \pi l \nu$  form factor, we compute the spin-averaged  $B_s$  kinetic mass on a subset of the available ensembles in order to tune the bare  $\kappa$  value for bottom (and hence the corresponding bare quark mass) to its physical value [35]. We then use the tuned value of  $\kappa_b$  for the  $B \rightarrow \pi l \nu$  form-factor production runs. Table I shows the values of the clover coefficient and tuned  $\kappa_b$  used in our calculation.

In order to take advantage of the improved action in the calculation of the  $B \rightarrow \pi l \nu$  form factor, we must also improve the flavor-changing vector current to the same order in the heavy-quark expansion. We remove errors of  $\mathcal{O}(1/m_Q)$  in the vector current by rotating the heavy-quark field used in the matrix element calculation as

$$\psi_b \rightarrow \Psi_b = (1 + ad_1 \vec{\gamma} \cdot \vec{D}_{\text{lat}}) \psi_b, \quad (5)$$

where  $\vec{D}_{\text{lat}}$  is the symmetric, nearest-neighbor, covariant difference operator. We set  $d_1$  to its tadpole-improved tree-level value of [35]

$$d_1 = \frac{1}{u_0} \left( \frac{1}{2 + m_0 a} - \frac{1}{2(1 + m_0 a)} \right). \quad (6)$$

The values of the rotation parameter used in our calculation are given in Table I.

In order to convert dimensionful quantities determined in our lattice simulations into physical units, we need to know the value of the lattice spacing,  $a$ , which we find by computing a physical quantity that can be compared directly with experiment. We first determine the relative lattice scale by calculating the ratio  $r_1/a$  on each ensemble, where  $r_1$  is related to the force between static quarks,  $r_1^2 F(r_1) = 1.0$  [48]. These  $r_1/a$  estimates are then smoothed by fitting to a smooth function of the gauge coupling and quark masses. This scale-setting method has the advantage that the ratio  $r_1/a$  can be determined precisely from a fit to the static quark potential [49,50]. We convert all of our data from lattice spacing units into  $r_1$  units before performing any chiral fits in order to account for slight differences in the value of the lattice spacing between ensembles. In this work we use the value of  $r_1^{\text{phys}} = 0.3108(15)(^{+26}_{-79})$  obtained by combining a recent lattice determination of  $r_1 f_\pi$  [51] with the PDG value of  $f_\pi = 130.7 \pm 0.1 \pm 0.36$  MeV [52] to convert lattice results from  $r_1$  units to physical units.

## B. Heavy-light meson correlation functions

The  $B \rightarrow \pi l \nu$  semileptonic form factors parameterize the hadronic matrix element of the  $b \rightarrow u$  quark flavor-

changing vector current  $\mathcal{V}^\mu \equiv i\bar{u}\gamma^\mu b$ :

$$\begin{aligned} \langle \pi | \mathcal{V}^\mu | B \rangle &= f_+(q^2) \left( p_B^\mu + p_\pi^\mu - \frac{m_B^2 - m_\pi^2}{q^2} q^\mu \right) \\ &\quad + f_0(q^2) \frac{m_B^2 - m_\pi^2}{q^2} q^\mu, \end{aligned} \quad (7)$$

where  $q^2$  is the momentum transferred to the outgoing lepton pair. For calculations on the lattice and in HQET, it is more convenient to write the matrix element as [53]

$$\langle \pi | \mathcal{V}^\mu | B \rangle = \sqrt{2m_B} [v^\mu f_{\parallel}(E_\pi) + p_\perp^\mu f_{\perp}(E_\pi)], \quad (8)$$

where  $v^\mu = p_B^\mu/m_B$  is the four-velocity of the  $B$  meson,  $p_\perp^\mu = p_\pi^\mu - (p_\pi \cdot v)v^\mu$  is the component of the pion momentum orthogonal to  $v$ , and  $E_\pi = p_\pi \cdot v = (m_B^2 + m_\pi^2 - q^2)/(2m_B)$  is the energy of the pion in the  $B$ -meson rest frame ( $\vec{p}_B = \vec{0}$ ). In this frame the form factors  $f_{\parallel}(E_\pi)$  and  $f_{\perp}(E_\pi)$  are directly proportional to the hadronic matrix elements

$$f_{\parallel}(E_\pi) = \frac{\langle \pi | \mathcal{V}^0 | B \rangle}{\sqrt{2m_B}}, \quad (9)$$

$$f_{\perp}(E_\pi) = \frac{\langle \pi | \mathcal{V}^i | B \rangle}{\sqrt{2m_B}} \frac{1}{p_\pi^i}. \quad (10)$$

We therefore first calculate the hadronic matrix elements in Eqs. (9) and (10) in the rest frame of the  $B$  meson to obtain  $f_{\parallel}(E_\pi)$  and  $f_{\perp}(E_\pi)$ , and then extract the standard form factors  $f_0(q^2)$  and  $f_+(q^2)$  using the following relations:

$$\begin{aligned} f_0(q^2) &= \frac{\sqrt{2m_B}}{m_B^2 - m_\pi^2} [(m_B - E_\pi)f_{\parallel}(E_\pi) \\ &\quad + (E_\pi^2 - m_\pi^2)f_{\perp}(E_\pi)], \end{aligned} \quad (11)$$

$$f_+(q^2) = \frac{1}{\sqrt{2m_B}} [f_{\parallel}(E_\pi) + (m_B - E_\pi)f_{\perp}(E_\pi)]. \quad (12)$$

These relations automatically satisfy the kinematic constraint  $f_+(0) = f_0(0)$ .

The 2-point and 3-point correlation functions needed to extract the lattice matrix element for  $B \rightarrow \pi\ell\nu$  decay are

$$C_2^\pi(t; \vec{p}_\pi) = \sum_{\vec{x}} e^{i\vec{p}_\pi \cdot \vec{x}} \langle \mathcal{O}_\pi(0, \vec{0}) \mathcal{O}_\pi^\dagger(t, \vec{x}) \rangle, \quad (13)$$

$$C_2^B(t) = \sum_{\vec{x}} \langle \mathcal{O}_B(0, \vec{0}) \mathcal{O}_B^\dagger(t, \vec{x}) \rangle, \quad (14)$$

$$C_{3,\mu}^{B \rightarrow \pi}(t, T; \vec{p}_\pi) = \sum_{\vec{x}, \vec{y}} e^{i\vec{p}_\pi \cdot \vec{y}} \langle \mathcal{O}_\pi(0, \vec{0}) V_\mu(t, \vec{y}) \mathcal{O}_B^\dagger(T, \vec{x}) \rangle, \quad (15)$$

where  $\mathcal{O}_B$  and  $\mathcal{O}_\pi$  are interpolating operators for the  $B$  meson and pion, respectively, and  $V_\mu$  is the heavy-light vector current on the lattice.

In practice, to construct the heavy-light bilinears we must combine a staggered light quark, which is a 1-component spinor, with a 4-component Wilson-type bottom quark; we do so using the method established by Wingate *et al.* in Ref. [54]. For the  $B$  meson we use a mixed-action interpolating operator:

$$\mathcal{O}_{B,\Xi}(x) = \bar{\psi}_\alpha(x) \gamma_5^5 \Omega_{\beta\Xi}(x) \chi(x), \quad (16)$$

where  $\alpha, \beta$  are spin indices and  $\Omega(x) \equiv \gamma_0^{x_0} \gamma_1^{x_1} \gamma_2^{x_2} \gamma_3^{x_3}$ . The fields  $\psi$  and  $\chi$  are the 4-component clover quark field and 1-component staggered field, respectively. Based on the transformation properties of  $\mathcal{O}_{B,\Xi}(x)$  under shifts by one lattice spacing,  $\Xi$  plays the role of a (fermionic) taste index [31,54]. Once  $\mathcal{O}_{B,\Xi}(x)$  is summed over  $2^4$  hypercubes in the correlation functions that we compute,  $\Xi$  also takes on the role of a taste degree of freedom, in the sense of Refs. [55,56]. Because the heavy-quark field  $\bar{\psi}_\alpha(x)$  is slowly varying over a hypercube, it does not affect the construction of Refs. [55,56].

For the pion we use the local pseudoscalar interpolating operator

$$\mathcal{O}_\pi(x) = \varepsilon(x) \bar{\chi}(x) \chi(x), \quad (17)$$

where  $\varepsilon(x) \equiv (-1)^{(x_1+x_2+x_3+x_4)}$ . We take the vector current to be

$$V_\Xi^\mu(x) = \bar{\Psi}_\alpha(x) \gamma_\alpha^\mu \Omega_{\beta\Xi}(x) \chi(x), \quad (18)$$

where  $\Psi$  is the rotated heavy-quark field given in Eq. (5). When forming  $C_2^B(t)$  and  $C_{3,\mu}^{B \rightarrow \pi}(t, T; \vec{p}_\pi)$ , we sum over the taste index. This yields the same correlation functions, with respect to taste, as in Ref. [54]. Our principal difference with Ref. [54] is to use 4-component heavy quarks instead of 2-component nonrelativistic (NRQCD) quarks, and to derive the correlators in the staggered formalism, without the introduction of naive fermions.

We work in the rest frame of the  $B$  meson, so only the pions carry momentum. We compute both the 2-point function  $C_2^\pi(t; \vec{p}_\pi)$  and the 3-point function  $C_{3,\mu}^{B \rightarrow \pi}(t, T; \vec{p}_\pi)$  at discrete values of the momenta  $\vec{p}_\pi = 2\pi(0, 0, 0)/L, 2\pi(1, 0, 0)/L, 2\pi(1, 1, 0)/L, 2\pi(1, 1, 1)/L$ , and  $2\pi(2, 0, 0)/L$  allowed by the finite spatial lattice volume. We use only data through momentum  $\vec{p}_\pi = 2\pi(1, 1, 1)/L$ , however, because the statistical errors in the correlators increase significantly with momentum.

We use a local source for the pions throughout the calculation, while we smear the  $B$ -meson wave function in both the 2-point function  $C_2^B(t)$  and the 3-point function  $C_{3,\mu}^{B \rightarrow \pi}(t, T; \vec{p}_\pi)$ :

$$\tilde{\mathcal{O}}_{B,\Xi}(t, \vec{x}) = \sum_{\vec{y}} S(\vec{y}) \bar{\psi}_\alpha(t, \vec{x} + \vec{y}) \gamma_5^5 \Omega_{\beta\Xi}(t, \vec{x}) \chi(t, \vec{x}), \quad (19)$$

where  $S(\vec{y})$  is the spatial smearing function. This reduces contamination from heavier excited states and allows a

better determination of the desired ground-state amplitude. In our study of choices for how to smear the  $B$  meson, we found that a wall source,  $S(\vec{y}) = 1$ , worked extremely well for suppressing excited-state contamination, but at the cost of large statistical errors in the 2-point and 3-point correlation functions. In contrast, use of a 1S wave function  $S(\vec{y}) = \exp(-\mu|\vec{y}|)$ , optimized to have good overlap with the charmonium ground state led to smaller statistical errors at the cost of undesirably large excited-state contributions to the 3-point function that would make it difficult to extract the ground-state amplitude. In order to achieve a balance between small statistical errors and minimal excited-state contamination, we tune the coefficient of the exponential in the 1S wavefunction to the smallest value (i.e., the widest smearing) for which the  $B$ -meson 2-point effective mass is still well behaved; we find a value of  $a\mu = 0.20$  for the coarse ensembles. We note that our determination of the optimal  $B$ -meson smearing function is consistent with the theoretical expectation that the  $B$ -meson wave function should be wider than the charmonium wave function.

For the calculation of the 3-point function, we fix the location of the pion source at  $t_i = 0$  and the location of the  $B$ -meson sink at  $t_f = T$ , and vary the position of the operator over all times  $t$  in between. If the source-sink separation is too small, then the entire time range  $0 < t < T$  is contaminated by excited states, but if the source-sink separation is too large, then the correlation function becomes extremely noisy. In practice, we set the sink time to  $T = 16$  on the coarse lattices; we have checked, however, that our results using this choice are consistent with those determined from using  $T = 12$  and  $T = 20$ . On the fine lattices we scale the source-sink separation by the approximate ratio of the lattice spacings  $a_{\text{fine}}/a_{\text{coarse}}$  and use  $T = 24$ .

In order to minimize the statistical errors given the available number of configurations in each ensemble, we compute the necessary 2-point and 3-point correlations not only with a source time of  $t_i = 0$ , but also with source times of  $t_i = n_t/4$ ,  $n_t/2$ , and  $3n_t/4$  ( $n_t$  is the temporal extent of the lattice) and the sink time  $T$  shifted accordingly. We then average the results from the four source times; this effectively increases our statistics by a factor of 4.

### C. Heavy-light current renormalization

In order to recover the desired continuum matrix element, the lattice amplitude must be multiplied by the appropriate renormalization factor  $Z_{V_\mu}^{bl}$ :

$$\langle \pi | \mathcal{V}_\mu | B \rangle = Z_{V_\mu}^{bl} \times \langle \pi | V_\mu | B \rangle, \quad (20)$$

where  $V_\mu$  and  $\mathcal{V}_\mu$  are the lattice and continuum  $b \rightarrow u$  vector currents, respectively. This removes the dominant discretization errors from the lattice current operator. In terms of the form factors, Eq. (20) can be rewritten as

$$f_{\parallel} = Z_{V_0}^{bl} \times f_{\parallel}^{\text{lat}}, \quad (21)$$

$$f_{\perp} = Z_{V_i}^{bl} \times f_{\perp}^{\text{lat}}, \quad (22)$$

where explicit expressions relating  $f_{\parallel}^{\text{lat}}$  and  $f_{\perp}^{\text{lat}}$  to correlation functions are given in Eqs. (40) and (41).

In this work, we calculate  $Z_{V_\mu}^{bl}$  via the mostly nonperturbative method used in the earlier quenched Fermilab calculation [53]. We first rewrite  $Z_{V_\mu}^{bl}$  as

$$Z_{V_\mu}^{bl} = \rho_{V_\mu}^{hl} \sqrt{Z_V^{bb} Z_V^{ll}}. \quad (23)$$

The flavor-conserving renormalization factors  $Z_V^{bb}$  and  $Z_V^{ll}$  account for most of the value of  $Z_V^{bl}$  [37]. They can be determined from standard heavy-light meson charge normalization conditions

$$Z_V^{ll} \times \langle D | V^{ll,0} | D \rangle = 1, \quad (24)$$

$$Z_V^{bb} \times \langle B | V^{bb,0} | B \rangle = 1, \quad (25)$$

where the light-light and heavy-heavy lattice vector currents are given by

$$V_{\Xi\Xi'}^{ll,\mu}(x) = \chi^\dagger(x) \Omega(x)_{\Xi\alpha}^\dagger \gamma_{\alpha\beta}^\mu \Omega(x)_{\beta\Xi'} \chi(x), \quad (26)$$

$$V^{bb,\mu}(x) = \bar{\Psi}_{b\alpha}(x) \gamma_{\alpha\beta}^\mu \Psi_{b\beta}(x), \quad (27)$$

respectively. In order to reduce the statistical errors in  $Z_V^{ll}$ , we compute the lattice matrix element  $\langle D | V^{ll,0} | D \rangle$  using a clover charm quark as the spectator in the 3-point correlation function. We eliminate contamination from staggered oscillating states in the determination of  $Z_V^{bb}$  by using a clover strange quark for the spectator in the 3-point correlation function  $\langle B | V^{bb,0} | B \rangle$ . Once  $Z_V^{ll}$  and  $Z_V^{bb}$  have been determined nonperturbatively, the remaining correction factor in Eq. (23),  $\rho_{V_\mu}^{hl}$ , is expected to be close to unity because most of the radiative corrections, including contributions from tadpole graphs, cancel in the ratio [37]. We therefore estimate  $\rho_{V_\mu}^{hl}$  from 1-loop lattice perturbation theory [47].

The matching factor  $\rho_{V_\mu}^{hl}$  has been calculated by a subset of the present authors, and a separate publication describing the details is in preparation [57]. The corrections to  $\rho_{V_\mu}^{hl}$  can be expressed as a perturbative series expansion in powers of the strong coupling

$$\rho_{V_\mu}^{hl} = 1 + 4\pi\alpha_V(q^*)\rho_{V_\mu}^{hl[1]} + \mathcal{O}(\alpha_V^2), \quad (28)$$

where  $\alpha_V(q^*)$  is the renormalized coupling constant in the  $V$ -scheme and is determined from the static quark potential with the same procedure as is used in Ref. [58]. The scale  $q^*$ , which should be the size of a typical gluon loop momentum, is computed via an extension of the methods outlined by Brodsky, Lepage, and Mackenzie [47,59] and Hornbostel, Lepage, and Morningstar [60]. The value of  $q^*$

ranges from 2.0–4.5 GeV for the parameters used in our simulations. The 1-loop coefficient  $\rho_{V_\mu}^{h[1]}$  and higher moments are calculated using automated perturbation theory and numerical integration as described in Refs. [61,62]. We find that the perturbative corrections to matrix elements of the temporal vector current,  $V_0$ , are less than a percent, while the corrections to matrix elements of the spatial vector current  $V_i$  are 3–4%.

### III. ANALYSIS

In this section, we describe the three-step analysis procedure used to calculate the  $B \rightarrow \pi\ell\nu$  semileptonic form factor  $f_+(q^2)$ . In the first subsection, Sec. III A, we describe how we fit the pion and  $B$ -meson 2-point correlation functions in order to determine the pion energies and  $B$ -meson mass. We use both of these quantities in the later determination of the lattice form factors  $f_{\parallel}(E_\pi)$  and  $f_{\perp}(E_\pi)$ . Next, in Sec. III B, we construct a useful ratio of the 3-point correlation function  $\langle \pi|V|B \rangle$  to the 2-point functions. We then fit this ratio to a simple plateau ansatz to extract the desired form factors. Finally, in Sec. III C, we extrapolate the form factors calculated at unphysically heavy-quark masses and finite lattice spacing to the physical light-quark masses and zero lattice spacing using NLO HMS $\chi$ PT expressions extended with next-to-next-to-leading order (NNLO) analytic terms. (We perform a simultaneous extrapolation in  $m_q$  and  $a$  and interpolation in  $E_\pi$ .) We then take the appropriate linear combination of  $f_{\parallel}(E_\pi)$  and  $f_{\perp}(E_\pi)$  to determine the desired form factor,  $f_+(q^2)$ , with statistical errors.

#### A. Two-point correlator fits

The pion and  $B$ -meson 2-point correlators obey the following functional forms:

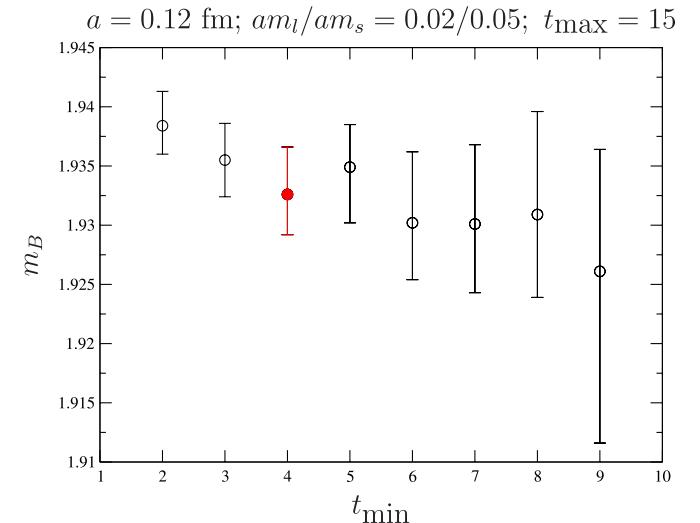
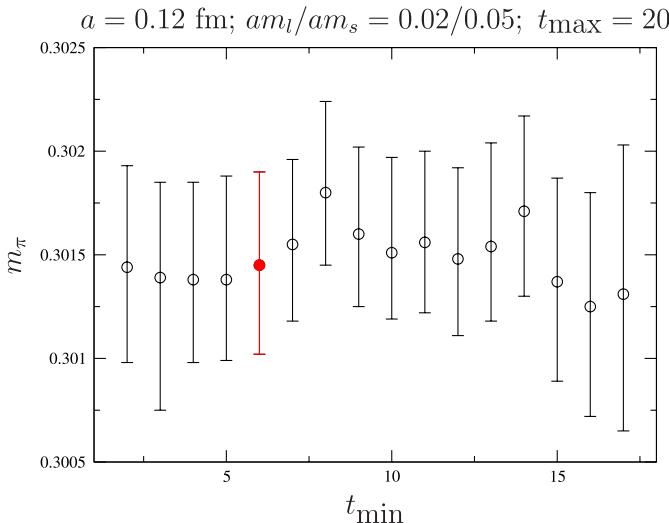


FIG. 1 (color online). Pion mass (left plot) and  $B$ -meson mass (right plot) versus minimum timeslice in 2-point correlator fit. The red (filled) data points show the fit ranges selected for use in the  $B \rightarrow \pi\ell\nu$  form-factor analysis.

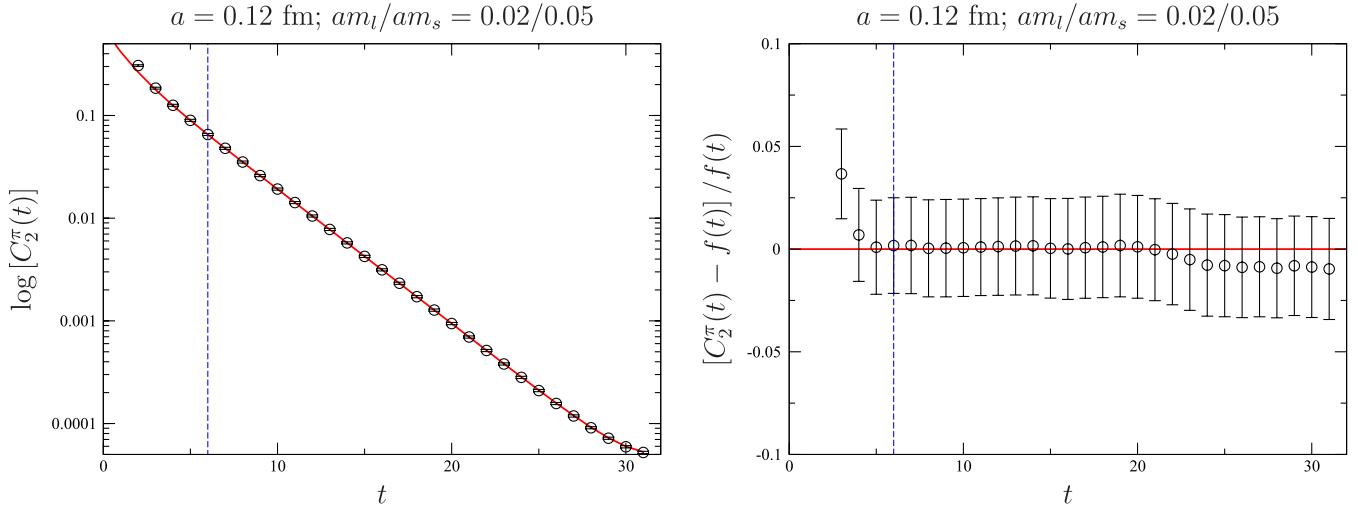


FIG. 2 (color online). Pion correlator fit corresponding to the red data point in the left-hand graph of Fig. 1. The left plot shows the fit (red line) to the zero-momentum pion propagator on a log scale, while the right plot shows the deviation of the fit from the data point for each timeslice. On both plots the dashed vertical line indicates  $t_{\min}$ . Single-elimination jackknife statistical errors are shown.

within statistical errors as a constrained fit that includes up to three or four pairs of states.

Figure 1 shows examples of both  $m_\pi$  vs  $t_{\min}$  (left plot) and  $m_B$  vs  $t_{\min}$  (right plot) on the  $am_l/am_s = 0.02/0.05$  coarse ensemble, which has the largest light-quark mass of the coarse ensembles. The masses are stable as  $t_{\min}$  is reduced, and the statistical errors in  $m_B$  become smaller as additional timeslices are added to the fit. The statistical errors are determined by performing a separate fit to 500 bootstrap ensembles; each fit uses the full single-elimination jackknife correlation matrix, which is remade before every fit. The size of the statistical errors does not change when the number of bootstrap ensembles is increased by factors of two or four. We select the time range

to use in the  $B \rightarrow \pi \ell \nu$  analysis based on several criteria: a good correlated confidence level, relatively symmetric upper and lower bootstrap error bars, no  $5\sigma$  or greater outliers in the bootstrap distribution, and no sign of excited-state contamination. The red (filled) data points in Fig. 1 mark the chosen fit ranges for the ensemble in the example plots. Figures 2 and 3 show the corresponding pion and  $B$ -meson correlator fits, respectively, which go through the data points (shown with jackknife errors) quite well.

The gauge configurations have been recorded every six trajectories, and the remaining autocorrelations between consecutive configurations cannot be neglected. We address this by averaging a block of successive configurations

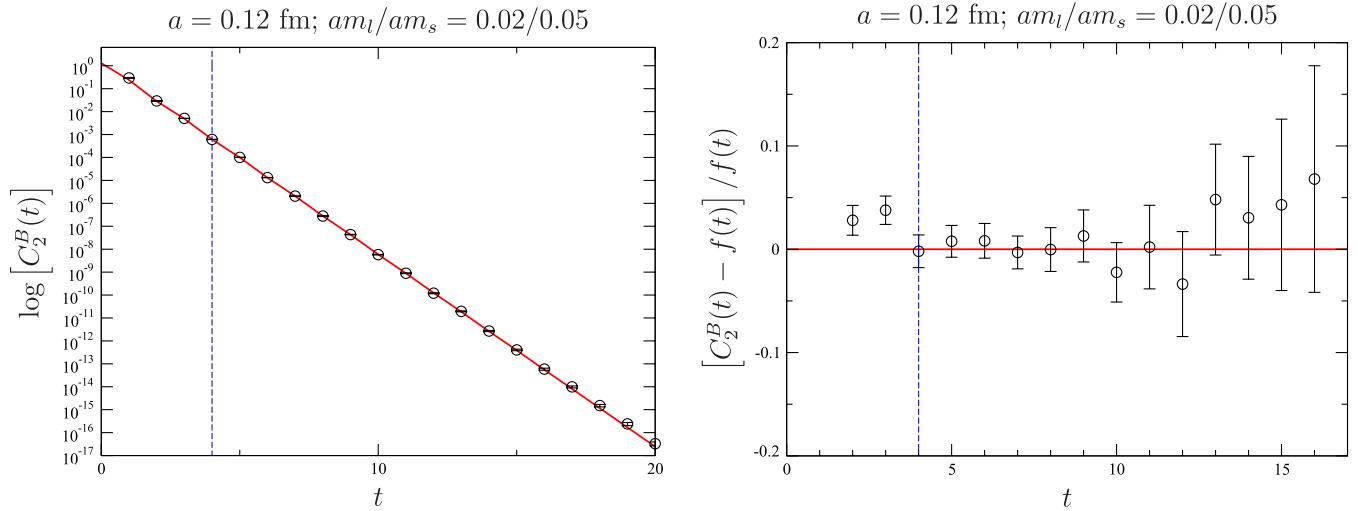


FIG. 3 (color online).  $B$ -meson correlator fit corresponding to the red data point in the right-hand graph of Fig. 1. The left plot shows the fit (red line) to the  $B$ -meson propagator on a log scale, while the right plot shows the deviation of the fit from the data point for each timeslice. On both plots the dashed vertical line indicates  $t_{\min}$ . Single-elimination jackknife statistical errors are shown.

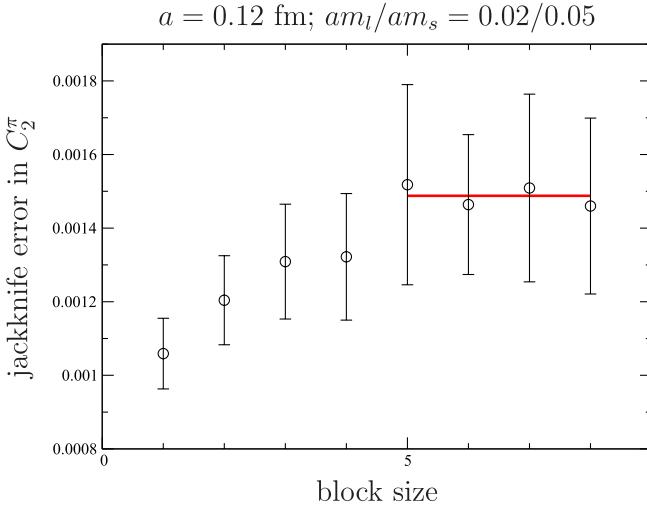


FIG. 4 (color online). Single-elimination jackknife error versus block size in the zero-momentum pion propagator at  $t = 6$ . The statistical errors in the errors are calculated with an additional single-elimination jackknife loop. The red line is an average of the errors for block sizes 5–8 and is only to make it easier to see that the statistical error plateaus after a block size of 5; it is not used in the form-factor analysis.

together before calculating the correlation matrix and performing the fit. We determine the optimal block size by increasing the number of configurations in a block until the single-elimination jackknife statistical error in the correlator data remains constant within errors. This is shown for a representative timeslice of the pion propagator on a coarse

ensemble in Fig. 4. We find that it is necessary to use a block size of 5 on the coarse ensembles and 8 on the fine ensembles, and we use these values for the rest of the form-factor analysis. We note that the size of the statistical errors that arises from blocking by 5 on the coarse ensemble is consistent with that estimated based on a calculation of the integrated autocorrelation time.

The pion energy  $E_\pi$  that is extracted from fitting the 2-point function,  $C_2^\pi(t; \vec{p}_\pi)$ , should satisfy the dispersion relation  $E_\pi^2 = |\vec{p}_\pi|^2 + m_\pi^2$  in the continuum limit due to the restoration of rotational symmetry. Similarly, the pion amplitude  $Z_\pi \equiv |\langle 0 | \mathcal{O}_\pi | \pi \rangle|$  should be independent of  $\vec{p}_\pi$  as  $a \rightarrow 0$ . As shown in Fig. 5, our results are consistent with these continuum relations within statistical errors.<sup>1</sup>

We therefore replace the pion energy  $E_\pi$  by  $\sqrt{|\vec{p}_\pi|^2 + m_\pi^2}$  when calculating the lattice form factors  $f_{\parallel}(E_\pi)$  and  $f_{\perp}(E_\pi)$  in order to reduce the total statistical uncertainty. The pion amplitude drops out of the form-factor calculation, however, because we take suitable ratios of 3-point to 2-point correlators.

## B. Three-point correlator fits

The  $B \rightarrow \pi$  3-point correlator obeys the following functional form:

$$C_{3,\mu}^{B \rightarrow \pi}(t, T) = \sum_{m,n} (-1)^{mt} (-1)^{n(T-t)} A_\mu^{mn} e^{-E_\pi^{(m)} t} e^{-m_B^{(n)}(T-t)}, \quad (31)$$

where

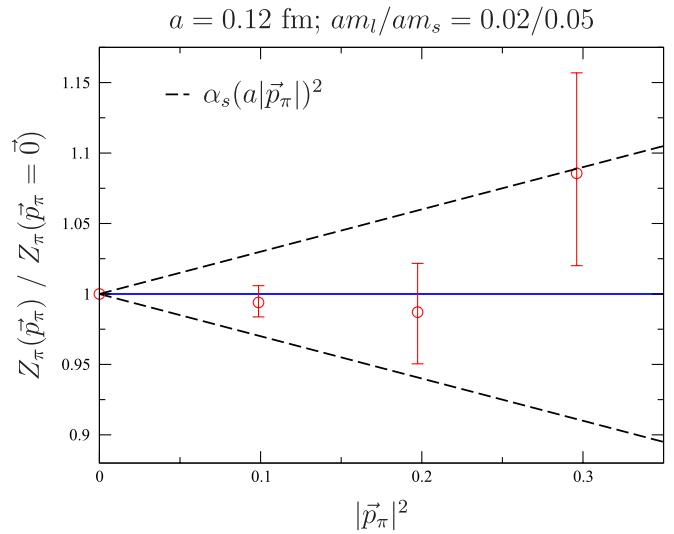
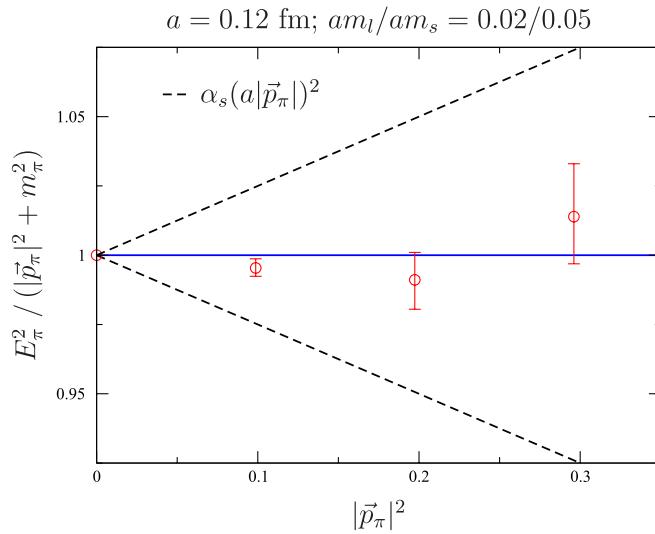


FIG. 5 (color online). Comparison of pion energy  $E_\pi$  (left plot) and amplitude  $Z_\pi$  (right plot) with the prediction of the continuum dispersion relation. We also show a power-counting estimate for the size of momentum-dependent discretization errors, which are of  $\mathcal{O}(\alpha_s(a|\vec{p}_\pi|)^2)$ , as dashed black lines.

<sup>1</sup>As this analysis was being completed we generated data with 4 times the statistics on the  $am_l/am_s = 0.02/0.05$  coarse ensemble. In order to make the comparison to the continuum expectation clearer, we use the higher statistics data in Fig. 5.

$$A_\mu^{mn} \equiv \frac{\langle 0 | \mathcal{O}_\pi | \pi^{(m)} \rangle}{2E_\pi^{(m)}} \langle \pi^{(m)} | V_\mu | B^{(n)} \rangle \frac{\langle B^{(n)} | \mathcal{O}_B | 0 \rangle}{2m_B^{(n)}}. \quad (32)$$

Writing out the first four terms of  $C_{3,\mu}^{B \rightarrow \pi}(t, T)$  makes the behavior of the 3-point correlator as a function of both  $t$  and  $T$  more transparent:

$$\begin{aligned} C_{3,\mu}^{B \rightarrow \pi}(t, T) = & A_\mu^{00} e^{-E_\pi^{(0)} t} e^{-m_B^{(0)}(T-t)} \\ & + (-1)^{(T-t)} A_\mu^{01} e^{-E_\pi^{(0)} t} e^{-m_B^{(1)}(T-t)} \\ & + (-1)^t A_\mu^{10} e^{-E_\pi^{(1)} t} e^{-m_B^{(0)}(T-t)} \\ & + (-1)^T A_\mu^{11} e^{-E_\pi^{(1)} t} e^{-m_B^{(1)}(T-t)} + \dots \end{aligned} \quad (33)$$

As in the case of the pion and  $B$ -meson propagators, the leading contributions from the opposite-parity excited states (the  $A_\mu^{10}$  and  $A_\mu^{01}$  terms) change sign when  $t \rightarrow t + 1$ ; these produce visible oscillations in the correlation function along the time direction. The subleading contribution from the opposite-parity excited states (the  $A_\mu^{11}$  term), however, only changes sign when the source-sink separation is varied, e.g.,  $T \rightarrow T + 1$ ; this contribution is not as clearly visible in the data as those that oscillate with the time slice  $t$ .

The lattice form factors are related to the ground-state amplitude of the 3-point function  $C_{3,\mu}^{B \rightarrow \pi}(t, T)$  as follows:

$$f_{\parallel}^{\text{lat}} = A_0^{00} \left( \frac{2E_\pi \sqrt{2m_B}}{Z_\pi Z_B} \right), \quad (34)$$

$$f_{\perp}^{\text{lat}} = A_i^{00} \left( \frac{2E_\pi \sqrt{2m_B}}{Z_\pi Z_B} \right) \frac{1}{p_\pi^i}, \quad (35)$$

where, as before,  $Z_\pi \equiv |\langle 0 | \mathcal{O}_\pi | \pi \rangle|$  and  $Z_B \equiv |\langle 0 | \mathcal{O}_B | B \rangle|$ . The pion and  $B$ -meson energies and amplitudes are known from the 2-point fits described in the previous subsection. Thus, the goal is to determine the 3-point amplitude  $A_\mu^{00}$  for  $\mu$  along both the spatial and temporal directions.

In principle, the easiest way to determine the coefficient  $A_\mu^{00}$  is to divide the 3-point function  $C_{3,\mu}^{B \rightarrow \pi}(t, T)$  by the appropriate 2-point functions and fit to a constant (plateau) ansatz in a region of time slices  $0 \ll t \ll T$  that are sufficiently far from both the pion and  $B$ -meson sources, such that excited-state contamination can be neglected. In practice, however, oscillating excited-state contributions are significant throughout the interval between the pion and  $B$ -meson, so our raw correlator data cannot be fit to such a simple function. Therefore, we construct an average correlator in which the oscillations are reduced before performing any fits. This method for determining the form factors requires knowledge of  $E_\pi$  and  $m_B$ ; we use the values determined in the 2-point fits described in the

previous subsection and propagate the bootstrap uncertainties in order to properly account for correlations.

The final ratio of correlators used to determine  $A_\mu^{00}$  entails several pieces. To begin consider the carefully constructed average of the value of the  $B$ -meson propagator at time slice  $t$  with that at  $t + 1$ :

$$\begin{aligned} C_2^B(t) \rightarrow C_2'^B(t) = & \frac{e^{-m_B^{(0)} t}}{2} \left[ \frac{C_2^B(t)}{e^{-m_B^{(0)} t}} + \frac{C_2^B(t+1)}{e^{-m_B^{(0)}(t+1)}} \right] \\ = & \frac{Z_B^2}{2m_B^{(0)}} e^{-m_B^{(0)} t} + (-1)^t \frac{Z_B^2}{2m_B^{(1)}} e^{-m_B^{(1)} t} \\ & \times \left( \frac{1 - e^{-\Delta m_B}}{2} \right) + \dots, \end{aligned} \quad (36)$$

where  $\Delta m_B \equiv m_B^{(1)} - m_B^{(0)}$ . By removing the leading exponential behavior from the correlator *before* taking the average we suppress the leading oscillating contribution by a factor of the mass-splitting  $\Delta m_B/2$  while leaving the desired ground-state amplitude unaffected. Note also that, while this procedure affects the size of the excited-state amplitudes, it does not alter the functional form of the correlator, nor does it alter the energies in the exponentials. Therefore, the average in Eq. (36) is equivalent to using a smeared source that has a smaller coupling to the opposite-parity excited states. This averaging procedure can be iterated in order to make the oscillating terms arbitrarily small. Empirically, we find that two iterations are sufficient for all of our numerical data:

$$\begin{aligned} \bar{C}_2^B(t) \equiv & \frac{e^{-m_B^{(0)} t}}{4} \left[ \frac{C_2^B(t)}{e^{-m_B^{(0)} t}} + \frac{2C_2^B(t+1)}{e^{-m_B^{(0)}(t+1)}} + \frac{C_2^B(t+2)}{e^{-m_B^{(0)}(t+2)}} \right] \\ \approx & \frac{Z_B^2}{2m_B^{(0)}} e^{-m_B^{(0)} t} + (-1)^t \frac{Z_B^2}{2m_B^{(1)}} e^{-m_B^{(1)} t} \left( \frac{\Delta m_B^2}{4} \right) \\ & + \mathcal{O}(\Delta m_B^3). \end{aligned} \quad (37)$$

At our various light-quark masses and lattice spacings the mass splittings lie in the range  $0.1 \lesssim \Delta m_B \lesssim 0.3$  in lattice units; thus, use of the iterated average in Eq. (37) reduces the leading oscillating state amplitude by a factor of  $\sim 50\text{--}400$  such that it can be safely neglected.

In the case of the  $B \rightarrow \pi$  3-point correlation function, we wish to reduce both the oscillating contributions and the less visible nonoscillating contributions that arise from the cross term between the lowest-lying pion and  $B$ -meson opposite-parity states. If these contributions are reduced sufficiently, we can safely neglect all of them when extracting the ground-state amplitude  $A_\mu^{00}$ . We therefore construct a slightly more sophisticated average, which combines the correlator both at consecutive time slices ( $t$  and  $t + 1$ ) and at consecutive source-sink separations ( $T$  and  $T + 1$ ):

$$\begin{aligned} \bar{C}_{3,\mu}^{B \rightarrow \pi}(t, T) &= \frac{e^{-E_\pi^{(0)} t} e^{-m_B^{(0)}(T-t)}}{8} \left[ \frac{C_{3,\mu}^{B \rightarrow \pi}(t, T)}{e^{-E_\pi^{(0)} t} e^{-m_B^{(0)}(T-t)}} + \frac{C_{3,\mu}^{B \rightarrow \pi}(t, T+1)}{e^{-E_\pi^{(0)}(t+1)} e^{-m_B^{(0)}(T+1-t)}} + \frac{2C_{3,\mu}^{B \rightarrow \pi}(t+1, T)}{e^{-E_\pi^{(0)}(t+1)} e^{-m_B^{(0)}(T-t-1)}} \right. \\ &\quad \left. + \frac{2C_{3,\mu}^{B \rightarrow \pi}(t+1, T+1)}{e^{-E_\pi^{(0)}(t+2)} e^{-m_B^{(0)}(T-t)}} + \frac{C_{3,\mu}^{B \rightarrow \pi}(t+2, T)}{e^{-E_\pi^{(0)}(t+2)} e^{-m_B^{(0)}(T-t-2)}} + \frac{C_{3,\mu}^{B \rightarrow \pi}(t+2, T+1)}{e^{-E_\pi^{(0)}(t+2)} e^{-m_B^{(0)}(T-t-1)}} \right] \\ &\approx A_\mu^{00} e^{-E_\pi^{(0)} t} e^{-m_B^{(0)}(T-t)} + (-1)^T A_\mu^{11} e^{-E_\pi^{(1)} t} e^{-m_B^{(1)}(T-t)} \left( \frac{\Delta m_B}{2} \right) + \mathcal{O}(\Delta E_\pi^2, \Delta E_\pi \Delta m_B, \Delta m_B^2). \end{aligned} \quad (38)$$

This average reduces the unwanted parity states' contamination significantly. It eliminates both the leading  $\mathcal{O}(1)$  and subleading  $\mathcal{O}(\Delta E_\pi)$  contributions to the oscillating  $A^{10}$  term, the two lowest-order  $\mathcal{O}(1, \Delta m_B)$  contributions to the oscillating  $A^{01}$  term, and the  $\mathcal{O}(1, \Delta E_\pi)$  contributions to the nonoscillating  $A^{11}$  term. The size of the remaining  $A^{11}$  term is a factor of  $\sim 7\text{--}20$  times smaller than in the unsmeared 3-point correlator.

We can now safely ignore contamination from opposite-parity states and determine the lattice form factors in a simple manner. We construct the following ratio of the smeared correlators:

$$\bar{R}_{3,\mu}^{B \rightarrow \pi}(t, T) \equiv \frac{\bar{C}_{3,\mu}^{B \rightarrow \pi}(t, T)}{\sqrt{\bar{C}_2^\pi(t) \bar{C}_2^B(T-t)}} \sqrt{\frac{2E_\pi}{e^{-E_\pi^{(0)} t} e^{-m_B^{(0)}(T-t)}}}. \quad (39)$$

The lattice form factors are then

$$f_{\parallel}^{\text{lat}} = \bar{R}_{3,0}^{B \rightarrow \pi}(t, T), \quad (40)$$

$$f_{\perp}^{\text{lat}} = \frac{1}{p_\pi^i} \bar{R}_{3,i}^{B \rightarrow \pi}(t, T). \quad (41)$$

We fit  $f_{\parallel}^{\text{lat}}$  and  $f_{\perp}^{\text{lat}}$  as defined in Eqs. (40) and (41) to a plateau in the region  $0 \ll t \ll T$ , where ordinary excited-state contributions can be neglected. Figure 6 shows the determinations of  $f_{\parallel}^{\text{lat}}$  (left plot) and  $f_{\perp}^{\text{lat}}$  (right plot) for all of the momenta that we use in the chiral extrapolation on the coarse ensemble with  $am_l/am_s = 0.02/0.05$ . In practice, we fit a range of four time slices, choosing the interval that results in the best correlated confidence level. We have cross-checked the determination of the form factors via Eqs. (40) and (41) against determinations of the form factor that explicitly include excited-state dependence in the fit ansatz and find that the results agree within errors. Our preferred method, however, yields the smaller statistical uncertainty in the form factors.

### C. Continuum and chiral extrapolation

The quark masses in our numerical lattice simulations are heavier than the physical up and down quark masses. The effects of nonzero lattice spacings in Asqtad simulations are also too large to be neglected. In order to account for these facts, we calculate the desired hadronic matrix elements for multiple values of the light-quark masses and lattice spacing, and then extrapolate to the physical quark masses and continuum using functional forms from heavy-

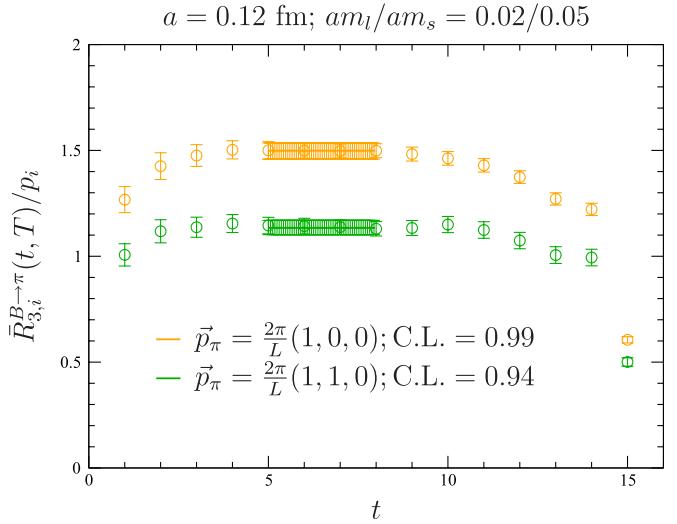
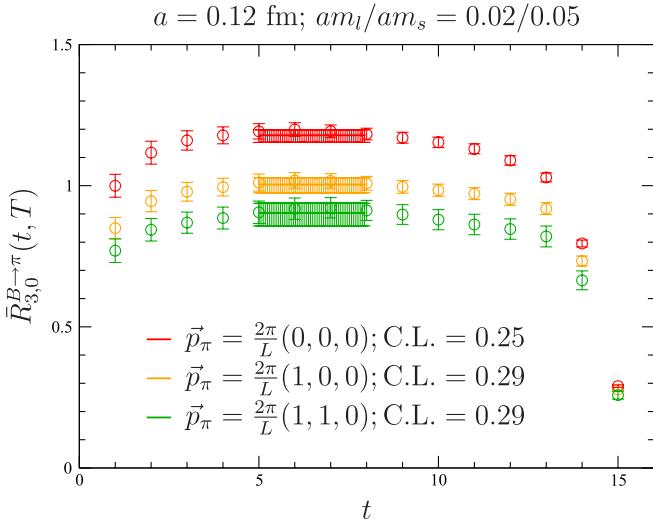


FIG. 6 (color online). Determination of the form factors  $f_{\parallel}$  (left plot) and  $f_{\perp}$  (right plot) from plateau fits to the ratios defined in Eqs. (40) and (41). The statistical errors on the data points are from a single-elimination jackknife. The statistical errors in the plateau determination are from separate fits of 500 bootstrap ensembles.

light meson staggered chiral perturbation theory (HMS $\chi$ PT) [41]. The HMS $\chi$ PT expressions are derived using the symmetries of the staggered lattice theory, and therefore contain the correct dependence of the form factors on the quark mass and lattice spacing. In the case of the  $B \rightarrow \pi\ell\nu$  form factors, the HMS $\chi$ PT expressions are also functions of the pion energy (recall that we work in the frame where the  $B$  meson is at rest).

HMS $\chi$ PT is a systematic expansion in inverse powers of the heavy-quark mass. In the chiral and soft pion limits ( $m_l \rightarrow 0$  and  $E_\pi \rightarrow 0$ ), the leading-order continuum HM $\chi$ PT expressions for  $f_{\parallel}$  and  $f_{\perp}$  take the following simple forms:

$$f_{\parallel}(E_\pi) = \frac{\phi_B}{f_\pi}, \quad (42)$$

$$f_{\perp}(E_\pi) = \frac{\phi_{B^*}}{f_\pi} \frac{g_{B^*B\pi}}{E_\pi + \Delta_B^*}, \quad (43)$$

where  $\phi_B \equiv f_B \sqrt{m_B}$ ,  $f_B$  is the  $B$ -meson decay constant, and  $f_\pi$  is the pion decay constant. The coefficient  $g_{B^*B\pi}$  parameterizes the size of the  $B^*$ - $B$ - $\pi$  coupling. In the static heavy-quark limit, heavy-quark spin symmetry does not distinguish between the pseudoscalar  $B$  meson and the vector  $B^*$  meson, which implies that the decay constant  $\phi_{B^*} = \phi_B$  and the mass difference  $\Delta_B^* \equiv m_{B^*} - m_B \rightarrow 0$ . Inclusion of the parameter  $\Delta_B^*$ , however, ensures the proper location of the pole at  $m_{B^*}^2$  in the physical form factor  $f_+(q^2)$ . At the next order in the heavy-quark expansion,  $\mathcal{O}(1/m_b)$  corrections split the degeneracy between the  $B$ - and  $B^*$ -meson masses and decay constants. Furthermore, in the chiral and soft pion limits, all  $1/m_b$  corrections can be absorbed into the values of the parameters  $\phi_B$ ,  $\phi_{B^*}$ ,  $g_{B^*B\pi}$ , and  $\Delta_B^*$  [65]; thus,  $f_{\parallel}$  and  $f_{\perp}$  retain the functional forms in Eqs. (42) and (43) even at NLO in HM $\chi$ PT.

At lowest order in S $\chi$ PT, discretization effects split the degeneracies among the 16 tastes of pseudo-Goldstone mesons

$$m_{xy,\Xi}^2 = \mu(m_x + m_y) + a^2 \Delta_{\Xi}, \quad (44)$$

where  $x$  and  $y$  indicate the quark flavors,  $\mu$  is a continuum low-energy constant, and  $\Delta_{\Xi}$  is the mass splitting of a meson with taste  $\Xi$ . An exact  $U(1)_A$  symmetry protects the taste-pseudoscalar meson from receiving a mass shift to all orders in S $\chi$ PT, implying that  $\Delta_P = 0$ . In addition, at  $\mathcal{O}(a^2)$ , a residual  $SO(4)$  taste symmetry preserves the degeneracies among mesons that are in the same irreducible representation:  $P, V, A, T, I$  [26]. Numerically, the size of the taste splittings turn out to be comparable to those of the pion masses for the  $a = 0.09$  fm and  $a = 0.12$  fm Asqtad staggered lattices used in this work [16].

We extrapolate our numerical form-factor data using HMS $\chi$ PT expressions derived to zeroth order in  $1/m_b$ .

The fit functions therefore depend upon the three remaining expansion parameters:  $m_l$ ,  $a$ , and  $E_\pi$ . The HMS $\chi$ PT expressions for the form factors to  $\mathcal{O}(m_l, a^2, E_\pi^2)$  are given explicitly in Eqs. (65)–(67) of Ref. [41]. Schematically, they read

$$\begin{aligned} f_{\parallel}(m_l, E_\pi, a) &= \frac{c_{\parallel}^{(0)}}{f_\pi} [1 + \text{logs} + c_{\parallel}^{(1)} m_l + c_{\parallel}^{(2)} (2m_l + m_s) \\ &\quad + c_{\parallel}^{(3)} E_\pi + c_{\parallel}^{(4)} E_\pi^2 + c_{\parallel}^{(5)} a^2], \end{aligned} \quad (45)$$

$$\begin{aligned} f_{\perp}(m_l, E_\pi, a) &= \frac{c_{\perp}^{(0)}}{f_\pi} \left[ \frac{1}{E_\pi + \Delta_B^* + \text{logs}} + \frac{1}{E_\pi + \Delta_B^*} \right. \\ &\quad \times \text{logs} \left. \right] + \frac{c_{\perp}^{(0)}/f_\pi}{E_\pi + \Delta_B^*} \\ &\quad \times [c_{\perp}^{(1)} m_l + c_{\perp}^{(2)} (2m_l + m_s) \\ &\quad + c_{\perp}^{(3)} E_\pi + c_{\perp}^{(4)} E_\pi^2 + c_{\perp}^{(5)} a^2], \end{aligned} \quad (46)$$

where “logs” indicate nonanalytic functions of the pseudo-Goldstone meson masses, e.g.,  $m_\pi^2 \ln(m_\pi^2/\Lambda_\chi^2)$ . The continuum low-energy constant  $g_{B^*B\pi}$  enters these expressions in the coefficients of the chiral logarithms, which are completely fixed at this order. We use the phenomenological value of  $g_{B^*B\pi} = 0.51$  [43] for the central value and vary  $g_{B^*B\pi}$  by a reasonable amount (see Sec. IV B) to estimate its contribution to the systematic uncertainty. Because the size of the mass splitting  $\Delta_B^*$  is poorly determined from the lattice data and is consistent with the physical value within statistical errors, we fix  $\Delta_B^*$  to the PDG value, 45.78 MeV [13], in our fits. The chiral logarithms also depend upon six extra constants that parameterize discretization effects due to the light staggered quarks: the four taste splittings  $a^2 \Delta_V, a^2 \Delta_A, a^2 \Delta_T, a^2 \Delta_I$  and the two flavor-neutral “hairpin” coefficients  $a^2 \delta'_V$  and  $a^2 \delta'_A$  [27]. These parameters can be determined separately from fits to light pseudoscalar meson masses and decay constants; we therefore hold them fixed to the values determined in Ref. [66], while performing the continuum-chiral extrapolation. The variation of these parameters within their statistical errors results in a negligible change to the extrapolated form factors. The five terms analytic in  $m_l$ ,  $a^2$ , and  $E_\pi$  absorb the dependence upon the scale in the chiral logarithms,  $\Lambda_\chi$ , such that the form factor is scale independent. We leave the tree-level coefficients  $c_{\parallel,\perp}^{(0)}$  and the NLO analytic term coefficients  $c_{\parallel,\perp}^{(1)} - c_{\parallel,\perp}^{(5)}$  as free parameters to be determined via the fit to the lattice form-factor data. In practice, we omit the analytic term proportional to  $(2m_l + m_s)$  from our fits because the strange sea-quark mass is tuned to approximately the same value on each of our ensembles and we have simulated only full QCD points. This term is therefore largely indistinguishable from the analytic term proportional to  $m_l$ . We have checked that omission of the sea-quark mass

analytic term has a negligible impact on the form factors in the chiral and continuum limits.

In both earlier unquenched analyses of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor [33,67], the chiral extrapolation is performed as a two-step procedure: first interpolate the lattice data to fiducial values of  $E_\pi$  and then extrapolate the results to the physical quark masses and continuum independently at each value of  $E_\pi$ . The function used for the interpolation (which is different in the two analyses) introduces a systematic uncertainty that is difficult to estimate. In both cases, the chiral-continuum extrapolation makes use of the correct functional forms derived in HMS $\chi$ PT, but, by extrapolating the results for each value of  $E_\pi$  separately, the constraint that the low-energy constants of the chiral effective Lagrangian are independent of the pion energy is lost. This omission of valuable information about the form-factor shape introduces a further error that is unnecessary. The new analysis presented here instead employs a simultaneous fit using HMS $\chi$ PT to our entire data set (all values of  $m_l$ ,  $a$ , and  $E_\pi$ ) to extrapolate to physical quark masses and the continuum and interpolate in the pion energy [68]. This improved method eliminates the systematic uncertainties introduced in the two-step interpolate-then-extrapolate procedure, and exploits the available information in an optimal way.

We perform our combined chiral and continuum extrapolation using the method of constrained curve fitting [69]. Although we know that lattice data generated with sufficiently small quark masses and fine lattice spacings, and, in the case of the  $B \rightarrow \pi\ell\nu$  form factor, sufficiently low pion energies, must be described by lattice  $\chi$ PT, we do not know precisely the range of validity of the effective theory. Furthermore, the order in  $\chi$ PT to which we must work and the allowed parameter values depend upon both the quantity of interest and the size of the statistical errors. We therefore need a fitting procedure that both incorporates our general theoretical understanding of the suitable chiral effective theory and accounts for our limited knowledge of the values of the low-energy constants and sizes of the higher-order terms. Constrained curve fitting provides just such a method.

Next-to-leading order  $\chi$ PT breaks down for pion energies around and above the kaon mass. Less than half of our numerical form-factor data, however, is below this cutoff. Therefore, although we do not expect NLO HMS $\chi$ PT to describe our data through momentum  $p = 2\pi(1, 1, 0)/L$ , we cannot remove those points without losing the majority of our data. Nor can we abandon the NLO HMS $\chi$ PT expressions for  $f_{\parallel}$  and  $f_{\perp}$ , Eqs. (45) and (46), which are the only effective field theory guides that we have for extrapolating the numerical lattice form-factor data to the continuum and physical quark masses. We therefore perform the continuum-chiral extrapolation using the full NLO HMS $\chi$ PT expressions for  $f_{\parallel}$  and  $f_{\perp}$ , including the 1-loop chiral logarithms, *plus* additional NNLO analytic

terms to allow a good fit to the data through  $p = 2\pi(1, 1, 0)/L$ . The NNLO terms smoothly interpolate between the region in which  $\chi$ PT is valid and the region in which the pion energies are too large and the higher-order chiral logarithms in  $E_\pi$  can be approximated as polynomials.

We express the analytic terms in the formulae for  $f_{\parallel}$  and  $f_{\perp}$ , Eqs. (45) and (46), as products of dimensionless expansion parameters

$$\chi_{m_l} = \frac{2\mu m_l}{8\pi^2 f_\pi^2} \sim 0.05\text{--}0.19, \quad (47)$$

$$\chi_{a^2} = \frac{a^2 \bar{\Delta}}{8\pi^2 f_\pi^2} \sim 0.03\text{--}0.09, \quad (48)$$

$$\chi_{E_\pi} = \frac{\sqrt{2}E_\pi}{4\pi f_\pi} \sim 0.22\text{--}0.78, \quad (49)$$

where  $\bar{\Delta}$  is the average staggered taste splitting and we show the range of values for each of these parameters corresponding to our numerical lattice data. (Note that we omit the  $\vec{p} = 2\pi(1, 1, 1)/L$  data points from our chiral fits because these would lead to  $\chi_{E_\pi} \gtrsim 1$ .) Because each of the above expressions is normalized by the chiral scale  $\Lambda_\chi \approx 4\pi f_\pi$ , the undetermined coefficients  $c_{\parallel,\perp}^{(1)} - c_{\parallel,\perp}^{(5)}$  should be of  $\mathcal{O}(1)$  in these units. We therefore constrain the values of the low-energy constants  $c_{\parallel,\perp}^{(0)} - c_{\parallel,\perp}^{(5)}$  in our fits with Gaussian priors of width 2 centered about 0.

The statistical errors in the numerical lattice data come from the 3-point fits described in the previous subsection. In order to account for the correlations among the various pion energies on the same sea-quark ensemble in the chiral-continuum extrapolation, we preserve the bootstrap distributions. We perform a separate correlated fit to each of the 500 bootstrap ensembles in which we remake the full bootstrap covariance matrix for each fit. We average the 68% upper and lower bounds on the form-factor distributions to determine the statistical and systematic errors in  $f_{\parallel}$  and  $f_{\perp}$  that are plotted in Fig. 7 and presented in Table II below.

Because we do not know *a priori* how many terms are necessary to describe the available lattice data, we begin with strictly NLO fits using the formulae for  $f_{\parallel}$  and  $f_{\perp}$  in Eqs. (45) and (46). We fit the lattice data for  $f_{\parallel}$  and  $f_{\perp}$  separately even though the ratio of leading-order coefficients  $c_{\perp}^{(0)}/c_{\parallel}^{(0)}$  is predicted to be equal  $g_{B^*B\pi}$  to NLO in  $\chi$ PT; this is because the value of  $g_{B^*B\pi}$  is known to only  $\sim 50\%$  from phenomenology. We obtain a good fit of the  $f_{\perp}$  lattice data to the NLO expression without the inclusion of higher-order NNLO terms. This is probably because the shape of  $f_{\perp}$  is dominated by the  $1/(E_\pi + \Delta^*)$  behavior and therefore largely insensitive to the other terms. We cannot, however, obtain a good fit of  $f_{\parallel}$  to the strictly NLO

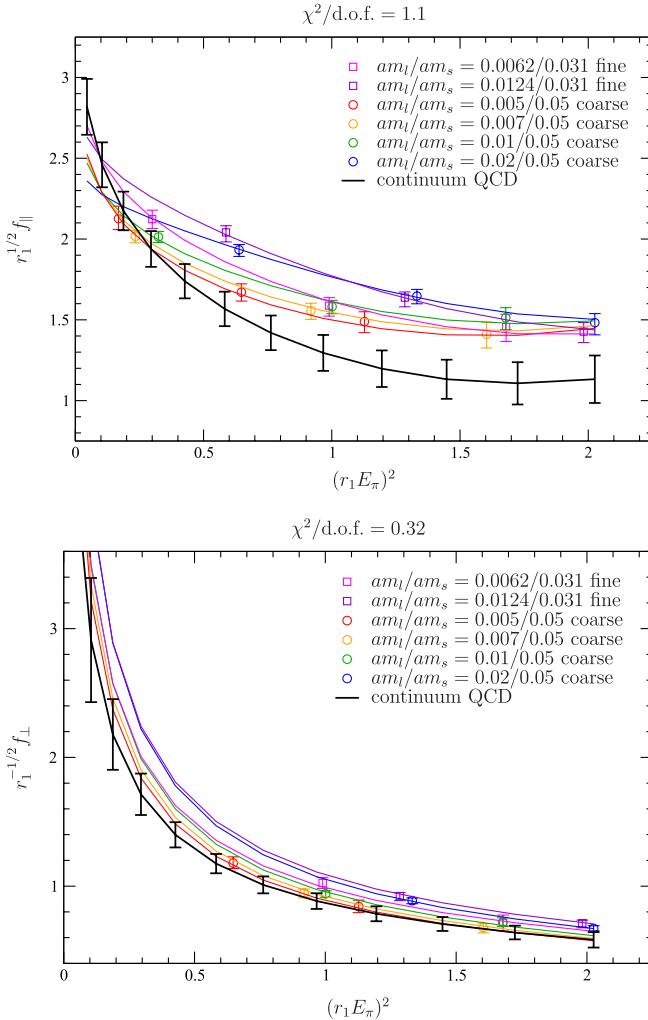


FIG. 7 (color online). Chiral-continuum extrapolation of  $f_{\parallel}$  (upper) and  $f_{\perp}$  (lower) using constrained NLO  $\text{HMS}\chi\text{PT}$  plus all NNLO analytic terms with  $g_{B^*B\pi} = 0.51$  and  $r_1 = 0.311$  fm. The square symbols indicate  $a \approx 0.09$  fm lattice data points, while the circular symbols indicate  $a \approx 0.12$  fm coarse data points. The black curve is the chiral-continuum extrapolated form-factor symmetrized bootstrap statistical errors only.

expression, and must add higher-order terms in order to obtain a successful fit. Specifically, NNLO analytic terms proportional to  $m_l E_\pi$  and  $E_\pi^3$  are both necessary to achieve a confidence level better than 10%.

Although we could, at this point, choose to truncate the  $\text{HMS}\chi\text{PT}$  extrapolation formulae to include only those terms necessary for a good confidence level, we instead include “extra” NNLO analytic terms to both the  $f_{\parallel}$  and  $f_{\perp}$  fits, constraining the values of their coefficients with Gaussian priors of  $0 \pm 2$ . The introduction of more free parameters increases the statistical errors in the extrapolated values of the form factors; these larger errors reflect the uncertainty in the size of the newly included higher-order contributions. We continue to add higher-order analytic terms until the central values of the extrapolated form

factors stabilize and the statistical errors in the form factors reach a maximum. This indicates that any further terms are of sufficiently high order, that they do not affect the fit, and can safely be neglected. We find that this occurs once the extrapolation formulae for  $f_{\parallel}$  and  $f_{\perp}$  contain all eight sea-quark mass-independent NNLO analytic terms. The inclusion of NNNLO analytic terms does not further increase the size of the error bars.

Figure 7 shows the preferred constrained fits of  $f_{\parallel}$  (upper plot) and  $f_{\perp}$  (lower plot) versus  $E_\pi^2$ , where both the  $x$  and  $y$  axes are in  $r_1$  units.<sup>2</sup> Each fit is to the NLO  $\text{HMS}\chi\text{PT}$  expression, Eqs. (45) and (46), plus all sea-quark mass-independent NNLO analytic terms. The square symbols indicate fine lattice data, while the circles denote coarse data. The six colored curves show the fit result projected onto the masses and lattice spacings of the six sea-quark ensembles; the red line should go through the red circles, and so forth. The thick black curve shows the form factor in the continuum at physical quark masses with symmetrized bootstrap statistical errors.

We use functions and constraints based on  $\text{HMS}\chi\text{PT}$  to perform the chiral-continuum extrapolation because we know that  $\text{HMS}\chi\text{PT}$  is the correct low-energy effective description of the lattice theory. Nevertheless, we must compare various properties with theoretical expectations in order to check for overall consistency. An essential first test is that we can successfully fit the data with good confidence levels and obtain low-energy coefficients that are of the predicted size. We can also verify the convergence of the series expansion by calculating the ratios of the higher-order contributions to the leading-order form-factor contributions

$$\frac{f_{\parallel}^{\text{NLO}}}{f_{\parallel}^{\text{LO}}} \Big|_{E_\pi=500 \text{ MeV}} \approx 47\%, \quad \frac{f_{\perp}^{\text{NLO}}}{f_{\perp}^{\text{LO}}} \Big|_{E_\pi=500 \text{ MeV}} \approx 48\%, \quad (50)$$

$$\frac{f_{\parallel}^{\text{NNLO}}}{f_{\parallel}^{\text{LO}}} \Big|_{E_\pi=500 \text{ MeV}} \approx 3\%, \quad \frac{f_{\perp}^{\text{NNLO}}}{f_{\perp}^{\text{LO}}} \Big|_{E_\pi=500 \text{ MeV}} \approx 4\%, \quad (51)$$

where we choose a nominal value of  $E_\pi = 500$  MeV for illustration because it is on the high end of the expected range of validity of  $\chi\text{PT}$ . Finally, because the leading-order coefficient  $c_{\parallel}^{(0)}$  is expected to be equal to  $\phi_B \equiv f_B \sqrt{m_B}$  in  $\text{HM}\chi\text{PT}$ , we can compare its value with that of  $\phi_B$  determined from our preliminary decay constant analysis.

<sup>2</sup>As a cross-check of the constrained fits, we also perform unconstrained fits of  $f_{\parallel}$  and  $f_{\perp}$  with only the minimal number of analytic terms needed for a good fit. The results are consistent, but the unconstrained fit results have smaller statistical errors because they include 6–8 fewer fit parameters.

TABLE II. Statistical and systematic error contributions to the  $B \rightarrow \pi\ell\nu$  form factor. Each source of uncertainty is discussed in Sec. IV. For each of the 12  $q^2$  bins, the error is shown as a percentage of the total form factor,  $f_+(q^2)$ , which is given in the second row from the top. Because the bootstrap errors in the form factor are asymmetric, the errors shown are the average of the upper and lower bootstrap errors. In order to facilitate the use of our result, we also present the normalized statistical and systematic bootstrap correlation matrices in Table IV and the total bootstrap covariance matrix in Table V.

$q^2$ (GeV $^2$ )	26.5	25.7	25.0	24.3	23.5	22.8	22.1	21.3	20.6	19.8	19.1	18.4
$f_+(q^2)$	9.04	6.32	4.75	3.75	3.06	2.56	2.19	1.91	1.69	1.51	1.37	1.27
statistics + $\chi$ PT(%)	24.4	18.5	13.5	9.6	7.1	6.3	6.5	6.9	7.2	7.5	8.2	9.8
$g_{B^*B\pi}$ uncertainty	1.1	0.3	0.8	1.8	2.4	2.8	2.9	2.8	2.6	2.5	2.6	2.9
$r_1$	0.4	0.7	0.9	1.1	1.2	1.3	1.4	1.4	1.5	1.5	1.4	1.4
$\hat{m}$	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3
$m_s$	0.6	0.6	0.6	0.7	0.7	0.8	0.8	0.9	1.0	1.1	1.2	1.3
$m_b$	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
Heavy-quark discretization	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4	3.4
Nonperturbative $Z_V$	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4	1.4
Perturbative $\rho$	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
$u_0$	2.9	2.1	1.2	0.5	0.3	0.8	1.1	1.3	1.4	1.3	1.3	1.3
Finite volume	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Total systematics (%)	5.9	5.4	5.3	5.4	5.7	5.9	6.0	6.1	6.0	6.0	6.0	6.2

Although the  $B$ -meson decay constant calculation uses the same staggered gauge configurations, it employs different heavy-light meson 2-point correlation functions with the axial current, a different HMS $\chi$ PT fit function, and different perturbative renormalization factors, and is therefore largely independent of the  $B \rightarrow \pi\ell\nu$  semileptonic form-factor calculation. For the preferred  $f_{\parallel}$  fit shown in Fig. 7, we find  $c_{\parallel}^{(0)} = 0.81 \pm 0.07$ , where the errors are statistical only. This is quite close to our current preliminary result,  $r_1^{3/2}\phi_B = 0.92 \pm 0.03$  (statistical error only) [70,71], especially considering that the HMS $\chi$ PT extrapolation formula for  $f_{\parallel}$  neglects some of the  $\mathcal{O}(1/m_b)$  contributions.

An interesting use of our numerical  $B \rightarrow \pi\ell\nu$  form-factor data is to determine the approximate value of the  $B^*-B-\pi$  coupling,  $g_{B^*B\pi}$ , from lattice QCD. For the preferred fits shown in Fig. 7, we find that the ratio of leading-order coefficients is

$$g_{B^*B\pi} \approx \frac{c_{\perp}^{(0)}}{c_{\parallel}^{(0)}} = 0.22 \pm 0.07, \quad (52)$$

and is independent of the choice for  $g_{B^*B\pi}$  in the chiral logarithms within statistical errors. This determination omits the  $\mathcal{O}(1/m_b)$  corrections to the chiral logarithms in the HMS $\chi$ PT extrapolation formulae for  $f_{\parallel}$  and  $f_{\perp}$ , Eqs. (45) and (46), and neglects the difference between  $\phi_B$  and  $\phi_{B^*}$ ; we do not attempt to estimate the systematic uncertainty introduced by these or other effects. The value is lower than the determination of Stewart,  $g_{B^*B\pi} = 0.51$ , which comes from a combined analysis of several experimental quantities, including the  $D^*$ -meson decay width, through  $\mathcal{O}(1/m_c)$  in HM $\chi$ PT [43]. It is consistent, however, with the range of values determined by the HPQCD Collaboration, who allowed  $g_{B^*B\pi}$  to be a free parameter

in their chiral-continuum extrapolation and found  $0 < g_{B^*B\pi} \lesssim 0.45$  [33].

#### IV. ESTIMATION OF SYSTEMATIC ERRORS

In this section, we discuss all of the sources of systematic uncertainty in our calculation of the  $B \rightarrow \pi\ell\nu$  form factor  $f_+(q^2)$ . We present each error in a separate subsection for clarity. The value of the form-factor  $f_+(q^2)$ , along with the total error budget, is given in Table II.

##### A. Chiral-continuum extrapolation fit ansatz

We use the method of constrained curve fitting to estimate the effect of neglected higher-order terms in the HMS $\chi$ PT chiral-continuum extrapolation formulae. Our fit procedure is described in detail in Sec. III C. Therefore, the errors in  $f_{\parallel}$  and  $f_{\perp}$  extrapolated to physical quark masses and zero lattice spacing shown in Fig. 7 reflect both the statistical errors in the Monte Carlo data and the systematic errors due to our limited knowledge of the higher-order terms, which we specified with priors. We do not need to include a separate systematic uncertainty due to the choice of fit function, as would be the case had we used an unconstrained fit with fewer terms.

##### B. $g_{B^*B\pi}$ uncertainty

We fix the size of the  $B^*-B-\pi$  coupling to  $g_{B^*B\pi} = 0.51$  in the coefficients of the chiral logarithms, while extrapolating to the physical light-quark masses and continuum. Our choice is based upon the following considerations: Because the coupling  $g_{B^*B\pi}$  is expected to be approximately equal to  $g_{D^*D\pi}$  due to heavy-quark symmetry, we use the phenomenological value of the  $D^*-D-\pi$  coupling determined by Stewart in Ref. [43], which comes from a

combined analysis of several experimental quantities that includes the  $D^*$ -meson decay width [72]. This value is presented without errors, and is an update of Stewart's earlier analysis in Ref. [73], which incorporates additional experimental results. His earlier calculation finds a significantly lower value of  $g_{D^*D\pi} = 0.27^{+0.04+0.05}_{-0.02-0.02}$ , where the first errors are experimental and the second errors come from an estimate of the sizes of the 1-loop counterterms [73]. A more recent phenomenological determination of the  $D^*-D-\pi$  coupling by Kamenik and Fajfer, which also includes up-to-date experimental data, improves upon the analysis method of Stewart by including contributions from both positive and negative parity heavy mesons in the loops [74]. They find an even higher value of  $g_{D^*D\pi} = 0.66^{+0.08}_{-0.06}$ , where the uncertainty only reflects the error due to counterterms. We therefore conclude that, although recent experimental measurements of the  $D^*$  width may constrain the coupling [72] at tree-level, the size of  $g_{B^*B\pi}$  is not well determined in the literature.

In order to determine the error in the form factor from the uncertainty  $g_{B^*B\pi}$  we vary the parameter over a generous range. The smallest value of  $g_{B^*B\pi}$  that we have seen in the literature is  $g_{D^*D\pi} = 0.27$  [73]. The largest is  $g_{D^*D\pi} = 0.67$ , which comes from a quenched lattice calculation [75]. (There has not yet been an unquenched “2 + 1” flavor determination of  $g_{B^*B\pi}$ .) We therefore vary  $g_{B^*B\pi}$  over the entire range from 0.27–0.67 and take the largest difference from the preferred determination of  $f_+(q^2)$  using  $g_{B^*B\pi} = 0.51$  as the systematic error due to the uncertainty in the  $B^*-B-\pi$  coupling. The lattice data is largely insensitive to the value of  $g_{B^*B\pi}$  in the coefficient of the chiral logarithms; all values of the parameter yield similar fit confidence levels. The resulting systematic uncertainty in  $f_+(q^2)$  is less than 3% for all  $q^2$  bins despite varying  $g_{B^*B\pi}$  by almost 50%.

### C. Scale ( $r_1$ ) uncertainty

We use the MILC Collaboration's determination of the scale from their calculation of  $f_\pi$ ,  $r_1 = 0.311$  fm, to convert between lattice and physical units [51]. The parameter  $r_1$  enters the form-factor calculation in a number of places: we use the PDG values of  $f_\pi$  and  $\Delta_B^*$  in the chiral-continuum extrapolation formulae [13], we set  $m_\pi$  to the PDG value in the resulting fit functions to determine the form factors at the physical point, and we convert  $f_{\parallel}$  and  $f_{\perp}$  to physical units via  $r_1$  before combining them to extract  $f_+(q^2)$ . An alternative determination using the HPQCD Collaboration's lattice data for the Y 2S-1S [76] splitting yields a result that is ∼2% larger,  $r_1 = 0.317$  fm. We therefore repeat the chiral-continuum extrapolation of  $f_{\parallel}$  and  $f_{\perp}$  using this higher value of  $r_1$ , combine them into the dimensionless form factor  $f_+(q^2)$  using this higher value of  $r_1$ , and take the difference from the preferred form-factor result as the systematic error due to uncertainty in the overall lattice scale. The difference ranges from 1–

1.5% for most  $q^2$  values. This is consistent with our naive expectation that a ∼2% difference in  $r_1$  will result in a ∼1% difference in  $f_+(q^2)$  because  $f_{\parallel}$  has dimensions of  $\text{GeV}^{1/2}$  and  $f_{\perp}$  has dimensions of  $\text{GeV}^{-1/2}$ .

### D. Light-quark mass ( $\hat{m}, m_s$ ) determinations

We obtain the form factors  $f_{\parallel}$  and  $f_{\perp}$  in continuum QCD by setting the lattice spacing to zero and the light-quark masses to their physical values in the HMS $\chi$ PT expressions, once the coefficients have been determined from fits to the numerical lattice data. We use the most recent calculations of the bare quark masses by the MILC Collaboration from fits to light pseudoscalar meson masses

$$r_1 \hat{m} \times 10^3 = 3.78(16), \quad (53)$$

$$r_1 m_s \times 10^3 = 102(4), \quad (54)$$

where  $\hat{m}$  is the average of the up and down quark masses and the quoted errors include both statistics and systematics [51]. We vary the bare light-quark mass,  $r_1 \hat{m}$ , within its stated uncertainty and take the maximal difference from the preferred form-factor result to be the systematic error; we find that the error is 0.3% or less for all values of  $q^2$ . We perform the same procedure for the bare strange quark mass, and find that the resulting error ranges from ∼0.5–1.5% over the various  $q^2$  bins.

### E. Bottom quark mass ( $m_b$ ) determination

The value of the form factor  $f_+(q^2)$  depends upon the  $b$ -quark mass, which we fix to its physical value throughout the calculation. Specifically, we first determine the value of the hopping parameter,  $\kappa$ , in the SW action for which the lattice kinetic mass agrees with the experimentally measured  $B_s$ -meson mass. We then use this tuned  $\kappa_b$  when calculating all of the 2- and 3-point heavy-light correlators needed for the  $B \rightarrow \pi \ell \nu$  form factor. With our current tuning procedure we are able to determine  $\kappa_b$  to ∼6% accuracy. This uncertainty in  $\kappa_b$  is conservative; it is primarily due to poor statistics, and will decrease considerably after the analysis of the larger data set that is currently being generated.

The uncertainty in  $\kappa_b$  produces an uncertainty in the form factor. We estimate this by calculating the form factor  $f_+$  at two additional values of  $\kappa$  (one above and one below the tuned value) on the  $am_l/am_s = 0.02/0.05$  coarse ensemble. This is sufficient because the heavy-quark mass-dependence of the form factor is largely independent of the sea-quark masses and lattice spacing. We find the largest dependence upon  $\kappa_b$  at momentum  $\vec{p} = 2\pi(1, 1, 0)/L$ , shown in Fig. 8, for which a 6% uncertainty in  $\kappa_b$  produces a 1.2% uncertainty in the form factor. We therefore take 1.2% to be the systematic error in  $f_+(q^2)$  due to uncertainty in the determination of the  $b$ -quark mass.

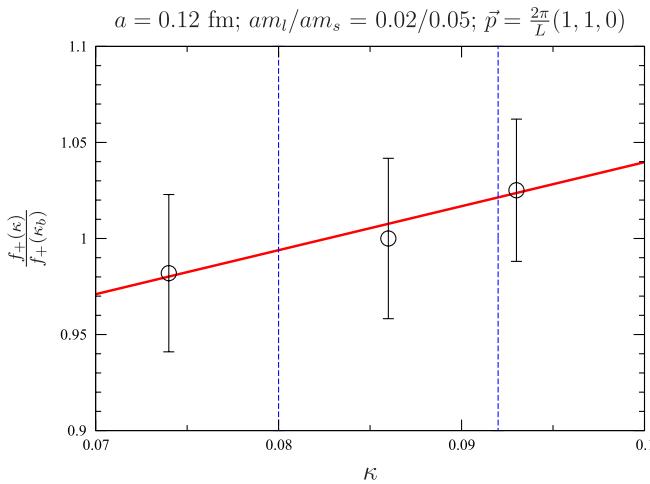


FIG. 8 (color online). Normalized form factor  $f_+$  at momentum  $\vec{p} = 2\pi(1, 1, 0)/L$  as a function of  $\kappa$  on the  $am_l/am_s = 0.02/0.05$  coarse ensemble. The central data point corresponds to the tuned  $\kappa_b$ , and the thick red line shows a linear fit to the three data points. The two dashed vertical lines indicate the upper and lower bounds on  $\kappa_b$ .

### F. Gluon and light-quark discretization errors

We estimate the size of discretization errors in the form factor  $f_+(q^2)$  with power counting. We choose conservative values for the parameters that enter the estimates:  $\Lambda = 700$  MeV and  $\alpha_V(q^*) = 1/3$ , which is a typical value on the fine lattice spacing [57,62].

We calculate the  $B \rightarrow \pi\ell\nu$  semileptonic form factor using a one-loop Symanzik-improved gauge action for the gluons [77–80] and the Asqtad-improved staggered action for the light up, down, and strange quarks [81,82]. Because both the gluon and light-quark actions are  $\mathcal{O}(a^2)$  improved, the leading discretization effects are of  $\mathcal{O}(\alpha_s(a\Lambda)^2)$ . We parameterize these errors in the fit to numerical lattice form-factor data by including analytic terms proportional to  $a^2$  in the HMS $\chi$ PT extrapolation formulae for  $f_{\parallel}$  and  $f_{\perp}$ . Because we only have data at two lattice spacings, however, we do not include a separate term proportional to  $\alpha_s(a\Lambda)^2$  to account for the fact that  $\alpha_s$  differs by a few percent between the lattice spacings. We then remove the majority of light-quark and gluon discretization effects from the final result by taking  $a \rightarrow 0$ . Similarly, we identify and remove higher-order discretization effects in the chiral-continuum extrapolation through the NNLO analytic terms in the fit functions. The remaining gluon and light-quark discretization errors are negligible.

The calculation of the  $B \rightarrow \pi\ell\nu$  form factor requires 2-point and 3-point functions with nonzero momenta, which introduces momentum-dependent discretization errors. The leading  $p$ -dependent discretization error is of  $\mathcal{O}(\alpha_s(ap)^2)$ . We parameterize these errors, up to variations in  $\alpha_s$  at the two lattice spacings, with the two NNLO analytic terms proportional to  $a^2 E_\pi^2$  and  $a^2 m_l$  in the ex-

trapolation formulae for  $f_{\parallel}$  and  $f_{\perp}$  and remove them by taking the continuum limit of the resulting fit functions. This also largely removes errors of  $\mathcal{O}(\alpha_s^2(ap)^2)$ . The remaining momentum-dependent discretization effects are of  $\mathcal{O}(\alpha_s(ap)^4)$ . On the  $28^3$  fine lattices,  $\alpha_s(ap)^4 = 0.003$  for our highest-momentum data points with  $a\vec{p} = 2\pi(1, 1, 0)/28$ . Therefore, the uncertainty in  $f_+(q^2)$  due to momentum-dependent discretization effects is negligible compared with our other systematic errors.

### G. Heavy-quark discretization errors

We use HQET as a theory of cutoff effects to estimate the size of discretization errors due to use of the Fermilab action for the heavy bottom quark. Because both the lattice and continuum theories can be described by HQET, heavy-quark discretization effects can be classified as a short-distance mismatch of higher-dimension operators [36–38]. Each contribution to the error is given by [83]

$$\text{error}_i = |[\mathcal{C}_i^{\text{lat}}(m_Q, m_0 a) - \mathcal{C}_i^{\text{cont}}(m_Q)]\langle \mathcal{O}_i \rangle|, \quad (55)$$

where  $\mathcal{O}_i$  is an effective operator, and  $\mathcal{C}_i^{\text{lat}}(m_Q, m_0 a)$  and  $\mathcal{C}_i^{\text{cont}}(m_Q)$  are the corresponding short-distance coefficients when HQET is used to describe lattice gauge theory or continuum QCD, respectively. The coefficient mismatch can be written as

$$\mathcal{C}_i^{\text{lat}}(m_Q, m_0 a) - \mathcal{C}_i^{\text{cont}}(m_Q) = a^{\dim \mathcal{O}_i - 4} f_i(m_0 a), \quad (56)$$

and the relative error in our matrix elements can be estimated by setting  $\langle \mathcal{O}_i \rangle \sim \Lambda_{\text{QCD}}^{\dim \mathcal{O}_i - 4}$ . Then each contribution to the error is

$$\text{error}_i = f_i(m_0 a)(a \Lambda_{\text{QCD}})^{\dim \mathcal{O}_i - 4}, \quad (57)$$

recovering the counting in powers of  $a$  familiar from Symanzik, while maintaining the full  $m_0 a$  dependence. The functions  $f_i$  can be deduced from Refs. [35,84] and are compiled in Appendix A. Adding all contributions of  $\mathcal{O}(\alpha_s a)$  and  $\mathcal{O}(a^2)$  from the action and the current, we obtain a relative error of 2.84% (4.16%) for  $f_{\parallel}$  and 3.40% (4.98%) for  $f_{\perp}$  on the fine (coarse) lattices. We therefore take 3.4% to be the error in  $f_+(q^2)$  due to heavy-quark discretization effects.

### H. Heavy-light current renormalization

We determine the majority of the heavy-light current renormalization nonperturbatively. The dependence of  $Z_V^{bb}$  on the sea-quark masses and on the mass of the light spectator quark in the 3-point correlator are both negligible; the statistical error in  $Z_V^{bb}$  is  $\sim 1\%$ . The dependence of  $Z_V^{ll}$  on the sea-quark masses is also negligible, and the statistical error in  $Z_V^{ll}$  is  $\sim 1\%$ . We therefore include  $\sqrt{(1\%)^2 + (1\%)^2} \approx 1.4\%$  as the systematic uncertainty in

the form-factor  $f_+(q^2)$  due to the uncertainty in the non-perturbative renormalization factors  $Z_V^{bb}$  and  $Z_V^{ll}$  for all values of  $q^2$ .

We determine the remaining renormalization of the heavy-light current using lattice perturbation theory. The 1-loop correction to  $f_\perp$  is  $\sim 3\%$  on the fine ensembles and  $\sim 4\%$  on the coarse ensembles. Because we calculate  $\rho_{V_\mu}^{hl}$  to  $\mathcal{O}(\alpha_s)$ , the leading corrections are of  $\mathcal{O}(\alpha_s^2)$ . We might therefore expect the 2-loop corrections to  $\rho_{V_\mu}^{hl}$  to be a factor of  $\alpha_s$  smaller, or  $\sim 1\%$ . In order to be conservative, however, we take the entire size of the 1-loop correction on the fine lattices, or 3%, as the systematic uncertainty in  $f_+(q^2)$  due to higher-order perturbative contributions for all  $q^2$  bins.

### I. Tadpole parameter ( $u_0$ ) tuning

In order to improve the convergence of lattice perturbation theory, we use tadpole-improved actions for the gluons, light quarks, and heavy quarks [47]. We take  $u_0$  from the average plaquette for the gluon and sea-quark action [18]. On the fine lattice, we make the same choice for the valence quarks. For historical reasons, however, we use  $u_0$  determined from the average link in Landau gauge for the valence quarks on the coarse ensembles. The difference between  $u_0$  from the two methods is 3–4% on the coarse ensembles. We must, therefore, estimate the error in the form factor due to this poor choice of tuning.

The tadpole-improvement factor enters the calculation of  $f_+(q^2)$  in several ways. The factor of  $u_0$  that enters the normalization of the heavy Wilson and light staggered quark fields cancels exactly between the  $\langle \pi | V^\mu | B \rangle$  lattice matrix element and the nonperturbative renormalization factor  $\sqrt{Z_V^{bb} Z_V^{ll}}$ . The most significant effect of the mixed  $u_0$  values is in the chiral-continuum extrapolation of  $f_\parallel$  and  $f_\perp$ . The different choices for valence and sea-quark actions imply that the coarse lattice data is partially quenched. We study this effect by performing the chiral extrapolation in two ways: one assuming that both valence and sea quarks have the mass of the sea quark and the other assuming that both have the mass of the valence quark. This leads to a 3% error in the highest  $q^2$  bin, and a  $\sim 1$ –1.5% error in the bins that affect the determination of  $|V_{ub}|$ . Most of the other effects of changing  $u_0$  in the lattice action and current can be absorbed into our estimate of the uncertainty from higher-order perturbative corrections to  $\rho_V^{hl}$ , to discretization errors, and to the normalization of the Naik term. All but the last are already budgeted in Table II. The Naik term in the Asqtad action ensures that the leading discretization errors in the pion dispersion relation are  $\mathcal{O}(\alpha_s a^2 p^2)$ . We therefore estimate the error in  $f_+(q^2)$  due to different Naik terms in the valence and sea sectors to be equal to the largest value of  $\alpha_s a^2 p^2$  on the coarse lattice times the ratio of the Landau link over plaquette  $u_0$  cubed, or  $\sim 0.2\%$ .

We add the flat error from the Naik term to the bin-by-bin error due to the light-quark mass used in the chiral extrapolation in quadrature to obtain the total uncertainty. Although our estimate is of necessity rather rough, we find that the errors due to  $u_0$  tuning are much smaller than the dominant errors in  $f_+(q^2)$ . Our error estimate is therefore adequate for the determinations of the  $B \rightarrow \pi \ell \nu$  form factor and  $|V_{ub}|$  presented in this work.

### J. Finite-volume effects

We estimate the uncertainty in the form factor  $f_+(q^2)$  due to finite-volume effects using 1-loop finite volume HMS $\chi$ PT. The finite-volume corrections to the HMS $\chi$ PT expressions for  $f_\parallel$  and  $f_\perp$  are given in Ref. [41] in terms of integrals calculated in Ref. [85]. It is therefore straightforward to find the relevant corrections for our simulation parameters. We find that the 1-loop finite-volume corrections are well below a percent for all of our lattice data points. Because finite-volume errors increase as the light-quark mass decreases, they are largest on the  $am_l/am_s = 0.007/0.05$  coarse ensemble. The biggest correction is to  $f_\perp$  at  $\vec{p} = 2\pi(1, 1, 0)/L$ , and is 0.5%. We therefore take this to be the uncertainty in  $f_+(q^2)$  due to finite-volume errors for all  $q^2$  bins.

## V. MODEL-INDEPENDENT DETERMINATION OF $|V_{ub}|$

It is well established that analyticity, crossing symmetry, and unitarity largely constrain the possible shapes of semi-leptonic form factors [86–89]. In this section we apply constraints based on these general properties to our lattice result for the form factor  $f_+(q^2)$  and thereby extract a model-independent value for the CKM matrix element  $|V_{ub}|$ .

Until now the standard procedure used to extract  $|V_{ub}|$  from  $B \rightarrow \pi \ell \nu$  semileptonic decays has been to integrate the form factor  $|f_+(q^2)|^2$  over a region of  $q^2$ , and then combine the result with the experimentally measured decay rate in this region:

$$\frac{\Gamma(q_{\min})}{|V_{ub}|^2} = \frac{G_F^2}{192\pi^3 m_B^3} \int_{q_{\min}^2}^{q_{\max}^2} dq^2 [(m_B^2 + m_\pi^2 - q^2)^2 - 4m_B^2 m_\pi^2]^{3/2} |f_+(q^2)|^2. \quad (58)$$

The integration, however, necessitates a continuous parameterization of the form factor over the full range from  $q_{\min}^2$  to  $q_{\max}^2$ .

In our earlier, preliminary unquenched analysis, we determine  $f_+(q^2)$  by fitting the lattice data points to the Bećirević-Kaidalov (BK) parameterization [90],

$$f_+(q^2) = \frac{f_+(0)}{(1 - \tilde{q}^2)(1 - \alpha \tilde{q}^2)}, \quad (59)$$

$$f_0(q^2) = \frac{f_+(0)}{(1 - \tilde{q}^2/\beta)}, \quad (60)$$

where  $\tilde{q}^2 \equiv q^2/m_{B^*}^2$ . The BK ansatz contains three free parameters and incorporates many of the known properties of the form factor such as the kinematic constraint at  $q^2 = 0$ , heavy-quark scaling, and the location of the  $B^*$  pole. The HPQCD Collaboration instead uses the four-parameter Ball-Zwicky (BZ) parameterization [91], which is the same as the BK function in Eq. (59) plus an additional pole to capture the effects of multiparticle states. In both cases, however, the choice of fit function introduces a systematic uncertainty that is difficult to quantify.

It is likely the BK and BZ parameterizations can be safely used to interpolate between data points, whether they be at high  $q^2$  from lattice QCD or at low  $q^2$  from experiment. It is less clear, however, how well these ansatze can be trusted to extrapolate the form-factor shape beyond the reach of the data points. Furthermore, comparisons of lattice and experimental determinations of BK or BZ fit parameters are not necessarily meaningful. For example, if the slope parameters  $\alpha$  from experiment and lattice QCD were found to be inconsistent, we would not know whether theory and experiment disagree, or whether the parameterization is simply inadequate. A parameterization that circumvents this issue is therefore desirable. In this work we pursue an analysis based on the model-independent  $z$  parameterization, which is pedagogically reviewed in Ref. [87].

### A. Analyticity, unitarity, and heavy-quark constraints on heavy-light form factors

All form factors are analytic functions of  $q^2$  except at physical poles and threshold branch points. In the case of the  $B \rightarrow \pi l \nu$  form factors,  $f(q^2)$  is analytic below the  $B\pi$  production region except at the location of the  $B^*$  pole. The fact that analytic functions can always be expressed as convergent power series allows the form factors to be written in a particularly useful manner.

Consider mapping the variable  $q^2$  onto a new variable,  $z$ , in the following way:

$$z(q^2, t_0) = \frac{\sqrt{1 - q^2/t_+} - \sqrt{1 - t_0/t_+}}{\sqrt{1 - q^2/t_+} + \sqrt{1 - t_0/t_+}}, \quad (61)$$

where  $t_+ \equiv (m_B + m_\pi)^2$ ,  $t_- \equiv (m_B - m_\pi)^2$ , and  $t_0$  is a free parameter. Although this mapping appears complicated, it actually has a simple interpretation in terms of  $q^2$ ; this transformation maps  $q^2 > t_+$  (the production region) onto  $|z| = 1$  and maps  $q^2 < t_+$  (which includes the semileptonic region) onto real  $z \in [-1, 1]$ . In terms of  $z$ , the form factors have a simple form:

$$f(q^2) = \frac{1}{P(q^2)\phi(q^2, t_0)} \sum_{k=0}^{\infty} a_k(t_0) z(q^2, t_0)^k, \quad (62)$$

where the Blaschke factor  $P(q^2)$  is a function that contains subthreshold poles and the outer function  $\phi(q^2, t_0)$  is an arbitrary analytic function (outside the cut from  $t_+ < q^2 < \infty$ ) whose choice only affects the particular values of the series coefficients  $a_k$ .

For the case of the  $B \rightarrow \pi\ell\nu$  form factor  $f_+(q^2)$ , the Blaschke factor  $P_+(q^2) = z(q^2, m_{B^*}^2)$  accounts for the  $B^*$  pole. In this work we use the same outer function as in Ref. [43]:

$$\begin{aligned} \phi_+(q^2, t_0) = & \sqrt{\frac{3}{96\pi\chi_J^{(0)}}} (\sqrt{t_+ - q^2} + \sqrt{t_+ - t_0})(\sqrt{t_+ - q^2} \\ & + \sqrt{t_+ - t_-})^{3/2} (\sqrt{t_+ - q^2} + \sqrt{t_+})^{-5} \\ & \times \frac{(t_+ - q^2)}{(t_+ - t_0)^{1/4}}, \end{aligned} \quad (63)$$

where  $\chi_J^{(0)}$  is a numerical factor that can be calculated via the operator product expansion [88,92]. This choice of  $\phi_+(q^2, t_0)$ , when combined with unitarity and crossing-symmetry, leads to a particularly simple constraint on the series coefficients in Eq. (62). Although the  $t$  dependence of Eq. (63) appears complicated, it is designed so that the sum over the squares of the series coefficients is  $t$  independent:

$$\sum_{k=0}^{\infty} a_k^2 = \frac{1}{2\pi i} \oint \frac{dz}{z} |P(z)\phi(z)f(z)|^2 \equiv A, \quad (64)$$

where the value of the constant  $A$  depends upon the choice of  $\chi_J^{(0)}$  in Eq. (63). Because the decay process  $B \rightarrow \pi\ell\nu$  is related to the scattering process  $\ell\nu \rightarrow B\pi$  by crossing symmetry, the sum of the series coefficients is bounded by unitarity, i.e., the fact that the production rate of  $B\pi$  states is less than or equal to the production of all final states that couple to the  $b \rightarrow u$  vector current. In particular, if one chooses the numerical factor  $\chi_J^{(0)}$  to be equal to the appropriate integral of the inclusive rate  $\ell\nu \rightarrow X_b$ , the sum of the coefficients is bounded by unity:

$$\sum_{k=0}^N a_k^2 \leqslant 1, \quad (65)$$

where this constraint holds for any value of  $N$  and the “ $\leqslant$ ” symbol indicates higher-order corrections to  $\chi_J^{(0)}$  in  $\alpha_s$  and the operator product expansion.

Such higher-order corrections turn out to be negligible for the  $B \rightarrow \pi\ell\nu$  form factor because the bound in Eq. (65) is far from saturated, i.e., the sizes of the coefficients turn out to be much less than one. Becher and Hill [93] have pointed out that this is due to the fact that the  $b$ -quark mass is so large. In the heavy-quark limit, the leading contributions to the integral in Eq. (64) are of  $\mathcal{O}(\Lambda^3/m_b^3)$ , where  $\Lambda$  is a typical hadronic scale. Assuming that the ratio  $\Lambda/m_b \sim 0.1$ , the heavy-quark bound on the  $a_k$ 's is approxi-

mately 30 times more constraining than the bound from unitarity alone:

$$\sum_{k=0}^N a_k^2 \sim \left(\frac{\Lambda}{m_B}\right)^3 \approx 0.001. \quad (66)$$

We point out that the authors of Ref. [45] have recently proposed a slightly different parameterization of the  $B \rightarrow \pi\ell\nu$  form factor with a simpler choice of outer function,  $\phi = 1$ :

$$f_+(q^2) = \frac{1}{1 - q^2/m_{B^*}^2} \sum_{k=0}^{\infty} b_k(t_0) z(q^2, t_0)^k. \quad (67)$$

This choice enforces the correct scaling behavior,  $f_+(q^2) \sim 1/q^2$  as  $q^2 \rightarrow \infty$ . It leads, however, to a more complicated constraint on the series coefficients:

$$\sum_{j,k=0}^N B_{jk} b_j b_k \leq 1, \quad (68)$$

where the elements of the symmetric matrix  $B_{jk}$  are calculable functions of  $t_0$ . Because  $B \rightarrow \pi\ell\nu$  semileptonic decay is far from  $q^2 \rightarrow \infty$ , and because the unitarity bound is so far from being saturated, the choice of outer function should make a negligible impact on the resulting determination of  $|V_{ub}|$ . We therefore use the more standard outer function given in Eq. (63) because the constraint in Eq. (65) is independent of the number of terms in the power series, and is therefore simpler to implement.

The free parameter  $t_0$  can be chosen to make the maximum value of  $|z|$  as small as possible in the semileptonic region; we choose  $t_0 = 0.65t_-$  as in Ref. [43]. For  $B \rightarrow \pi\ell\nu$  semileptonic decays this maps the physical region onto

$$0 < t < t_- \mapsto 0.34 < z < 0.22. \quad (69)$$

The bound on the coefficients in the  $z$  expansion combined with the small numerical values of  $|z|$  in the physical region ensures that one needs only the first few terms in the  $z$  expansion to accurately describe the form-factor shape. Moreover, as the precision of both the lattice calculations and experimental measurements improve, one may easily include higher-order terms as needed.

## B. Determination of $|V_{ub}|$ using $z$ parameterization

In 2007 the *BABAR* Collaboration published a measurement of the shape of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor with results for 12 separate  $q^2$  bins between  $q_{\min}^2 \approx 1 \text{ GeV}^2$  and  $q_{\max}^2 \approx 24 \text{ GeV}^2$  [42]. This suggests that lattice QCD calculations are now needed primarily to provide a precise form-factor normalization at one value of  $q^2$  in order to determine  $|V_{ub}|$ . The minimal error in  $|V_{ub}|$  can, of course, still be attained by using all of the available information on the form-factor shape and normalization, pro-

vided that one analyzes the data in a model-independent way.

Because as many terms can be added as are needed to describe the  $B \rightarrow \pi\ell\nu$  form factor to the desired accuracy, use of the convergent series expansion allows for a systematically improvable determination of  $|V_{ub}|$ . We fit our lattice numerical Monte Carlo data and the 12-bin *BABAR* experimental data together to the  $z$  expansion, leaving the relative normalization factor  $|V_{ub}|$  as a free parameter to be determined by the fit. In this way we determine  $|V_{ub}|$  in an optimal, model-independent way.

We first fit the lattice numerical Monte Carlo data and the 12-bin *BABAR* experimental data *separately* to the  $z$  expansion in order to check for consistency. We use Gaussian priors with central value 0 and width 1 on each coefficient in the  $z$  expansion to impose the unitarity constraint. Although this manner of constraining the coefficients is less stringent than the strict bound given in Eq. (65), the choice does not matter because the unitarity bound is far from saturated and the individual coefficients all turn out to be much less than 1. We obtain identical fit results even when the coefficients are completely unconstrained.

The left-hand plot in Fig. 9 shows the *BABAR* measurement of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor,  $f_+(q^2)$  [42]. The right-hand plot shows the same data multiplied by the functions  $P_+(q^2)$  and  $\phi_+(q^2, t_0)$  and plotted versus the variable  $z$ . After remapping from  $q^2$  to  $z$  there is almost no curvature in the experimental data. This indicates that most of the curvature in the data is due to well-understood QCD effects that are parameterized by the functions  $P_+(q^2)$  and  $\phi_+(q^2, t_0)$ . Consequently, the experimental data is well described by a normalization ( $a_0$ ) and slope ( $a_1/a_0$ ), as shown in Fig. 9. The slope of the *BABAR* experimental  $B \rightarrow \pi\ell\nu$  form-factor data is

$$\frac{a_1}{a_0} = -1.60 \pm 0.26. \quad (70)$$

If one includes a curvature term in the  $z$  fit, the coefficient  $a_2$  is poorly determined, but is found to be negative at  $\sim 1.5\sigma$ . The value of  $a_1$  is consistent with the result of the linear fit.

Figure 10 shows the lattice determination of the  $B \rightarrow \pi\ell\nu$  semileptonic form factor,  $f_+$  vs  $q^2$  (left plot) and the remapped form factor,  $P_+ \phi_+ f_+$  vs  $z$  (right plot). As is the case for the experimental data, the shape of the lattice form factor is less striking after taking out the  $B^*$  pole and other known QCD effects. When the lattice calculation of the form factor is fit to the  $z$  parameterization, however, it determines both a slope and a curvature. One cannot, in fact, successfully fit the lattice data without including a curvature term. The slope and curvature of the lattice determination of the  $B \rightarrow \pi\ell\nu$  form factor are

$$\frac{a_1}{a_0} = -1.75 \pm 0.91, \quad (71)$$

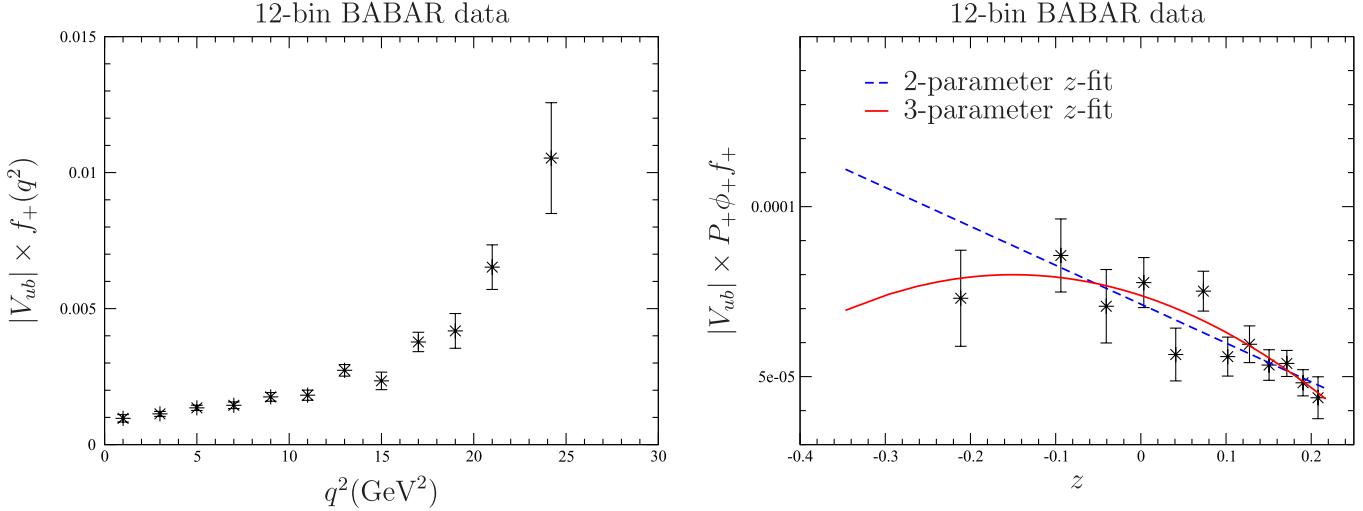


FIG. 9 (color online). Experimental data for the  $B \rightarrow \pi \ell \nu$  form factor times the CKM element  $|V_{ub}|$  from the *BABAR* Collaboration [42]. The left plot shows  $|V_{ub}| \times f_+$  versus  $q^2$ , while the right plot shows  $|V_{ub}| \times f_+$  multiplied by the functions  $P_+ \phi_+$  and plotted against the new variable  $z$ . Both the 2-parameter fit (dashed blue line) and 3-parameter fit (solid red curve) have good  $\chi^2/\text{d.o.f.}$ 's.

$$\frac{a_2}{a_0} = -5.22 \pm 1.39. \quad (72)$$

The above uncertainties are the standard errors computed from the inverse of the parameter Hessian matrix that result from a fit using the full covariance matrix determined from the bootstrap distributions of chiral-continuum extrapolated values of  $f_\parallel$  and  $f_\perp$ , including systematics.

Because the shapes of the lattice calculation and experimental measurement of the form factor are consistent, we now proceed to fit them simultaneously to the  $z$  expansion and determine  $|V_{ub}|$ . The numerical lattice and measured

experimental data are independent, so we construct a block-diagonal covariance matrix, where one block is the total lattice error matrix and the other is the total experimental error matrix. The combined fit function includes the series coefficients ( $a_k$ 's) plus an additional parameter for the relative normalization between the lattice and experimental results ( $|V_{ub}|$ ). In order to account for the systematic uncertainty in  $|V_{ub}|$  due to poorly constrained higher-order terms in  $z$ , we continue to add terms in the series until the error in  $|V_{ub}|$  reaches a maximum. This occurs once we include the term proportional to  $z^3$ . The resulting combined

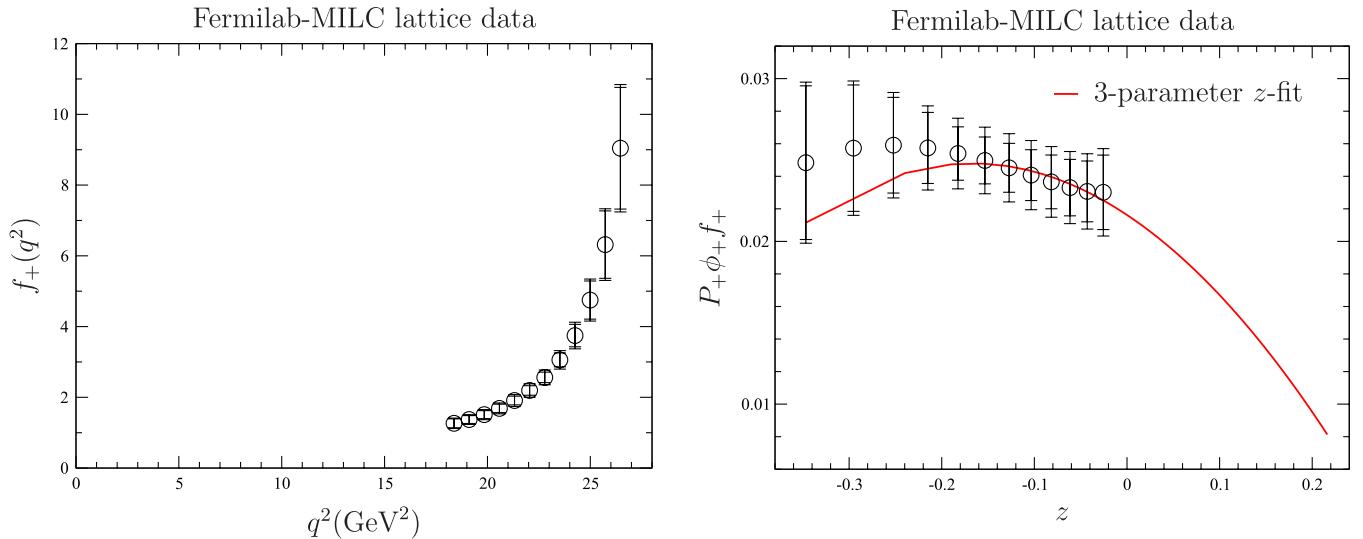


FIG. 10 (color online). Lattice calculation of the  $B \rightarrow \pi \ell \nu$  form factor. The left plot shows  $f_+$  vs  $q^2$ , while the right plot shows  $P_+ \phi_+ f_+$  vs  $z$ . The inner error bars indicate the statistical error, while the outer error bars indicate the sum of the statistical and systematic added in quadrature. A 3-parameter  $z$  fit is needed to describe the lattice data with a good  $\chi^2/\text{d.o.f.}$

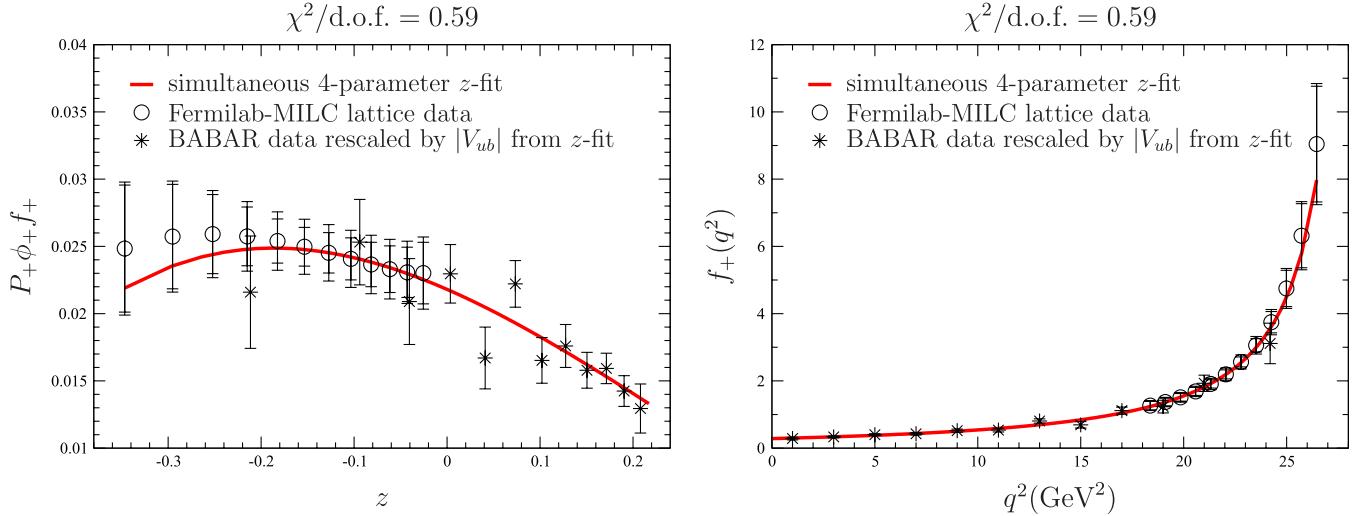


FIG. 11 (color online). Model-independent determination of  $|V_{ub}|$  from a simultaneous fit of lattice and experimental  $B \rightarrow \pi\ell\nu$  semileptonic form-factor data to the  $z$  parameterization. The left plot shows  $P_+ \phi_+ f_+$  vs  $z$ , while the right plot shows  $f_+$  vs  $q^2$ . Inclusion of terms in the power-series through  $z^3$  yields the maximum uncertainty in  $|V_{ub}|$ ; the corresponding 4-parameter  $z$  fit is given by the red curve in both plots. The circles denote the Fermilab-MILC lattice data, while the stars indicate the 12-bin *BABAR* experimental data, rescaled by the value of  $|V_{ub}|$  determined in the simultaneous  $z$ -fit.

$z$  fit is shown in Fig. 11, and the corresponding fit parameters are

$$|V_{ub}| \times 10^3 = 3.38 \pm 0.36, \quad (73)$$

$$a_0 = 0.0218 \pm 0.0021, \quad (74)$$

$$a_1 = -0.0301 \pm 0.0063, \quad (75)$$

$$a_2 = -0.059 \pm 0.032, \quad (76)$$

$$a_3 = 0.079 \pm 0.068. \quad (77)$$

The values of the coefficients are all much smaller than 1, as expected from heavy-quark power-counting. The sum of the squares of the coefficients is  $\sum a_k^2 = 0.011 \pm 0.012$ , and is consistent with the prediction of Becher and Hill within uncertainties in the series coefficients and in the choice of the hadronic scale in Eq. (66) [93].

By combining all of the available numerical lattice Monte Carlo data and 12-bin *BABAR* experimental data for the  $B \rightarrow \pi\ell\nu$  form factor in a simultaneous fit we are able to determine  $|V_{ub}|$  to  $\sim 11\%$  accuracy. This error is independent (within  $\lesssim 0.5\%$ ) of the choice of the parameter  $t_0$  used in the change of variables from  $q^2$  to  $z(q^2, t_0)$  and in the outer function  $\phi_+(q^2, t_0)$ . In order to demonstrate the advantage of the combined fit method, we compare the error in  $|V_{ub}|$  given in Eq. (73) with that obtained from separate  $z$  fits of the lattice and experimental data. A  $z$  fit to the 12-bin *BABAR* experimental data alone determines the normalization  $a_0^{\text{exp}}$  to  $\sim 8\%$ , while a  $z$  fit to our

numerical lattice data determines  $a_0^{\text{lat}}$  to  $\sim 14\%$ . Thus, separate fits lead to a determination of  $|V_{ub}| \equiv a_0^{\text{exp}}/a_0^{\text{lat}}$  with an approximately 16% total uncertainty.<sup>3</sup> The combined fit yields a significantly smaller error and is thus preferred.

When the numerical lattice data and experimental data are fit simultaneously, utilizing all of the available data points is of secondary importance for reducing the total uncertainty in  $|V_{ub}|$ . For example, we can evaluate the importance of the low  $q^2$  experimental points to the extraction of  $|V_{ub}|$  by removing them from the combined  $z$  fit. Including only the three experimental data points with  $q^2 > 18 \text{ GeV}^2$ , we find a consistent value of  $|V_{ub}|$  with only a  $\sim 1\%$  larger uncertainty. Similarly, we can evaluate the importance of having many lattice data points, rather than only a single point, by using only the most precise lattice point with a total error of  $\sim 9\%$ . This allows the form-factor shape to be completely determined by the

<sup>3</sup>Because the values of the coefficients of the power series in  $z$  depend upon the choice of the parameter  $t_0$  in Eqs. (61)–(63), we could, in principle, choose a different value of  $t_0$  in order to minimize the error in either  $a_0^{\text{exp}}$  or  $a_0^{\text{lat}}$ . For example, use of  $t_0 = 22.8 \text{ GeV}^2$  reduces the uncertainty in the lattice normalization because the error in the lattice form factor is smallest at this  $q^2$  value. Use of  $t_0 = 22.8 \text{ GeV}^2$  greatly increases the uncertainty in the experimental normalization, however, because the experimental data is poorly determined at large values of  $q^2$ . Ultimately, this choice of  $t_0$  leads to an even worse determination of  $|V_{ub}|$  than from our standard choice of  $t_0 = 0.65t_-$ . Although we did not attempt to determine the value of  $t_0$  that minimizes the total error in  $|V_{ub}|$ , the errors resulting from separate fits were greater than that obtained with the simultaneous fit for all values of  $t_0$  that we tried.

experimental data. We find a consistent value of  $|V_{ub}|$  but with an even larger error of  $\sim 13\%$ . We therefore conclude that combining all of the numerical lattice data with all of the experimentally measured *BABAR* data minimizes the total uncertainty in  $|V_{ub}|$ . Because the small error in our final determination of  $|V_{ub}|$  is primarily due to the power of the combined  $z$ -fit method, one could easily use the procedure outlined in this section to improve the exclusive determination of  $|V_{ub}|$  from existing lattice QCD calculations of the  $B \rightarrow \pi\ell\nu$  form factor such as that by the HPQCD Collaboration [33].

## VI. RESULTS AND CONCLUSIONS

Combining our latest unquenched lattice calculation of the  $B \rightarrow \pi\ell\nu$  form factor with the 12-bin *BABAR* experimental data, we find the following model-independent value for  $|V_{ub}|$ <sup>4</sup>:

$$|V_{ub}| \times 10^3 = 3.38 \pm 0.36. \quad (78)$$

The total error is  $\sim 11\%$ , and it is nontrivial to separate the error precisely into contributions from statistical, systematic, and experimental uncertainty because of the combined  $z$ -fit procedure used. If we assume, however, that the error in  $|V_{ub}|$  is dominated by the most precisely determined lattice point (which is not quite true, as shown in the previous section), we can estimate that the contributions are roughly equally divided as  $\sim 6\%$  lattice statistical,  $\sim 6\%$  lattice systematic, and  $\sim 6\%$  experimental.

Our result is consistent with, although slightly lower than, our earlier, preliminary determination of  $|V_{ub}|$ . The reduction in central value is primarily due to a change in the lattice determination of the form factor, not the procedure used to determine  $|V_{ub}|$ . Because our new analysis uses a second lattice spacing, we are able to take the continuum limit of the form factor. We find that the continuum extrapolation increases the overall normalization of  $f_+(q^2)$ , and hence decreases the value of  $|V_{ub}|$ . Our errors are smaller than those of previous exclusive determinations primarily because we have reduced the size of the discretization errors, which are significantly smaller than in the previous Fermilab-MILC calculation ( $\sim 7\% \rightarrow 3\%$ ) because of the additional finer lattice spacing.

Our new result is  $\sim 1-2\sigma$  lower than most inclusive determinations of  $|V_{ub}|$ , which typically range from  $4.0 - 4.5 \times 10^{-3}$  [32]. Much of the variation among the inclusive values is due to the choice of input parameters—in

<sup>4</sup>At three conferences during Summer 2008 we presented a version of this model-independent analysis with a numerical value for  $|V_{ub}|$  that is  $1-\sigma$  lower than that given here in Eq. (78). We have since improved several aspects of the lattice calculation, most notably reducing the statistical errors that enter the chiral and continuum extrapolations of  $f_{\perp}$  and  $f_{\parallel}$  and, hence,  $f_+$ . Equation (78) is our final result for  $|V_{ub}|$  based on the lattice data from the ensembles in Table I and the methodology of Secs. III and V.

particular that of the  $b$ -quark mass [94]. The recent determination of  $m_b$  by Kühn, Steinhauser, and Sturm using experimental data for the cross section for  $e^+e^- \rightarrow$  hadrons in the bottom threshold region yields the value of  $m_b$  to percent-level accuracy [95], and is consistent with the PDG average [13]. Neubert has shown, however, that an updated extraction of  $m_b$  from fits to  $B \rightarrow X_c\ell\nu$  moments using only the theoretically cleanest channels (excluding  $b \rightarrow X_s\gamma$ ) results in a larger  $b$ -quark mass and hence smaller inclusive value of  $|V_{ub}|$ , thereby reducing the tension between inclusive and exclusive determinations [96].

Our result is consistent with the currently preferred values for  $|V_{ub}|$  determined by the global CKM unitarity triangle analyses of the CKMfitter Collaboration,  $|V_{ub}| \times 10^3 = 3.44^{+0.22}_{-0.17}$  [97], and UTfit Collaboration,  $|V_{ub}| \times 10^3 = 3.48 \pm 0.16$ , [98]. Further reduction in the errors is therefore essential for a more stringent test of the CKM framework and a more sensitive probe of physics beyond the standard model.

The dominant uncertainty in our lattice calculation of the  $B \rightarrow \pi\ell\nu$  form factor comes from the statistical errors in the 2-point and 3-point correlations. This error can be reduced in a straightforward manner with use of an improved source for the pion and/or additional gauge configurations. The statistical errors in the nonperturbative renormalization factors  $Z_V^{bb}$  and  $Z_V^{ll}$  can be brought to below a percent in the same way. The chiral-continuum extrapolation error, which is inextricably linked to the statistical errors in the correlation functions, can also be improved by simulating at more light-quark masses and an additional finer lattice spacing of  $a \sim 0.06$  fm. Presumably a better constrained chiral and continuum extrapolation will reduce the size of other  $q^2$ -dependent errors such as those from  $g_{B^*B\pi}$ ,  $r_1$ , and the light-quark masses by some unknown amount as well. Use of a finer lattice with  $a \sim 0.06$  fm will further decrease the momentum-dependent and heavy-quark discretization errors, which we now estimate with power counting. The extraction of  $|V_{ub}|$  can also be improved by including more experimental measurements of the  $B \rightarrow \pi\ell\nu$  branching fraction. This, however, will require understanding the correlations among the various systematic uncertainties. Given these refinements of the current calculation, an even more precise, model-independent value of  $|V_{ub}|$  can be obtained in the near future.

## ACKNOWLEDGMENTS

We thank R. Hill and E. Lunghi for helpful discussions, and T. Becher for valuable comments on the manuscript. Computations for this work were carried out in part on facilities of the USQCD Collaboration, which are funded by the Office of Science of the U.S. Department of Energy, and on facilities of the NSF Teragrid under Allocation No. TG-MCA93S002. This work was supported in part by the United States Department of Energy under Grant

Nos. DE-FC02-06ER41446 (C.D., L.L.), DE-FG02-91ER40661 (S.G.), DE-FG02-91ER40677 (A.X.K., R.T.E., E.G.), DE-FG02-91ER40628 (C.B., J.L.), DE-FG02-04ER41298 (D.T.) and by the National Science Foundation under Grant Nos. PHY-0555243, PHY-0757333, PHY-0703296 (C.D., L.L.), PHY-0555235 (J.L.), PHY-0456556 (R.S.). R.T.E. and E.G. were supported in part by URA. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

## APPENDIX A: ESTIMATE OF HEAVY-QUARK DISCRETIZATION ERRORS

In this appendix we collect the short-distance functions  $f_i$  used to estimate the heavy-quark discretization effects. For more background, see Refs. [35–38,83,84].

### 1. $\mathcal{O}(a^2)$ errors

We start with these because explicit expressions for the functions  $f_i(m_0a)$  are available.

#### a. $\mathcal{O}(a^2)$ errors from the Lagrangian

There are two bilinears,  $\bar{h}\vec{D} \cdot \vec{E}h$  and  $\bar{h}i\vec{\Sigma} \cdot [\vec{D} \times \vec{E}]h$ , and many four-quark operators. At tree level the coefficients of all four-quark operators vanish and the coefficients of the two bilinears are the same. The mismatch function is given by

$$f_E(m_0a) = \frac{1}{8m_E^2a^2} - \frac{1}{2(2m_2a)^2}. \quad (\text{A1})$$

Using explicit expressions for  $1/m_2$  [35] and  $1/m_E^2$  [84], one finds

$$f_E(m_0a) = \frac{1}{2} \left[ \frac{c_E(1 + m_0a) - 1}{m_0a(2 + m_0a)(1 + m_0a)} - \frac{1}{4(1 + m_0a)^2} \right]. \quad (\text{A2})$$

We use  $c_E = 1$  in our numerical simulations.

#### b. $\mathcal{O}(a^2)$ errors from the current

There are three terms with nonzero coefficients,  $\bar{q}\Gamma\vec{D}^2h$ ,  $\bar{q}\Gamma i\vec{\Sigma} \cdot \vec{B}h$ , and  $\bar{q}\Gamma\vec{\alpha} \cdot \vec{E}h$ , which can be deduced from Eq. (A17) of Ref. [35]. Their coefficients can be read off from Eqs. (A19) [35]. When  $c_B = r_s$  the first two share the same coefficient:

$$\begin{aligned} f_X(m_0a) &= \frac{1}{8m_X^2a^2} - \frac{\zeta d_1(1 + m_0a)}{m_0a(2 + m_0a)} - \frac{1}{2(2m_2a)^2}, \\ &= \frac{1}{2} \left[ \frac{1}{(2 + m_0a)(1 + m_0a)} + \frac{1}{2(1 + m_0a)} \right. \\ &\quad \left. - \frac{1}{4(1 + m_0a)^2} - \frac{1}{(2 + m_0a)^2} \right], \\ &= \frac{1}{2} \left[ \frac{1}{2(1 + m_0a)} - \left( \frac{m_0a}{2(2 + m_0a)(1 + m_0a)} \right)^2 \right], \end{aligned} \quad (\text{A3})$$

where the last term on the second line comes from using the tree-level  $d_1$ . For the third operator,  $\bar{q}\Gamma\vec{\alpha} \cdot \vec{E}h$ ,

$$\begin{aligned} f_Y(m_0a) &= \frac{1}{2} \left[ \frac{d_1}{m_2a} - \frac{\zeta(1 - c_E)(1 + m_0a)}{m_0a(2 + m_0a)} \right], \\ &= \frac{2 + 4m_0a + (m_0a)^2}{4(1 + m_0a)^2(2 + m_0a)^2}, \end{aligned} \quad (\text{A4})$$

where the last line reflects the choices made for  $c_E$  and  $d_1$ .

### 2. $\mathcal{O}(\alpha_s a)$ errors

Because we improve both the action and current, the mismatch functions  $f_i(m_0a)$  start at order  $\alpha_s$ , and we do not have explicit expressions for them. (The calculation of these functions would be needed to match at the one-loop level.) So we shall take unimproved tree-level coefficients as a guide to the combinatoric factors and consider asymptotic behavior in the limits  $m_0a \rightarrow 0, \infty$ .

#### a. $\mathcal{O}(\alpha_s a)$ errors from the Lagrangian

There are two bilinears, the kinetic energy  $\bar{h}\vec{D}^2h$  and the chromomagnetic moment  $\bar{h}i\vec{\Sigma} \cdot \vec{B}h$ . There is no mismatch in the coefficient of the kinetic energy, by assumption, since we identify the kinetic mass with the heavy-quark mass. This tuning is imperfect, but the associated error is budgeted in Sec. IV E.

At the tree level the chromomagnetic mismatch is

$$f_B^{[0]}(m_0a) = \frac{c_B - 1}{2(1 + m_0a)}. \quad (\text{A5})$$

This has the right asymptotic behavior in both limits, so our ansatz for the one-loop mismatch function is simply

$$f_B^{[0]}(m_0a) = \frac{\alpha_s}{2(1 + m_0a)}, \quad (\text{A6})$$

and  $\text{error}_B$  is this function multiplied by  $a\Lambda$ .

TABLE III. Relative error from mismatches in the heavy-quark Lagrangian and current for the bottom quark with  $\Lambda = 700$  MeV. To obtain the totals given in the text,  $E$  and  $X$  are counted twice, and 3 is counted twice for  $f_{\parallel}$  and 4 times for  $f_{\perp}$ . Entries are in percent.

$a$ (fm)	$\alpha_V(q^*)$	$m_0 a$	$B$	3	$E$	$X$	$Y$
0.09	0.33	2.018	1.76	1.32	0.28	0.80	0.24
0.12	0.41	2.617	2.48	1.94	0.39	1.26	0.33

### b. $\mathcal{O}(\alpha_s a)$ errors from the current

There is only one correction at tree level, but more generally there are two for the temporal current and four for the spatial current. (See Eqs. (2.27)–(2.32) of Ref. [37].)

The tree-level mismatch function ends up being the same as  $d_1$ :

$$f_3^{[0]}(m_0 a) = \frac{m_0 a}{2(2 + m_0 a)(1 + m_0 a)}. \quad (\text{A7})$$

It is, however, an accident that it vanishes as  $m_0 a \rightarrow 0$ . Therefore, we instead take

TABLE IV. Normalized statistical (upper) and systematic (lower) bootstrap correlation matrices for the  $B \rightarrow \pi\ell\nu$  form factor,  $f_+(q^2)$ . These should be combined with the values of  $f_+(q^2)$  presented in Table II to reconstruct the full correlation matrices.

$q^2$ (GeV $^2$ )	26.5	25.7	25.0	24.3	23.5	22.8	22.1	21.3	20.6	19.8	19.1	18.4
26.5	1.00	0.99	0.97	0.88	0.66	0.29	-0.01	-0.16	-0.19	-0.14	-0.04	0.05
25.7	0.99	1.00	0.99	0.92	0.73	0.38	0.07	-0.09	-0.13	-0.09	0.00	0.08
25.0	0.97	0.99	1.00	0.97	0.82	0.51	0.21	0.04	-0.02	0.01	0.07	0.13
24.3	0.88	0.92	0.97	1.00	0.93	0.69	0.42	0.25	0.18	0.17	0.20	0.21
23.5	0.66	0.73	0.82	0.93	1.00	0.91	0.72	0.56	0.48	0.43	0.39	0.32
22.8	0.29	0.38	0.51	0.69	0.91	1.00	0.94	0.85	0.77	0.69	0.58	0.42
22.1	-0.01	0.07	0.21	0.42	0.72	0.94	1.00	0.98	0.92	0.84	0.69	0.48
21.3	-0.16	-0.09	0.04	0.25	0.56	0.85	0.98	1.00	0.98	0.92	0.78	0.55
20.6	-0.19	-0.13	-0.02	0.18	0.48	0.77	0.92	0.98	1.00	0.97	0.86	0.66
19.8	-0.14	-0.09	0.01	0.17	0.43	0.69	0.84	0.92	0.97	1.00	0.95	0.80
19.1	-0.04	0.00	0.07	0.20	0.39	0.58	0.69	0.78	0.86	0.95	1.00	0.94
18.4	0.05	0.08	0.13	0.21	0.32	0.42	0.48	0.55	0.66	0.80	0.94	1.00

$q^2$ (GeV $^2$ )	26.5	25.7	25.0	24.3	23.5	22.8	22.1	21.3	20.6	19.8	19.1	18.4
26.5	1.00	0.98	0.95	0.89	0.85	0.88	0.9	0.91	0.91	0.92	0.91	0.90
25.7	0.98	1.00	0.98	0.92	0.87	0.87	0.88	0.89	0.90	0.91	0.9	0.88
25.0	0.95	0.98	1.00	0.97	0.94	0.94	0.94	0.94	0.95	0.95	0.95	0.94
24.3	0.89	0.92	0.97	1.00	0.99	0.99	0.98	0.98	0.98	0.98	0.98	0.97
23.5	0.85	0.87	0.94	0.99	1.00	0.99	0.99	0.98	0.98	0.98	0.98	0.98
22.8	0.88	0.87	0.94	0.99	0.99	1.00	1.0	1.00	0.99	0.99	0.99	0.99
22.1	0.90	0.88	0.94	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00
21.3	0.91	0.89	0.94	0.98	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00
20.6	0.91	0.90	0.95	0.98	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00
19.8	0.92	0.91	0.95	0.98	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00
19.1	0.91	0.9	0.95	0.98	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00
18.4	0.90	0.88	0.94	0.97	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00

$$f_3(m_0 a) = \frac{\alpha_s}{2(2 + m_0 a)}, \quad (\text{A8})$$

which has the right asymptotic behavior.

### 3. Numerical estimates

The relative errors due to mismatches in the heavy-quark Lagrangian and current on the MILC coarse and fine ensembles are tabulated in Table III. At the fine lattice spacing we take the typical  $\alpha_V(q^*)$  to be 1/3, and we use one-loop running to obtain  $\alpha_V(q^*)$  at the coarse lattice spacing. The contribution  $\text{error}_Y$  from the  $\vec{\alpha} \cdot \vec{E}$  error in the current is so small both because  $c_E = 1$  in our simulation and because  $d_1$  is small. Adding the individual errors given in Table III in quadrature, and taking into account multiple contributions of the same size, we find the total error to be 2.84% (4.16%) for  $f_{\parallel}$  and 3.40% (4.98%) for  $f_{\perp}$  on the fine (coarse) lattices.

### APPENDIX B: STATISTICAL AND SYSTEMATIC ERROR MATRICES

In this appendix we present the normalized statistical and systematic bootstrap correlation matrices for the

TABLE V. Total bootstrap covariance matrix for the  $B \rightarrow \pi\ell\nu$  form factor,  $f_+(q^2)$ , derived by adding the statistical and systematic errors in quadrature. The elements of the matrix are given by  $M_{ij} = \sigma_{f_+(q_i^2)} \times \sigma_{f_+(q_j^2)}$ , where  $\sigma_{f_+(q_i^2)}$  is the total uncertainty in  $f_+(q^2)$  in the  $i$ 'th  $q^2$  bin.

$q^2$ (GeV $^2$ )	26.5	25.7	25.0	24.3	23.5	22.8	22.1	21.3	20.6	19.8	19.1	18.4
26.5	5.13	2.74	1.49	0.79	0.39	0.18	0.06	0.01	-0.0	0.01	0.03	0.05
25.7	2.74	1.48	0.82	0.45	0.24	0.12	0.05	0.02	0.01	0.02	0.03	0.04
25.0	1.49	0.82	0.47	0.27	0.15	0.09	0.05	0.03	0.02	0.02	0.02	0.03
24.3	0.79	0.45	0.27	0.17	0.11	0.07	0.05	0.03	0.03	0.02	0.02	0.02
23.5	0.39	0.24	0.15	0.11	0.08	0.06	0.04	0.04	0.03	0.03	0.02	0.02
22.8	0.18	0.12	0.09	0.07	0.06	0.05	0.04	0.04	0.03	0.03	0.02	0.02
22.1	0.06	0.05	0.05	0.05	0.04	0.04	0.04	0.03	0.03	0.03	0.02	0.02
21.3	0.01	0.02	0.03	0.03	0.04	0.04	0.03	0.03	0.03	0.02	0.02	0.02
20.6	0.00	0.01	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02
19.8	0.01	0.02	0.02	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02
19.1	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
18.4	0.05	0.04	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02

$B \rightarrow \pi\ell\nu$  form factor,  $f_+(q^2)$ , which were used in our model-independent determination of  $|V_{ub}|$ . In order to

facilitate the use of our result, we also show the resulting total covariance matrix.

- 
- [1] S.-W. Lin *et al.* (Belle Collaboration), Nature (London) **452**, 332 (2008).  
[2] E. Lunghi and A. Soni, Phys. Lett. B **666**, 162 (2008).  
[3] T. Aaltonen *et al.* (CDF Collaboration), Phys. Rev. Lett. **100**, 161802 (2008).  
[4] V. M. Abazov *et al.* (D0 Collaboration), Phys. Rev. Lett. **101**, 241801 (2008).  
[5] M. Bona *et al.* (UTfit Collaboration), arXiv:0803.0659.  
[6] B. A. Dobrescu and A. S. Kronfeld, Phys. Rev. Lett. **100**, 241802 (2008).  
[7] E. Follana, C. T. H. Davies, G. P. Lepage, and J. Shigemitsu (HPQCD Collaboration), Phys. Rev. Lett. **100**, 062002 (2008).  
[8] L. Widhalm *et al.* (Belle), Phys. Rev. Lett. **100**, 241801 (2008).  
[9] B. Aubert *et al.* (BABAR Collaboration), Phys. Rev. Lett. **98**, 141801 (2007).  
[10] T. K. Pedlar *et al.* (CLEO Collaboration), Phys. Rev. D **76**, 072002 (2007).  
[11] K. M. Ecklund *et al.* (CLEO Collaboration), Phys. Rev. Lett. **100**, 161801 (2008).  
[12] S. Stone, arXiv:0806.3921.  
[13] C. Amsler *et al.* (Particle Data Group), Phys. Lett. B **667**, 1 (2008).  
[14] B. I. Eisenstein *et al.* (CLEO Collaboration), Phys. Rev. D **78**, 052003 (2008).  
[15] C. T. H. Davies *et al.* (HPQCD Collaboration, MILC Collaboration, and Fermilab Lattice Collaboration), Phys. Rev. Lett. **92**, 022001 (2004).  
[16] C. Aubin *et al.* (MILC Collaboration), Phys. Rev. D **70**, 114501 (2004).  
[17] I. F. Allison *et al.* (HPQCD Collaboration, MILC Collaboration, and Fermilab Lattice Collaboration), Phys. Rev. Lett. **94**, 172001 (2005).  
[18] C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **64**, 054506 (2001).  
[19] Y. Shamir, Phys. Rev. D **71**, 034509 (2005).  
[20] Y. Shamir, Phys. Rev. D **75**, 054503 (2007).  
[21] C. Bernard, Phys. Rev. D **73**, 114503 (2006).  
[22] C. Bernard, M. Golterman, and Y. Shamir, Phys. Rev. D **77**, 074505 (2008).  
[23] S. Prelovsek, Phys. Rev. D **73**, 014506 (2006).  
[24] C. Bernard, C. E. Detar, Z. Fu, and S. Prelovsek, Phys. Rev. D **76**, 094504 (2007).  
[25] C. Aubin, J. Laiho, and R. S. Van de Water, Phys. Rev. D **77**, 114501 (2008).  
[26] W.-J. Lee and S. R. Sharpe, Phys. Rev. D **60**, 114503 (1999).  
[27] C. Aubin and C. Bernard, Phys. Rev. D **68**, 034014 (2003).  
[28] S. R. Sharpe and R. S. Van de Water, Phys. Rev. D **71**, 114505 (2005).  
[29] S. Dürr, Proc. Sci., LAT2005 (2006) 021.  
[30] S. R. Sharpe, Proc. Sci., LAT2006 (2006) 022.  
[31] A. S. Kronfeld, Proc. Sci., LAT2007 (2007) 016.  
[32] F. Di Lodovico (HFAG Collaboration), update presented at ICHEP (2008), <http://www.slac.stanford.edu/xorg/hfag/seminar/ichep08/home.shtml>.  
[33] E. Dalgic *et al.* (HPQCD Collaboration), Phys. Rev. D **73**, 074502 (2006); **75**, 119906(E) (2007).  
[34] G. P. Lepage, L. Magnea, C. Nakhleh, U. Magnea, and K. Hornbostel, Phys. Rev. D **46**, 4052 (1992).  
[35] A. X. El-Khadra, A. S. Kronfeld, and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).  
[36] A. S. Kronfeld, Phys. Rev. D **62**, 014505 (2000).  
[37] J. Harada *et al.*, Phys. Rev. D **65**, 094513 (2002).  
[38] J. Harada, S. Hashimoto, A. S. Kronfeld, and T. Onogi, Phys. Rev. D **65**, 094514 (2002).

- [39] C. Aubin *et al.* (Fermilab Lattice Collaboration and MILC Collaboration), Phys. Rev. Lett. **95**, 122002 (2005).
- [40] C. Bernard *et al.*, Phys. Rev. D **78**, 078502(R) (2008); **79**, 014506 (2009).
- [41] C. Aubin and C. Bernard, Phys. Rev. D **76**, 014002 (2007).
- [42] B. Aubert *et al.* (*BABAR* Collaboration), Phys. Rev. Lett. **98**, 091801 (2007).
- [43] M. C. Arnesen, B. Grinstein, I. Z. Rothstein, and I. W. Stewart, Phys. Rev. Lett. **95**, 071802 (2005).
- [44] J. M. Flynn and J. Nieves, Phys. Rev. D **76**, 031302 (2007).
- [45] C. Bourrely, I. Caprini, and L. Lellouch, Phys. Rev. D **79**, 013008 (2009).
- [46] B. Sheikholeslami and R. Wohlert, Nucl. Phys. **B259**, 572 (1985).
- [47] G. P. Lepage and P. B. Mackenzie, Phys. Rev. D **48**, 2250 (1993).
- [48] R. Sommer, Nucl. Phys. **B411**, 839 (1994).
- [49] C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **62**, 034503 (2000).
- [50] C. Aubin *et al.* (MILC Collaboration), Phys. Rev. D **70**, 094505 (2004).
- [51] C. Bernard *et al.* (MILC Collaboration), Proc. Sci., LAT2007 (2007) 090.
- [52] W. M. Yao *et al.* (Particle Data Group), J. Phys. G **33**, 1 (2006).
- [53] A. X. El-Khadra, A. S. Kronfeld, P. B. Mackenzie, S. M. Ryan, and J. N. Simone, Phys. Rev. D **64**, 014502 (2001).
- [54] M. Wingate, J. Shigemitsu, C. T. H. Davies, G. P. Lepage, and H. D. Trottier (HPQCD Collaboration), Phys. Rev. D **67**, 054505 (2003).
- [55] F. Gliozzi, Nucl. Phys. **B204**, 419 (1982).
- [56] H. Kluberg-Stern, A. Morel, O. Napoly, and B. Petersson, Nucl. Phys. **B220**, 447 (1983).
- [57] A. X. El-Khadra, E. Gamiz, A. S. Kronfeld, and M. A. Nobes (unpublished).
- [58] Q. Mason *et al.* (HPQCD Collaboration), Phys. Rev. Lett. **95**, 052002 (2005).
- [59] S. J. Brodsky, G. P. Lepage, and P. B. Mackenzie, Phys. Rev. D **28**, 228 (1983).
- [60] K. Hornbostel, G. P. Lepage, and C. Morningstar, Phys. Rev. D **67**, 034023 (2003).
- [61] M. Lüscher and P. Weisz, Nucl. Phys. **B266**, 309 (1986).
- [62] A. X. El-Khadra, E. Gamiz, A. S. Kronfeld, and M. A. Nobes, Proc. Sci., LAT2007 (2007) 242.
- [63] M. F. L. Golterman and J. Smit, Nucl. Phys. **B245**, 61 (1984).
- [64] G. W. Kilcup and S. R. Sharpe, Nucl. Phys. **B283**, 493 (1987).
- [65] G. Burdman, Z. Ligeti, M. Neubert, and Y. Nir, Phys. Rev. D **49**, 2331 (1994).
- [66] C. Bernard *et al.* (MILC Collaboration), update of Ref. [51] (private communication).
- [67] M. Okamoto *et al.* (Fermilab Lattice Collaboration, MILC Collaboration, and HPQCD Collaboration), Nucl. Phys. B, Proc. Suppl. **140**, 461 (2005).
- [68] S. Aoki *et al.* (JLQCD Collaboration), Phys. Rev. D **64**, 114505 (2001).
- [69] G. P. Lepage *et al.*, Nucl. Phys. B, Proc. Suppl. **106**, 12 (2002).
- [70] C. Bernard *et al.* (Fermilab Lattice Collaboration and MILC Collaboration), Proc. Sci., LAT2007 (2007) 370.
- [71] C. Bernard *et al.* (Fermilab Lattice Collaboration and MILC Collaboration), update of Ref. [70], to be published in Proc. Sci. LAT **2008**, 278 (2008).
- [72] A. Anastassov *et al.* (CLEO Collaboration), Phys. Rev. D **65**, 032003 (2002).
- [73] I. W. Stewart, Nucl. Phys. **B529**, 62 (1998).
- [74] S. Fajfer and J. F. Kamenik, Phys. Rev. D **74**, 074023 (2006).
- [75] A. Abada *et al.*, Nucl. Phys. B, Proc. Suppl. **119**, 641 (2003).
- [76] A. Gray *et al.* (HPQCD Collaboration), Phys. Rev. D **72**, 094507 (2005).
- [77] M. G. Alford, W. Dimm, G. P. Lepage, G. Hockney, and P. B. Mackenzie, Phys. Lett. B **361**, 87 (1995).
- [78] C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **58**, 014503 (1998).
- [79] M. Lüscher and P. Weisz, Commun. Math. Phys. **97**, 59 (1985).
- [80] M. Lüscher and P. Weisz, Phys. Lett. B **158**, 250 (1985).
- [81] G. P. Lepage, Phys. Rev. D **59**, 074502 (1999).
- [82] K. Orginos, D. Toussaint, and R. L. Sugar (MILC Collaboration), Phys. Rev. D **60**, 054503 (1999).
- [83] A. S. Kronfeld, Nucl. Phys. B, Proc. Suppl. **129**, 46 (2004).
- [84] M. B. Oktay and A. S. Kronfeld, Phys. Rev. D **78**, 014504 (2008).
- [85] D. Arndt and C. J. D. Lin, Phys. Rev. D **70**, 014503 (2004).
- [86] C. Bourrely, B. Machet, and E. de Rafael, Nucl. Phys. **B189**, 157 (1981).
- [87] C. G. Boyd, B. Grinstein, and R. F. Lebed, Phys. Rev. Lett. **74**, 4603 (1995).
- [88] L. Lellouch, Nucl. Phys. **B479**, 353 (1996).
- [89] C. G. Boyd and M. J. Savage, Phys. Rev. D **56**, 303 (1997).
- [90] D. Bećirević and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000).
- [91] P. Ball and R. Zwicky, Phys. Rev. D **71**, 014015 (2005).
- [92] S. C. Generalis, J. Phys. G **16**, 785 (1990).
- [93] T. Becher and R. J. Hill, Phys. Lett. B **633**, 61 (2006).
- [94] P. Gambino, talk presented at CKM, 2008.
- [95] J. H. Kühn, M. Steinhauser, and C. Sturm, Nucl. Phys. **B778**, 192 (2007).
- [96] M. Neubert, arXiv:0801.0675.
- [97] J. Charles *et al.* (CKMfitter Collaboration), preliminary results for Summer 2008, [http://ckmfitter.in2p3.fr/plots\\_Summer2008/ckmEval\\_results.html](http://ckmfitter.in2p3.fr/plots_Summer2008/ckmEval_results.html).
- [98] L. Silvestrini (UTFit Collaboration), preliminary result presented at Lattice 2008, <http://www.utfit.org/>.

**$\bar{B} \rightarrow D^* \ell \bar{\nu}$  form factor at zero recoil from three-flavor lattice QCD:  
A model independent determination of  $|V_{cb}|$**

C. Bernard,<sup>1</sup> C. DeTar,<sup>2</sup> M. Di Pierro,<sup>3</sup> A. X. El-Khadra,<sup>4</sup> R. T. Evans,<sup>4</sup> E. D. Freeland,<sup>5</sup> E. Gamiz,<sup>4</sup> Steven Gottlieb,<sup>6</sup> U. M. Heller,<sup>7</sup> J. E. Hetrick,<sup>8</sup> A. S. Kronfeld,<sup>9</sup> J. Laiho,<sup>1,9</sup> L. Levkova,<sup>2</sup> P. B. Mackenzie,<sup>9</sup> M. Okamoto,<sup>9</sup> J. Simone,<sup>9</sup> R. Sugar,<sup>10</sup> D. Toussaint,<sup>11</sup> and R. S. Van de Water<sup>9</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>*Department of Physics, Washington University, St. Louis, Missouri, USA*

<sup>2</sup>*Physics Department, University of Utah, Salt Lake City, Utah, USA*

<sup>3</sup>*School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois, USA*

<sup>4</sup>*Physics Department, University of Illinois, Urbana, Illinois, USA*

<sup>5</sup>*Liberal Arts Department, The School of the Art Institute of Chicago, Chicago, Illinois, USA*

<sup>6</sup>*Department of Physics, Indiana University, Bloomington, Indiana, USA*

<sup>7</sup>*American Physical Society, Ridge, New York, USA*

<sup>8</sup>*Physics Department, University of the Pacific, Stockton, California, USA*

<sup>9</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois, USA*

<sup>10</sup>*Department of Physics, University of California, Santa Barbara, California, USA*

<sup>11</sup>*Department of Physics, University of Arizona, Tucson, Arizona, USA*

(Received 28 August 2008; published 22 January 2009)

We present the first lattice QCD calculation of the form factor for  $\bar{B} \rightarrow D^* \ell \bar{\nu}$  with three flavors of sea quarks. We use an improved staggered action for the light valence and sea quarks (the MILC configurations), and the Fermilab action for the heavy quarks. The form factor is computed at zero recoil using a new double ratio method that yields the form factor more directly than the previous Fermilab method. Other improvements over the previous calculation include the use of much lighter light-quark masses, and the use of lattice (staggered) chiral perturbation theory in order to control the light-quark discretization errors and chiral extrapolation. We obtain for the form factor,  $\mathcal{F}_{B \rightarrow D^*}(1) = 0.921(13)(20)$ , where the first error is statistical and the second is the sum of all systematic errors in quadrature. Applying a 0.7% electromagnetic correction and taking the latest PDG average for  $\mathcal{F}_{B \rightarrow D^*}(1)|V_{cb}|$  leads to  $|V_{cb}| = (38.7 \pm 0.9_{\text{exp}} \pm 1.0_{\text{theo}}) \times 10^{-3}$ .

DOI: 10.1103/PhysRevD.79.014506

PACS numbers: 12.38.Gc, 12.15.Hh, 13.25.Hw

## I. INTRODUCTION

The Cabibbo-Kobayashi-Maskawa matrix element  $V_{cb}$  plays an important role in the study of flavor physics [1]. Since  $|V_{cb}|$  is one of the fundamental parameters of the standard model, its value must be known precisely in order to search for new physics by looking for inconsistencies between standard model predictions and experimental measurements. For example, the standard model contribution to the kaon mixing parameter  $\epsilon_K$  depends sensitively on  $|V_{cb}|$  (as the fourth power), and the present errors on this quantity contribute errors to the theoretical prediction of  $\epsilon_K$  that are around the same size as the errors due to  $B_K$ , the kaon bag parameter, which has been the focus of much recent work [2–5]. It is possible to obtain  $|V_{cb}|$  from both inclusive and exclusive semileptonic  $B$  decays, and both determinations are limited by theoretical uncertainties. The inclusive method [6–10] makes use of the heavy-quark expansion and perturbation theory. The method also requires nonperturbative input from experiment, which is obtained from the measured moments of the inclusive

form factor  $\bar{B} \rightarrow X_c \ell \bar{\nu}_\ell$  as a function of the minimum electron momentum. The dominant uncertainties in this method are the truncation of the heavy-quark expansion and perturbation theory [11,12]. In order to be competitive with the inclusive determination of  $|V_{cb}|$  and thus serve as a cross-check, the exclusive method requires a reduction in the uncertainty of the  $B \rightarrow D^*$  semileptonic form factor  $\mathcal{F}_{B \rightarrow D^*}$ , which has been calculated previously using lattice QCD in the quenched approximation [13].

Given the phenomenological importance of  $|V_{cb}|$ , we have revisited the calculation of  $\mathcal{F}_{B \rightarrow D^*}$  at zero recoil using the 2 + 1 flavor MILC ensembles with improved light staggered quarks [14,15]. The systematic error due to quenching is thus eliminated. The systematic error associated with the chiral extrapolation to physical light-quark masses is also reduced significantly. Since staggered quarks are computationally less expensive than many other formulations, we are able to simulate at quite small quark masses; our lightest corresponds to a pion mass of roughly 240 MeV. Given the previous experience of the MILC Collaboration with chiral fits to light meson masses and

decay constants [16], we are in a regime where we expect rooted staggered chiral perturbation theory (rS $\chi$ PT) [17–21] to apply. We therefore use the rS $\chi$ PT result for the  $B \rightarrow D^*$  form factor [22] to perform the chiral extrapolation and to remove discretization effects particular to staggered quarks. In addition, we introduce a set of ratios that allows us to disentangle light- and heavy-quark discretization effects, and we suggest a strategy for future improvement. Finally, we extract the  $B \rightarrow D^*$  form factor using a different method from that originally proposed in Ref. [13]. This new method requires many fewer three-point correlation functions, and has allowed for a savings of roughly a factor of 10 in computing resources, while at the same time simplifying the analysis.

The differential rate for the semileptonic decay  $\bar{B} \rightarrow D^* \ell \bar{\nu}_\ell$  is

$$\frac{d\Gamma}{dw} = \frac{G_F^2}{4\pi^3} m_{D^*}^3 (m_B - m_{D^*})^2 \sqrt{w^2 - 1} \times \mathcal{G}(w) |V_{cb}|^2 |\mathcal{F}_{B \rightarrow D^*}(w)|^2, \quad (1)$$

where  $w = \mathbf{v}' \cdot \mathbf{v}$  is the velocity transfer from the initial state to the final state, and  $\mathcal{G}(w) |\mathcal{F}_{B \rightarrow D^*}(w)|^2$  contains a combination of four form factors that must be calculated nonperturbatively. At zero recoil  $\mathcal{G}(1) = 1$ , and  $\mathcal{F}_{B \rightarrow D^*}(1)$  reduces to a single form factor  $h_{A_1}(1)$ . Given  $h_{A_1}(1)$ , the measured decay rate determines  $|V_{cb}|$ .

The quantity  $h_{A_1}$  is a form factor of the axial-vector current

$$\langle D^*(v, \epsilon') | \mathcal{A}^\mu | \bar{B}(v) \rangle = i\sqrt{2m_B 2m_{D^*}} \bar{\epsilon}'^\mu h_{A_1}(1), \quad (2)$$

where  $\mathcal{A}^\mu$  is the continuum axial-vector current and  $\epsilon'$  is the polarization vector of the  $D^*$ . Heavy-quark symmetry plays a useful role in constraining  $h_{A_1}(1)$ , leading to the heavy-quark expansion [23,24]

$$h_{A_1}(1) = \eta_A \left[ 1 - \frac{\ell_V}{(2m_c)^2} + \frac{2\ell_A}{2m_c 2m_b} - \frac{\ell_P}{(2m_b)^2} \right], \quad (3)$$

up to order  $1/m_Q^2$ , and where  $\eta_A$  is a factor that matches heavy-quark effective theory (HQET) to QCD [25,26]. The  $\ell$ 's are long-distance matrix elements of the HQET. Heavy-quark symmetry forbids terms of order  $1/m_Q$  at zero recoil [27], and various methods have been used to compute the size of the  $1/m_Q^2$  coefficients, including quenched lattice QCD [13].

The earlier work by Hashimoto *et al.* [13] used three double ratios in order to obtain separately each of the three  $1/m_Q^2$  coefficients in Eq. (3). These three double ratios also determine three out of the four coefficients appearing at  $1/m_Q^3$  in the heavy-quark expansion. It was shown in Ref. [28] that, for the Fermilab method matched to tree level in  $\alpha_s$  and to next-to-leading order in HQET, the leading discretization errors for the double ratios for this quantity are of order  $\alpha_s(\bar{\Lambda}/2m_Q)^2 f_B(am_Q)$  and

$(\bar{\Lambda}/2m_Q)^3 f_i(am_Q)$ , where  $\bar{\Lambda}$  is a QCD scale stemming from the light degrees of freedom, such as that appearing in the HQET expansion for the heavy-light meson mass  $m_M = m_Q + \bar{\Lambda} + \dots$ . The functions  $f_i(am_Q)$  are coefficients depending on  $am_Q$  and  $\alpha_s$ , but not on  $\bar{\Lambda}$ . When  $am_Q \sim 1$ , the  $f_i(am_Q)$  are of order one; when  $am_Q \ll 1$ , they go like a power of  $am_Q$ , such that the continuum limit is obtained. The powers of 2 are combinatoric factors.

As discussed in Ref. [13], all uncertainties in the double ratios  $\mathcal{R}$  used in that work scale as  $\mathcal{R} - 1$  rather than as  $\mathcal{R}$ . Statistical errors in the numerator and denominator are highly correlated and largely cancel in these double ratios. Also, most of the normalization uncertainty in the lattice currents cancels, leaving a normalization factor close to 1, which can be computed reliably in perturbation theory. Finally, the quenching error, relevant to Ref. [13] but not to the present unquenched calculation, scales as  $\mathcal{R} - 1$  rather than as  $\mathcal{R}$ . This scaling of the error occurs because the double ratios constructed in Ref. [13] become the identity in the limit of equal bottom and charm quark masses.

In the calculation reported here, the form factor  $h_{A_1}(1)$  is computed more directly using only one double ratio

$$\mathcal{R}_{A_1} = \frac{\langle D^* | \bar{c} \gamma_j \gamma_5 b | \bar{B} \rangle \langle \bar{B} | \bar{b} \gamma_j \gamma_5 c | D^* \rangle}{\langle D^* | \bar{c} \gamma_4 c | D^* \rangle \langle \bar{B} | \bar{b} \gamma_4 b | \bar{B} \rangle} = |h_{A_1}(1)|^2, \quad (4)$$

which is exact to all orders in the heavy-quark expansion in the continuum.<sup>1</sup> The lattice approximation to this ratio still has discretization errors that are suppressed by inverse powers of heavy-quark masses [ $\alpha_s(\bar{\Lambda}/2m_Q)^2$  and  $(\bar{\Lambda}/2m_Q)^3$ ], but which again vanish in the continuum limit. The errors in the ratio introduced in Eq. (4) do not scale rigorously as  $\mathcal{R}_{A_1} - 1$  because  $\mathcal{R}_{A_1}$  is not one in the limit of equal bottom and charm quark masses. Nevertheless, this double ratio still retains the desirable features of the previous double ratios, i.e., large statistical error cancellations and the cancellation of most of the lattice current renormalization. Because the quenching error has been eliminated, the rigorous scaling of all the errors as  $\mathcal{R} - 1$ , including the quenching error, is no longer crucial. The more direct method introduced here has the significant advantage that extracting coefficients from fits to HQET expressions as a function of heavy-quark masses is not necessary, and no error is introduced from truncating the heavy-quark expansion to a fixed order in  $1/m_Q^n$ . In short, for an unquenched QCD calculation, the method using Eq. (4) gives a smaller total error than the method used in Ref. [13] for a fixed amount of computer time.

The currents of lattice gauge theory must be matched to the normalization of the continuum to obtain  $\mathcal{R}_{A_1}$ . The matching factors mostly cancel in the double ratio [29,30],

<sup>1</sup>Note that the notation  $\mathcal{R}_{A_1}$  stands for a different double ratio in Ref. [13].

leaving  $h_{A_1}(1) = \sqrt{\mathcal{R}_{A_1}} = \rho\sqrt{R_{A_1}}$ , where  $R_{A_1}$  is the lattice double ratio and  $\rho$ , the ratio of matching factors, is very close to 1. (For the remainder of this paper we shall use the convention that a script letter corresponds to a continuum quantity, while a nonscript letter corresponds to a lattice quantity.) This  $\rho$  factor has been calculated to one-loop order in perturbative QCD, and is found to contribute less than a 0.5% correction. We have exploited the  $\rho$  factors to implement a blind analysis. Two of us involved in the perturbative calculation applied a common multiplicative offset to the  $\rho$  factors needed to obtain  $h_{A_1}(1)$  at different lattice spacings. This offset was not disclosed to the rest of us until the procedure for determining the systematic error budget for the rest of the analysis had been finalized.

The unquenched MILC configurations generated with 2 + 1 flavors of improved staggered fermions make use of the fourth-root procedure for eliminating the unwanted four-fold degeneracy of staggered quarks. At nonzero lattice spacing, this procedure has small violations of unitarity [31–35] and locality [36]. Nevertheless, a careful treatment of the continuum limit, in which all assumptions are made explicit, argues that lattice QCD with rooted staggered quarks reproduces the desired local theory of QCD as  $a \rightarrow 0$  [37,38]. When coupled with other analytical and numerical evidence (see Refs. [39–41] for reviews), this gives us confidence that the rooting procedure is indeed correct in the continuum limit.

The outline of the rest of this paper is as follows: Section II describes the details of the lattice simulation. Section III discusses the fits to the double ratios accounting for oscillating opposite-parity states. Section IV summarizes the lattice perturbation theory calculation of the  $\rho$  factor. Section V introduces the rooted staggered chiral perturbation theory formalism and expressions used in the chiral extrapolations. Section VI then discusses our treatment of the chiral extrapolation and introduces our approach for disentangling heavy- and light-quark discre-

te effects. Section VII provides a detailed discussion of our systematic errors, and we conclude in Sec. VIII.

## II. LATTICE CALCULATION

The lattice calculation was done on the MILC ensembles at three lattice spacings with  $a \approx 0.15, 0.125$ , and  $0.09$  fm; these ensembles have an  $O(a^2)$  Symanzik improved gauge action and 2 + 1 flavors of “AsqTad” improved staggered sea quarks [42–47]. The parameters for the MILC lattices used in this calculation are shown in Table I. We have several light masses at both full QCD and partially quenched points ( $m_{\text{val}} \neq m_{\text{sea}}$ ), and our light-quark masses range between  $m_s/10$  and  $m_s/2$ . Table II shows the valence masses computed on each ensemble. In this work we follow the notation [16], where  $m_s$  is the physical strange quark mass,  $\hat{m}$  is the average  $u$ - $d$  quark mass, and  $\hat{m}', m'_s$  indicate the nominal values used in simulations. In practice, the MILC ensembles choose  $m'_s$  within 10–30% of  $m_s$  and a range of  $\hat{m}'$  to enable a chiral extrapolation.

The heavy quarks are computed using the Sheikholeslami-Wohlert (SW) “clover” action [48] with the Fermilab interpretation via HQET [49]. The SW action includes a dimension-five interaction with a coupling  $c_{\text{sw}}$  that has been adjusted to the value  $u_0^{-3}$  suggested by tadpole-improved, tree-level perturbation theory [50]. The value of  $u_0$  is calculated either from the plaquette ( $a \approx 0.15$  fm and  $a \approx 0.09$  fm), or from the Landau link ( $a \approx 0.12$  fm). The adjustment of  $c_{\text{sw}}$  is needed to normalize the heavy quark’s chromomagnetic moment correctly [49].

The tadpole-improved bare quark mass for SW quarks is given by

$$am_0 = \frac{1}{u_0} \left( \frac{1}{2\kappa} - \frac{1}{2\kappa_{\text{crit}}} \right), \quad (5)$$

where tuning the parameter  $\kappa$  to the critical quark hopping parameter  $\kappa_{\text{crit}}$  would lead to a massless pion. The spin-averaged  $B_s$  and  $D_s$  kinetic masses are computed on a

TABLE I. Parameters of the simulations. The columns from left to right are the approximate lattice spacing in fm, the sea quark masses  $a\hat{m}'/am'_s$ , the linear spatial dimension of the lattice ensemble in fm, the dimensionless factor  $m_\pi L$  ( $m_\pi$  corresponds to the taste-pseudoscalar pion composed of light sea quarks), the gauge coupling, the dimensions of the lattice in lattice units, the number of configurations used for this analysis, the bare hopping parameter used for the bottom quark, the bare hopping parameter used for the charm quark, and the clover term  $c_{\text{sw}}$  used for both bottom and charm quarks.

$a$ (fm)	$a\hat{m}'/am'_s$	$L$ (fm)	$m_\pi L$	$10/g^2$	Volume	# Configs	$\kappa_b$	$\kappa_c$	$c_{\text{sw}}$
0.15	0.0194/0.0484	2.4	5.5	6.586	$16^3 \times 48$	628	0.076	0.122	1.5673
0.15	0.0097/0.0484	2.4	3.9	6.572	$16^3 \times 48$	628	0.076	0.122	1.5673
0.12	0.02/0.05	2.4	6.2	6.79	$20^3 \times 64$	460	0.086	0.122	1.72
0.12	0.01/0.05	2.4	4.5	6.76	$20^3 \times 64$	592	0.086	0.122	1.72
0.12	0.007/0.05	2.4	3.8	6.76	$20^3 \times 64$	836	0.086	0.122	1.72
0.12	0.005/0.05	2.9	3.8	6.76	$24^3 \times 64$	528	0.086	0.122	1.72
0.09	0.0124/0.031	2.4	5.8	7.11	$28^3 \times 96$	516	0.0923	0.127	1.476
0.09	0.0062/0.031	2.4	4.1	7.09	$28^3 \times 96$	556	0.0923	0.127	1.476
0.09	0.0031/0.031	3.4	4.2	7.08	$40^3 \times 96$	504	0.0923	0.127	1.476

TABLE II. Valence masses used in the simulations. The columns from left to right are the approximate lattice spacing in fm, the sea quark masses  $a\hat{m}'/am'_s$  identifying the gauge ensemble, and the valence masses computed on that ensemble.

$a$ (fm)	$a\hat{m}'/am'_s$	$am_x$
$\approx 0.15$	0.0194/0.0484	0.0194
$\approx 0.15$	0.0097/0.0484	0.0097, 0.0194
$\approx 0.12$	0.02/0.05	0.02
$\approx 0.12$	0.01/0.05	0.01, 0.02
$\approx 0.12$	0.007/0.05	0.007, 0.02
$\approx 0.12$	0.005/0.05	0.005, 0.02
$\approx 0.09$	0.0124/0.031	0.0124
$\approx 0.09$	0.0062/0.031	0.0062, 0.0124
$\approx 0.09$	0.0031/0.031	0.0031, 0.0124

subset of the ensembles in order to tune the bare  $\kappa$  values for bottom and charm (and hence the corresponding bare quark masses) to their physical values. These tuned values were then used in the  $B \rightarrow D^*\ell\nu$  form-factor production run.

The relative lattice scale is determined by calculating  $r_1/a$  on each ensemble, where  $r_1$  is related to the force between static quarks by  $r_1^2 F(r_1) = 1.0$  [51,52]. To avoid introducing implicit dependence on  $\hat{m}'$ ,  $m'_s$  via  $r_1(\hat{m}', m'_s, g^2)$  (where, as above, primes denote simulation masses), we interpolate in  $m'_s$  and extrapolate in  $\hat{m}'$  to obtain  $r_1(\hat{m}, m_s, g^2)/a$  at the physical masses. We then convert from lattice units to  $r_1$  units with  $r_1(\hat{m}, m_s, g^2)/a$ . Below we shall call this procedure the mass-independent determination of  $r_1$ .

In order to fix the absolute lattice scale, one must compute a physical quantity that can be compared directly to experiment; we use the Y 2S-1S splitting [53] and the most recent MILC determination of  $f_\pi$  [54]. The difference between these determinations results in a systematic error that turns out to be much smaller than our other systematics. When the Y scale determination is combined with the continuum extrapolated  $r_1$  value at physical quark masses, a value  $r_1^{\text{phys}} = 0.318(7)$  fm [55] is obtained. The  $f_\pi$  determination is  $r_1^{\text{phys}} = 0.3108(15)(^{+26}_{-79})$  fm [54]. Given  $r_1^{\text{phys}}$ , it is then straightforward to convert quantities measured in  $r_1$  units to physical units.

The dependence on the lattice spacing  $a$  is mild in this analysis. Since  $a$  only enters the calculation through the adjustment of the heavy- and light-quark masses, the dependence of  $h_{A_1}(1)$  on  $a$  is small. Staggered chiral perturbation theory indicates that the  $a$  dependence coming from staggered-quark discretization effects is small [22], and this is consistent with the simulation data.

In this work, we construct lattice currents as in Ref. [49],

$$J_\mu^{hh'} = \sqrt{Z_{V_4}^{hh} Z_{V_4}^{h'h'}} \bar{\Psi}_h \Gamma_\mu \Psi_{h'}, \quad (6)$$

where  $\Gamma_\mu$  is either the vector ( $i\gamma^\mu$ ) or axial-vector ( $i\gamma^\mu \gamma_5$ )

current. The rotated field  $\Psi_h$  is defined by

$$\Psi_h = (1 + ad_1 \boldsymbol{\gamma} \cdot \mathbf{D}_{\text{lat}}) \psi_h, \quad (7)$$

where  $\psi_h$  is the (heavy) lattice quark field in the SW action.  $\mathbf{D}_{\text{lat}}$  is the symmetric, nearest-neighbor, covariant difference operator; the tree-level improvement coefficient is

$$d_1 = \frac{1}{u_0} \left( \frac{1}{2 + m_0 a} - \frac{1}{2(1 + m_0 a)} \right). \quad (8)$$

In Eq. (6) we choose to normalize the current by the factors of  $Z_{V_4}^{hh}$  ( $h = c, b$ ) since even for massive quarks they are easy to compute nonperturbatively. The continuum current is related to the lattice current by

$$\mathcal{J}_\mu^{hh'} = \rho_{J_\Gamma} J_\mu^{hh'} \quad (9)$$

up to discretization effects, where

$$\rho_{J_\Gamma}^2 = \frac{Z_{J_\Gamma}^{bc} Z_{J_\Gamma}^{cb}}{Z_{V_4}^{cc} Z_{V_4}^{bb}}, \quad (10)$$

and the matching factors  $Z_{J_\Gamma}^{hh'}$ 's are defined in Ref. [30].

Note that the factor  $\sqrt{Z_{V_4}^{bb} Z_{V_4}^{cc}}$  multiplying the lattice current in Eq. (6) cancels in the double ratio by design, leaving only the  $\rho$  factor, which is close to 1 and can be computed reliably using perturbation theory. The perturbative calculation of  $\rho_{J_\Gamma}$  is described in more detail in Sec. IV.

Interpolating operators are constructed from four-component heavy quarks and staggered quarks as follows: Let

$$\mathcal{O}_{D^*}(x) = \bar{\chi}(x) \Omega^\dagger(x) i\gamma_j \psi_c(x), \quad (11)$$

$$\mathcal{O}_B^\dagger(x) = \bar{\psi}_b(x) \gamma_5 \Omega(x) \chi(x), \quad (12)$$

where  $\chi$  is the one-component field in the staggered-quark action, and

$$\Omega(x) = \gamma_1^{x_1/a} \gamma_2^{x_2/a} \gamma_3^{x_3/a} \gamma_4^{x_4/a}. \quad (13)$$

The left (right) index of  $\Omega^\dagger$  ( $\Omega$ ) can be left as a free taste index [41] or  $\chi$  can be promoted to a four-component naive-quark field to contract all indices [56]. The resulting correlation functions are the same if the initial and final taste indices are set equal and then summed. The same kinds of operators have been used in previous calculations [57–59].

Lattice matrix elements are obtained from three-point correlation functions. The three-point correlation functions needed for the  $B \rightarrow D^*$  transition at zero-recoil are

$$C^{B \rightarrow D^*}(t_i, t_s, t_f) = \sum_{\mathbf{x}, \mathbf{y}} \langle 0 | \mathcal{O}_{D^*}(\mathbf{x}, t_f) \bar{\Psi}_c \gamma_j \gamma_5 \Psi_b(\mathbf{y}, t_s) \times \mathcal{O}_B^\dagger(\mathbf{0}, t_i) | 0 \rangle, \quad (14)$$

$$C^{B \rightarrow B}(t_i, t_s, t_f) = \sum_{\mathbf{x}, \mathbf{y}} \langle 0 | \mathcal{O}_B(\mathbf{x}, t_f) \bar{\Psi}_b \gamma_4 \Psi_b(\mathbf{y}, t_s) \times \mathcal{O}_B^\dagger(\mathbf{0}, t_i) | 0 \rangle, \quad (15)$$

$$C^{D^* \rightarrow D^*}(t_i, t_s, t_f) = \sum_{\mathbf{x}, \mathbf{y}} \langle 0 | \mathcal{O}_{D^*}(\mathbf{x}, t_f) \bar{\Psi}_c \gamma_4 \Psi_c(\mathbf{y}, t_s) \times \mathcal{O}_{D^*}^\dagger(\mathbf{0}, t_i) | 0 \rangle. \quad (16)$$

In  $C^{B \rightarrow D^*}$  the polarization of the  $D^*$  lies along spatial direction  $j$ . If the source-sink separation is large enough, then we can arrange for both  $t_s - t_i$  and  $t_f - t_s$  to be large so that the lowest-lying state dominates. Then

$$C^{B \rightarrow D^*}(t_i, t_s, t_f) = Z_{D^*}^{1/2} Z_B^{1/2} \frac{\langle D^* | \bar{\Psi}_c \gamma_j \gamma_5 \Psi_b | B \rangle}{\sqrt{2m_{D^*}} \sqrt{2m_B}} \times e^{-m_B(t_s - t_i)} e^{-m_{D^*}(t_f - t_s)} + \dots, \quad (17)$$

where  $m_B$  and  $m_{D^*}$  are the masses of the  $B$  and  $D^*$  mesons and  $Z_H = |\langle 0 | \mathcal{O}_H | H \rangle|^2$ .

In practice, the meson source and sink are held at fixed  $t_i = 0$  and  $t_f = T$ , while the operator time  $t_s = t$  is varied over all times in between. Using the correlators defined in Eqs. (14)–(16) we form the double ratio

$$R_{A_1}(t) = \frac{C^{B \rightarrow D^*}(0, t, T) C^{D^* \rightarrow B}(0, t, T)}{C^{D^* \rightarrow D^*}(0, t, T) C^{B \rightarrow B}(0, t, T)}. \quad (18)$$

All convention-dependent normalization factors, including the factors of  $\sqrt{Z_H / 2m_H}$ , cancel in the double ratio. In the window of time separations where the ground state dominates, a plateau should be visible, and the lattice ratio is simply related to the continuum ratio  $\mathcal{R}_{A_1}$  by a renormalization factor

$$\rho_{A_1} \sqrt{R_{A_1}} = \sqrt{\mathcal{R}_{A_1}} = h_{A_1}(1), \quad (19)$$

with  $\rho_{A_1}$  as in Eq. (10). The right-hand side of Eq. (17) is the first term in a series, with additional terms for each radial excitation, including opposite-parity states that arise with staggered quarks. Eliminating the opposite-parity states requires some care, and this is discussed in detail in the next section. In order to isolate the lowest-lying states we have chosen creation and annihilation operators,  $\mathcal{O}_B^\dagger$  and  $\mathcal{O}_{D^*}$ , which have a large overlap with the desired state. This was done by smearing the heavy quark and antiquark propagator sources with 1S Coulomb-gauge wave functions.

### III. FITTING AND OPPOSITE-PARITY STATES

Extracting correlation functions of operators with staggered quarks presents an extra complication because the contributions of opposite-parity states introduce oscillations in time into the correlator fits [56]. Three-point functions obey the functional form

$$C^{X \rightarrow Y}(0, t, T) = \sum_{k=0} \sum_{\ell=0} (-1)^{kt} \times (-1)^{\ell(T-t)} A_{\ell k} e^{-m_X^{(k)} t} e^{-m_Y^{(\ell)} (T-t)}. \quad (20)$$

For odd  $k$  and  $\ell$  the excited-state contributions change sign as the position of the operator varies by one time slice. Although they are exponentially suppressed, the parity partners of the heavy-light mesons are not that much heavier than the ground states in which we are interested, so the oscillations can be significant at the source-sink separations typical of our calculations. These separations cannot be too large because of the rapid decrease of the signal due to the presence of the heavy quark.

Although one can fit a given three-point correlator to Eq. (20), in the calculation of  $h_{A_1}(1)$  we use double ratios in which numerator and denominator are so similar that most of the fitting systematics cancel, and it is convenient to preserve this simplifying feature. We do this by forming a suitable average over correlator ratios with different (even and odd) source-sink separations. It turns out that the amplitudes of the oscillating states in  $B \rightarrow D^*$  correlation functions are much smaller than they are in many other heavy-light transitions [60,61], and that the oscillating states in  $B \rightarrow D^*$  are barely visible at the present level of statistics. Even so, we introduce an average that reduces them still further, to the point where they are negligible.

Although we shall take the average of the double ratio, let us first examine the average of an individual three-point function. Expanding Eq. (20) so that it includes the ground state and the first oscillating state, we have

$$\begin{aligned} C^{X \rightarrow Y}(0, t, T) &= A_{00}^{X \rightarrow Y} e^{-m_X t - m_Y (T-t)} \\ &\quad + (-1)^{T-t} A_{01}^{X \rightarrow Y} e^{-m_X t - m_Y^{(1)} (T-t)} \\ &\quad + (-1)^t A_{10}^{X \rightarrow Y} e^{-m_X^{(1)} t - m_Y (T-t)} \\ &\quad + (-1)^T A_{11}^{X \rightarrow Y} e^{-m_X^{(1)} t - m_Y^{(1)} (T-t)} + \dots \\ &= A_{00}^{X \rightarrow Y} e^{-m_X t - m_Y (T-t)} [1 + c^{X \rightarrow Y}(0, t, T) \\ &\quad + \dots], \end{aligned} \quad (21)$$

where in the last line we have pulled out the ground state amplitude and exponential dependence. The function  $c^{X \rightarrow Y}(0, t, T)$  is defined

$$\begin{aligned} c^{X \rightarrow Y}(0, t, T) &\equiv \frac{A_{01}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^{T-t} e^{-\Delta m_Y (T-t)} \\ &\quad + \frac{A_{10}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^t e^{-\Delta m_X t} \\ &\quad + \frac{A_{11}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^T e^{-\Delta m_X t - \Delta m_Y (T-t)}, \end{aligned} \quad (22)$$

where  $\Delta m_{X,Y} = m'_{X,Y} - m_{X,Y}$  is the splitting between the lowest-lying desired-parity state and the lowest-lying wrong-parity state. Note that the first two terms produce oscillations as the position of the operator is varied over the

time extent of the lattice. The third term, however, changes sign only when the total source-sink separation is varied. It is this term that our average is designed to suppress, since it will not be as clearly visible in the  $t$  dependence of the lattice data as those that oscillate in  $t$ .

We define the average to be

$$\bar{C}^{X \rightarrow Y}(0, t, T) \equiv \frac{1}{2}C^{X \rightarrow Y}(0, t, T) + \frac{1}{4}C^{X \rightarrow Y}(0, t, T+1) + \frac{1}{4}C^{X \rightarrow Y}(0, t+1, T+1). \quad (23)$$

Substituting the expression for  $C^{X \rightarrow Y}(0, t, T)$  from Eq. (21) into this definition gives

$$\begin{aligned} \bar{C}^{X \rightarrow Y}(0, t, T) &= A_{00}^{X \rightarrow Y} e^{-m_X t - m_Y(T-t)} \\ &\times [1 + \bar{c}^{X \rightarrow Y}(0, t, T) + \dots], \end{aligned} \quad (24)$$

where the function  $\bar{c}^{X \rightarrow Y}$  is

$$\begin{aligned} \bar{c}^{X \rightarrow Y}(0, t, T) &\equiv \frac{A_{01}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^T e^{-\Delta m_Y(T-t)} \\ &\times \left[ \frac{1}{2} + \frac{1}{4}(1 - e^{-\Delta m_Y}) \right] \\ &+ \frac{A_{10}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^t e^{-\Delta m_X t} \left[ \frac{1}{2} + \frac{1}{4}(1 - e^{-\Delta m_X}) \right] \\ &+ \frac{A_{11}^{X \rightarrow Y}}{A_{00}^{X \rightarrow Y}} (-1)^T e^{-\Delta m_X t - \Delta m_Y(T-t)} \\ &\times \left[ \frac{1}{2} - \frac{1}{4}(e^{-\Delta m_Y} + e^{-\Delta m_X}) \right]. \end{aligned} \quad (25)$$

Note that Eq. (25) has the same exponential time dependence as Eq. (22), but with the size of the amplitudes reduced by the factors in square brackets. Thus, the average is equivalent to a smearing that reduces the oscillating state amplitudes. It is possible to compute the  $\Delta m_X$  precisely from fits to two-point correlators. We find values between about 0.2 and 0.4 in lattice units. Given these values, the first two factors in brackets reduce their respective amplitudes by approximately a factor of 2, and the targeted, nonoscillating term is reduced by a factor of  $\sim 6\text{-}10$ .

Specializing to the  $B \rightarrow D^*$  case, consider the double ratio

$$\begin{aligned} R_{A_1}(0, t, T) &= \frac{A_{00}^{B \rightarrow D^*} A_{00}^{D^* \rightarrow B}}{A_{00}^{D^* \rightarrow D^*} A_{00}^{B \rightarrow B}} [1 + c^{B \rightarrow D^*}(0, t, T) \\ &+ c^{D^* \rightarrow B}(0, t, T) - c^{D^* \rightarrow D^*}(0, t, T) \\ &- c^{B \rightarrow B}(0, t, T) + \dots], \end{aligned} \quad (26)$$

where we have again factored out the ground state contribution. Equation (26) follows from Eq. (18) treating the  $c$ 's as small. Note that the  $c$ 's are expected to be similar in numerator and denominator, and to the extent that they are the same they will cancel in this expression. Applying the average in Eq. (23) directly to the double ratio,

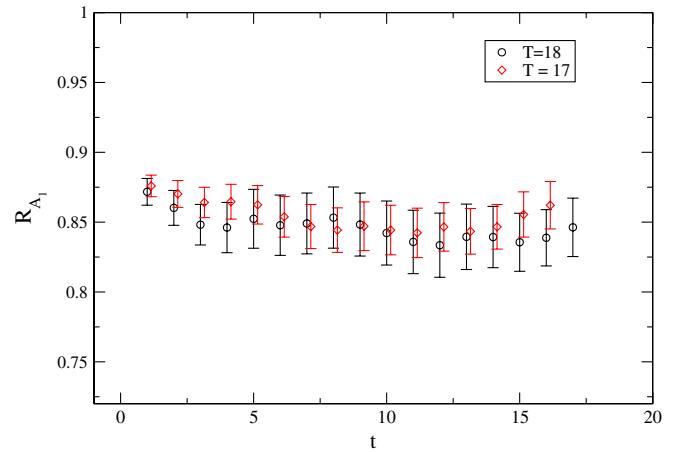


FIG. 1 (color online). Double ratio  $R_{A_1}$  on the  $a\hat{m}' = 0.0124$  fine ( $a = 0.09$  fm) ensemble. The source was fixed to time slice 0, and the operator position was varied as a function of time. Two different sink points were used with even and odd time separations between source and sink in order to study the effect of nonoscillating contributions from wrong parity states.

$$\bar{R}(0, t, T) \equiv \frac{1}{2}R(0, t, T) + \frac{1}{4}R(0, t, T+1) + \frac{1}{4}R(0, t+1, T+1), \quad (27)$$

we get

$$\begin{aligned} \bar{R}_{A_1}(0, t, T) &= \frac{A_{00}^{B \rightarrow D^*} A_{00}^{D^* \rightarrow B}}{A_{00}^{D \rightarrow D^*} A_{00}^{B \rightarrow B}} [1 + \bar{c}^{B \rightarrow D^*}(0, t, T) \\ &+ \bar{c}^{D \rightarrow B^*}(0, t, T) - \bar{c}^{D \rightarrow D^*}(0, t, T) \\ &- \bar{c}^{B \rightarrow B^*}(0, t, T) + \dots], \end{aligned} \quad (28)$$

where each of the oscillating state terms in the individual three-point functions is suppressed according to Eq. (25).

Although  $\Delta m_B$  and  $\Delta m_{D^*}$  can be obtained from fits to the two-point correlators, the oscillating state amplitudes

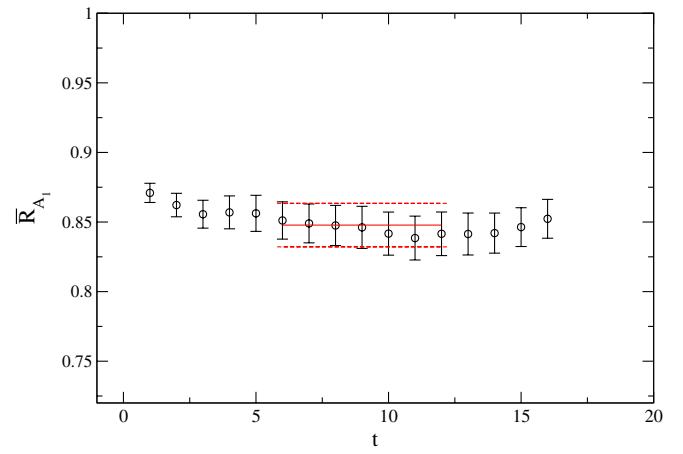


FIG. 2 (color online). Averaged double ratio,  $\bar{R}_{A_1}$ , of Eq. (27) on the  $a\hat{m}' = 0.0124$  fine ( $a = 0.09$  fm) ensemble. The plateau fit is shown with  $1\sigma$  error band.

appearing in the three-point correlators must be determined directly from the three-point correlator data. Figure 1 shows the double ratio  $R_{A_1}$  used to obtain  $h_{A_1}(1)$ . The source is at time slice 0, the sink is at  $T$ , and the operator position is varied along  $t$ . Two different source-sink separations were generated that differed by a single time unit at the sink ( $T = 17, 18$ ). The average of these two correlators was taken according to Eq. (27), and this average was fit (including the full covariance matrix) to a constant, as shown in Fig. 2. There is no detectable oscillation even before the average is taken, as can be seen in Fig. 1; according to Eq. (25) the oscillating contributions are reduced even further in the average so that their systematic errors can be safely neglected.

#### IV. PERTURBATION THEORY

Lattice perturbation theory is needed in order to calculate the short-distance coefficient  $\rho_{A_1}$  defined in Eq. (19). Although naive lattice perturbation theory appears to converge slowly, the two main causes have been identified [50]: the bare gauge coupling is a poor expansion parameter, and coefficients are large when tadpole diagrams occur. If a renormalized coupling is used as an expansion parameter, and one computes only those quantities for which the tadpole diagrams largely cancel, then lattice perturbation theory seems to converge as well as perturbation theory in continuum QCD.

Only the vertex correction contributes to the  $\rho$  factor, as the wave-function renormalization (including all tadpoles) cancels by construction. Even the vertex correction partially cancels, and the one-loop coefficient is found to be small. The perturbative corrections to the  $\rho$  factor can be written as

$$\rho_{J_\Gamma}^{hh'} \equiv \frac{Z_{J_\Gamma}^{hh'}}{\sqrt{Z_{V_4}^{hh'} Z_{V_4}^{h'h}}} = 1 + \alpha_V(q^*) 4\pi \rho_{J_\Gamma}^{hh'[1]} + \dots, \quad (29)$$

where  $\rho_{J_\Gamma}^{hh'[1]}$  is the coefficient of the one-loop correction, and the coupling  $\alpha_V$  is the renormalized strong coupling constant in the V scheme [50,62], which is based on the static-quark potential. The coupling is determined following the procedure of Ref. [63]. The scale  $q^*$  of the running coupling  $\alpha_V(q^*)$  should be chosen to be the typical momentum of a gluon in the loop. A prescription for calculating this scale was introduced by Brodsky, Lepage, and Mackenzie [50,62]. They defined  $q^*$  by

$$\ln(q^{*2}) = \frac{\int d^4 q f(q) \ln(q^2)}{\int d^4 q f(q)}, \quad (30)$$

where  $f(q)$  is the one-loop integrand and the numerator is the first log moment. This prescription was extended by Hornbostel, Lepage, and Morningstar (HLM) [64] to cases where the one-loop contribution is anomalously small leading to a break down of Eq. (30). The HLM prescription

TABLE III. Computed values of  $\rho_{A_1}$  in the HLM prescription [64]. The first three columns label each ensemble with the approximate lattice spacing in fm, the light sea quark mass  $a\hat{m}'$ , and the strange quark mass  $a\hat{m}'_s$ . The fourth column is  $aq_{\text{HLM}}^*$ , where the error is calculated using the statistical error from VEGAS for the 0th, 1st, and 2nd moments of the one-loop integrals. The fifth column is  $\rho_{A_1}$  on that ensemble, and the errors are the statistical errors from the VEGAS evaluation, including the one-loop coefficients and  $q_{\text{HLM}}^*$ .

$a$ (fm)	$a\hat{m}'$	$a\hat{m}'_s$	$aq_{\text{HLM}}^*$	$\rho_{A_1}$
0.15	0.0194	0.0484	2.03(10)	0.9966(2)
0.15	0.0097	0.0484	2.03(10)	0.9966(2)
0.12	0.02	0.05	1.96(10)	0.9964(2)
0.12	0.01	0.05	1.96(10)	0.9964(2)
0.12	0.007	0.05	1.96(10)	0.9964(2)
0.12	0.005	0.05	1.96(10)	0.9964(2)
0.09	0.0124	0.031	2.98(14)	1.00298(9)
0.09	0.0062	0.031	2.98(14)	1.00300(9)
0.09	0.0031	0.031	2.98(14)	1.00301(9)

for  $q^*$  takes into account two-loop contributions to the gluon propagator via the inclusion of second log moments. Since we do encounter anomalously small one-loop corrections in  $\rho_{A_1}$ , the HLM prescription was used to determine  $q^*$ . Results for  $q_{\text{HLM}}^*$  and  $\rho_{A_1}$  needed for this calculation are given in Table III. The  $\rho$  factor varies somewhat as a function of lattice spacing, and is even slightly different from ensemble to ensemble at the same nominal lattice spacing, due to the slightly different  $\beta$  values used to generate the gauge fields.

The calculation of  $\rho_{A_1}$  is described in Refs. [65,66]. It uses automated perturbation theory techniques to generate the Feynman rules and VEGAS [67] for the numerical integration of the loop integrals. As a check, it was verified that this calculation reproduces known results for the heavy-heavy currents with the Wilson plaquette action [29] and for the  $V_4$  current in the massless limit with the Symanzik improved gauge action.

As mentioned in the introduction, we have exploited the  $\rho$  factor to implement a blind analysis. Two of us applied a multiplicative offset close to 1 to the  $\rho$  factor, generated with a random key. The offset was not unlocked until the procedure for determining the systematic errors in the rest of the analysis had been finalized.

#### V. STAGGERED CHIRAL PERTURBATION THEORY

The simulation masses  $\hat{m}'_{\text{val}}$  and  $\hat{m}'_{\text{sea}}$  (for valence and sea) are all larger than the physical  $\hat{m}$ . A controlled chiral extrapolation can be guided by an appropriate chiral effective theory that includes the effect of staggered-quark discretization errors. Rooted staggered chiral perturbation theory (rS $\chi$ PT), which has been formulated for heavy-light quantities in Ref. [68], is such a theory. In rS $\chi$ PT, a replica

method is used to take into account the effect of rooting; this procedure has been justified in Refs. [33,69].

Because of taste-symmetry breaking, the staggered theory has 16 light pseudoscalar mesons instead of 1. The tree-level relation for the masses of light staggered mesons in the chiral theory is [17,18]

$$m_{xy,\Xi}^2 = \mu_0(m_x + m_y) + a^2 \Delta_\Xi, \quad (31)$$

where  $m_x$  and  $m_y$  are staggered-quark masses,  $\mu_0$  is the continuum low-energy constant, and  $a^2 \Delta_\Xi$  are the splittings of the 16 pions of taste  $\Xi$ . For staggered quarks there exists a residual SO(4) taste symmetry broken at  $\mathcal{O}(a^2)$ , such that there is some degeneracy among the 16 pions [17], and the taste index  $\Xi$  runs over the multiplets  $P, A, T, V, I$  with degeneracies 1, 4, 6, 4, 1. The splitting  $a^2 \Delta_P$  vanishes because there is an exact (nonsinglet) lattice axial symmetry.

Schematically, the next-to-leading-order (NLO) result for the relevant form factor is

$$h_{A_1}^{\text{NLO}}(1)/\eta_A = 1 + X_A(\Lambda_\chi) + \frac{g_{DD^*\pi}^2}{48\pi^2 f^2} \times \log_{1-\text{loop}}(\Lambda_\chi), \quad (32)$$

where  $X_A(\Lambda_\chi)$  is a low-energy constant of the chiral effective theory, and is therefore independent of light-quark mass and cancels the chiral scale dependence  $\Lambda_\chi$  of the chiral logarithms. By heavy-quark symmetry,  $X_A(\Lambda_\chi)$  is proportional to  $1/m_c^2$  in the heavy-quark expansion. The term  $\eta_A$  is a factor that matches heavy-quark effective theory to QCD, and contains perturbative-QCD logarithmic dependence on the heavy-quark masses; it is independent of light-quark mass. The term proportional to  $g_{DD^*\pi}^2$  is shorthand for the one-loop staggered chiral logarithms, and is given in the appendix for ease of reference. The rooted staggered expression was derived in Ref. [22]. The one-loop staggered logarithms depend on both valence and sea quark masses, and include taste-breaking effects coming from the light-quark sector. This expression also contains explicit dependence on the lattice spacing  $a$ , and requires as inputs the parameters of the staggered chiral Lagrangian  $\delta'_V$ ,  $\delta'_A$ , in addition to the staggered taste splittings  $\Delta_{P,A,T,V,I}$  [16]. These parameters can be obtained from chiral fits to the light pseudoscalar meson sector and are held fixed in the chiral extrapolation of  $h_{A_1}(1)$ . The continuum low-energy constant  $g_{DD^*\pi}$  appears, and below we take a generous range inspired by a combined fit to many different experimental inputs, including a leading-order analysis of the  $D^*$  width. The  $D^*-D$  splitting  $\Delta^{(c)}$  is well determined from experiment. The only other parameter that appears at NLO is the constant  $X_A(\Lambda)$ , and this is determined by our lattice data for  $h_{A_1}(1)$ .

Although the lattice data are well described by the NLO formula, it is useful to go beyond NLO and to include the next-to-next-to-leading-order (NNLO) analytic terms as a

way to estimate systematic errors. We do not include the NNLO logarithms because they are unknown and would require a two-loop calculation. The expression including analytic terms through NNLO is

$$h_{A_1}^{\text{NNLO}}(1)/\eta_A = 1 + \text{NLO} + c_1 m_{X_P}^2 + c_2 (2m_{U_P}^2 + m_{S_P}^2) + c_3 a^2, \quad (33)$$

where the subscript  $P$  on the meson masses indicates the taste pseudoscalar mass. We use the notation from the rS $\chi$ PT literature that  $m_{X_\Xi}$  is a taste  $\Xi$  meson made of two valence  $x$  quarks,  $m_{U_\Xi}$  is a taste  $\Xi$  meson made of two light sea quarks, and  $m_{S_\Xi}$  is a taste  $\Xi$  meson made of two strange sea quarks. By heavy-quark symmetry, the  $c_i$  are suppressed by a factor of  $1/m_c^2$ . Since the only free parameter through NLO is an overall constant, we include the NNLO analytic terms in the fit used for our central value. This leads to a larger statistical error and is more conservative.

## VI. TREATMENT OF CHIRAL EXTRAPOLATION

In this section, we discuss the approach we have developed to disentangle the heavy- and light-quark discretization effects and to perform the chiral and continuum extrapolations. In the Fermilab method, heavy-quark discretization errors can be estimated by comparing the heavy-quark expansions for lattice gauge theory and continuum QCD [28–30,70]. The dependence on  $a$  is not simply a power series (unless  $ma \ll 1$ ), so power-counting estimates in HQET are used. On the other hand, some of the light-quark discretization effects are constrained by rS $\chi$ PT. The heavy-quark errors are asymptotically constrained by the Symanzik low-energy Lagrangian when  $m_h a \ll 1$  and by heavy-quark symmetry even when  $m_h a$  is close to 1. In the region in between, the errors smoothly interpolate the asymptotic behavior [49,70]. The errors in the SW action used for the heavy quarks decrease with lattice spacing as  $\alpha_s a$  in the  $m_h a \ll 1$  region, as compared with the light-quark (improved staggered) discretization errors, which decrease much faster, as  $\alpha_s a^2$ .

The first step of the method is to normalize the numerical data for  $h_{A_1}(1)$  to a fiducial point by forming the ratio

$$\mathcal{R}_{\text{fid}}(m_x, \hat{m}', m'_s, a) \equiv \frac{h_{A_1}(m_x, \hat{m}', m'_s, a)}{h_{A_1}^{\text{fid}}(m_x^{\text{fid}}, \hat{m}^{\text{fid}}, m_s^{\text{fid}}, a)}, \quad (34)$$

where  $m^{\text{fid}}$  is a fiducial mass,  $m_x$  is the light (spectator) valence quark,  $\hat{m}'$  is the isospin averaged light sea quark on a particular ensemble, and  $m'_s$  is the strange sea quark on that ensemble. (Note that the factor of  $\eta_A$  in Eqs. (32) and (33) cancels in the ratio.) The principle advantage of this ratio is that heavy-quark discretization effects largely cancel, since the heavy quarks are the same in numerator and denominator. This allows us to disentangle the heavy-quark discretization effects from those of the light-quark

sector coming from staggered chiral logarithms, thus isolating the (taste-violating) discretization effects specific to the staggered light quarks. These light-quark discretization effects can appear in nonanalytic terms in rS $\chi$ PT and are due to violations of taste symmetry. They can be removed to a given order in rS $\chi$ PT (we work to NLO) in fits to the numerical data at multiple lattice spacings using the explicit rS $\chi$ PT formula of Eq. (33), since this formula includes the staggered lattice artifacts. The continuum limit of the ratio  $\mathcal{R}_{\text{fid}}$  can be obtained using our fitted values for parameters in rS $\chi$ PT and taking  $a \rightarrow 0$  in the rS $\chi$ PT expression for  $\mathcal{R}_{\text{fid}}$ . We do not need a more explicit ansatz for the functional form of the heavy-quark discretization effects, since they largely cancel in the ratio.

Normalizing the continuum extrapolated ratio  $\mathcal{R}_{\text{fid}}$  by  $h_{A_1}$  at the fiducial point on a very fine fiducial lattice where the heavy-quark discretization effects are small gives a value close to the physical continuum result,

$$h_{A_1}(\hat{m}, \hat{m}, m_s, 0) \approx h_{A_1}(m_x^{\text{fid}}, \hat{m}^{\text{fid}}, m_s^{\text{fid}}, a^{\text{fid}}) \\ \times \mathcal{R}_{\text{fid}}(\hat{m}, \hat{m}, m_s, 0), \quad (35)$$

where the relation becomes exact as  $a^{\text{fid}} \rightarrow 0$ . Note that the requirement that the heavy-quark discretization effects must be small enforces the condition that the improved staggered light-quark discretization effects be even smaller (and likely negligible) because the staggered discretization effects decrease much faster with lattice spacing. The fiducial masses  $m_x^{\text{fid}}$ ,  $\hat{m}^{\text{fid}}$ , and  $m_s^{\text{fid}}$  should be chosen large enough that it would be feasible to simulate this mass point on a very fine lattice (since the cost rises significantly as the mass of the light sea quarks is decreased), thus normalizing the lattice data to a point where the heavy-quark discretization effects are small. The fiducial masses should not be chosen so large, however, that rS $\chi$ PT would not be a reliable guide in performing the continuum and chiral extrapolation of  $\mathcal{R}_{\text{fid}}$ . This method can be considered the crudest form of step scaling, but it does illustrate that one does not need lattices, which are simultaneously fine enough for  $b$  quarks and large enough for light quarks in order to simulate, with high precision, quantities that involve both. In practice, we find  $m_x^{\text{fid}} = \hat{m}^{\text{fid}} \approx 0.4m_s$  and  $m_s^{\text{fid}} \approx m_s$  are reasonable values for the fiducial masses. The fiducial lattice spacing should be chosen as fine as is practical; a succession of progressively finer fiducial lattices would be desirable for verifying that the  $a$  dependence is of the expected size. In this work we take our finest lattice (0.09 fm) as the fiducial lattice, but we apply Eq. (35) with the coarser lattices taken as fiducial lattices in order to estimate discretization errors. We note that the method presented above can be applied to all calculations involving the Fermilab treatment of heavy quarks and staggered light quarks, not only the  $B \rightarrow D^* \ell \bar{\nu}$  form factor  $h_{A_1}$ . It may also be desirable to compute quantities at the fiducial point (or a succession of such points) using an even

further improved action for the heavy quarks. Once the fiducial lattice spacing is of the order 0.03–0.01 fm, even the bottom quark may be treated as a “light” quark with the highly improved staggered action [71] or with chiral fermions, for which mass dependent discretization effects are small. Conserved currents could then be used for many simple heavy-light quantities, removing the need for a perturbative renormalization.

For the chiral extrapolation of  $h_{A_1}$  we find it useful to form two additional ratios

$$\mathcal{R}_{\text{sea}}(\hat{m}', m_s', a) \equiv \frac{h_{A_1}(m_x^{\text{fid}}, \hat{m}', m_s', a)}{h_{A_1}(m_x^{\text{fid}}, \hat{m}^{\text{fid}}, m_s', a)}, \quad (36)$$

$$\mathcal{R}_{\text{val}}(m_x, \hat{m}', m_s', a) \equiv \frac{h_{A_1}(m_x, \hat{m}', m_s', a)}{h_{A_1}(m_x^{\text{fid}}, \hat{m}', m_s', a)}, \quad (37)$$

whose product is clearly  $\mathcal{R}_{\text{fid}}$ , Eq. (34).  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$  separate the sea and valence quark mass dependence, which makes it easier to assess systematic errors. The values of  $h_{A_1}$  that enter Eqs. (36) and (37) are obtained from

$$h_{A_1} = \rho \sqrt{\bar{R}_{A_1}}, \quad (38)$$

where  $\bar{R}_{A_1}$  is the average of double ratios defined in Eq. (28). The ratios in Eqs. (36) and (37) are now quadruple ratios, where the excited-state contamination is further suppressed over that of the double ratio. Performing the chiral extrapolation, taking the continuum limit of the two ratios, and multiplying them together we recover  $\mathcal{R}_{\text{fid}}(\hat{m}, \hat{m}, m_s, 0)$  by construction. Thus, we can rewrite Eq. (35) as

$$h_{A_1}^{\text{phys}} \approx h_{A_1}(m_x^{\text{fid}}, \hat{m}^{\text{fid}}, m_s^{\text{fid}}, a^{\text{fid}}) \\ \times [\mathcal{R}_{\text{sea}}(\hat{m}, m_s, 0) \times \mathcal{R}_{\text{val}}(\hat{m}, \hat{m}, m_s, 0)], \quad (39)$$

where, again, the relation becomes exact as  $a^{\text{fid}} \rightarrow 0$ .

To the extent that the extrapolation in sea quark masses is mild, the ratio  $\mathcal{R}_{\text{sea}}$  should be close to 1, since the valence light mass is the same in both numerator and denominator.  $\mathcal{R}_{\text{val}}$  contains less trivial chiral behavior. However, since the numerator and denominator are computed on the same ensemble (with different valence masses), they are correlated, and statistical errors tend to cancel in  $\mathcal{R}_{\text{val}}$ . The ratio  $\mathcal{R}_{\text{sea}}$  has small statistical errors because the valence mass  $m_x^{\text{fid}}$  in that ratio is relatively heavy. Of course, the heavy-quark discretization errors are significantly suppressed in both ratios, isolating the light-quark mass dependence and staggered discretization effects. A direct chiral fit to the numerical data (not involving the ratios introduced here) would require a more explicit

TABLE IV. Fiducial masses used at the three different lattice spacings. The first four columns are the approximate lattice spacing in fm, the fiducial valence quark mass, the fiducial light sea quark mass, and the fiducial strange quark mass. The fifth and sixth columns are the values of  $\sqrt{\bar{R}_{A_1}}$  and  $h_{A_1}^{\text{fid}}$ , respectively, computed at that fiducial point.

Lattice spacing (fm)	$am_x^{\text{fid}}$	$a\hat{m}^{\text{fid}}$	$am_s^{\text{fid}}$	$\sqrt{\bar{R}_{A_1}}$	$h_{A_1}^{\text{fid}}$
0.15	0.0194	0.0194	0.0484	0.9211(73)	0.9180(73)
0.12	0.02	0.02	0.05	0.9112(73)	0.9079(73)
0.09	0.0124	0.0124	0.031	0.9210(85)	0.9237(85)

ansatz for the treatment of the heavy-quark discretization effects than is needed in the ratio fits.<sup>2</sup> Note that in the ratios the fiducial point need not be tuned to the same mass at every lattice spacing; differences can be accounted for in the fit itself. The fiducial points used at different lattice spacings are  $m_x^{\text{fid}} = \hat{m}^{\text{fid}} = 0.4m'_s$  and  $m_s^{\text{fid}} = m'_s$ . The explicit values are given in Table IV, along with the calculated values of  $\sqrt{\bar{R}_{A_1}}$  and  $h_{A_1}^{\text{fid}}$  at that fiducial point.

The constant term  $X_A(\Lambda_\chi)$  in Eq. (32) cancels in the ratios  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$ , so the behavior of these ratios is completely predicted through NLO in the chiral expansion. We find good agreement between the predicted form and the numerical data. However, given that our fiducial spectator quark mass is rather large (around  $0.4m_s$ ), we include the NNLO analytic terms in the ratio fits in order to estimate systematic errors associated with the chiral expansion. There are only two new continuum low-energy constants introduced at this higher order, and the ratios  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$  determine one each. There is also an analytic term proportional to  $a^2$  appearing at this order, but it cancels in each of the  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$  ratios.

In future calculations, it would be feasible to use a much finer lattice spacing for the fiducial point, thereby further reducing heavy-quark discretization errors. For now, however, we use  $h_{A_1}(m_x^{\text{fid}}, \hat{m}^{\text{fid}}, m_s^{\text{fid}}, 0.09 \text{ fm})$ , with the fiducial masses in Table IV, in Eq. (39). As a way to estimate discretization errors we use our results for  $h_{A_1}^{\text{fid}}$  at the two coarser lattice spacings in Eq. (39) also.

At the lattice spacings used in this work the light-quark discretization effects may still be non-negligible compared with heavy-quark discretization effects. With rS $\chi$ PT it is possible to remove from  $h_{A_1}^{\text{fid}}$  the discretization effects associated with staggered chiral logarithms, although purely analytic discretization errors remain. Removing this subset of staggered effects leads to a value for the

<sup>2</sup>A direct (correlated) chiral fit would still, however, reflect the correlations, which cause cancellations in the statistical errors in the ratios.

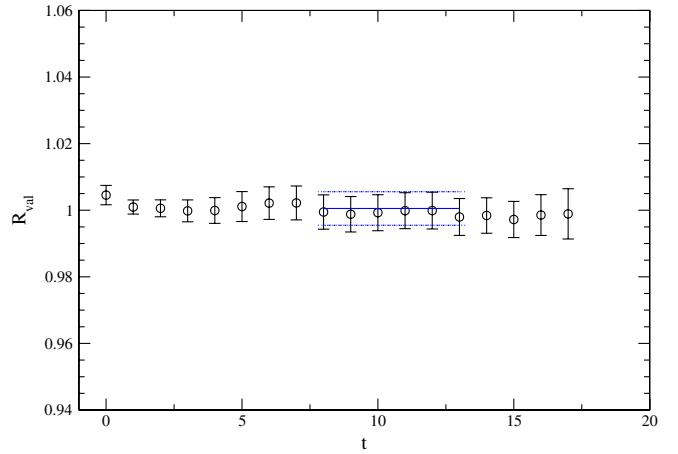


FIG. 3 (color online).  $\mathcal{R}_{\text{val}}$  on the  $a\hat{m}' = 0.0062$  fine ensemble. The valence mass in the numerator is the full QCD value of  $am'_x = 0.0062$  while the fiducial valence mass in the denominator is  $am_x^{\text{fid}} = 0.0124$ . The fit to a constant has a  $\chi^2/\text{d.o.f} = 0.20$ .

fiducial form factor, which we call the “taste-violations-out” value. Not removing them leads to the “taste-violations-in” value. The difference turns out to be negligible, less than 0.1% on our coarsest ensemble and less than 0.01% on the fine ensemble. Thus, the discretization effects in our lattice data coming from taste violations in the staggered chiral logarithms are extremely small at the fiducial point mass, and we neglect this difference in the analysis.

Figure 3 shows the plateau fit to the ratio  $\mathcal{R}_{\text{val}}$  on the fine ensemble with  $(a\hat{m}', am'_s) = (0.0062, 0.031)$ . The valence mass in the numerator is the full QCD value of  $am'_x = 0.0062$ , while the fiducial valence mass in the denominator is  $am_x^{\text{fid}} = 0.0124$ . Both numerator and denominator are computed on the same ensemble, so they have the same sea quark masses, and correlated statistical errors largely cancel in the ratio, as expected. Excited-state contamination is also reduced. Computed values for  $\mathcal{R}_{\text{sea}}$  on all of our ensembles are given in Table V, and the computed values for  $\mathcal{R}_{\text{val}}$  are given in Table VI.

TABLE V. Computed values of  $\mathcal{R}_{\text{sea}}$ . The first three columns are the arguments of  $\mathcal{R}_{\text{sea}}$  as defined in Eq. (36); they are the light sea quark mass  $\hat{m}'$ , the strange quark mass  $m'_s$ , and the approximate lattice spacing in fm. The fourth column is  $\mathcal{R}_{\text{sea}}$ .

$a\hat{m}'$	$am'_s$	$a$ (fm)	$\mathcal{R}_{\text{sea}}$
0.0097	0.0484	0.15	1.009(12)
0.01	0.05	0.12	1.0070(98)
0.007	0.05	0.12	1.0027(91)
0.005	0.05	0.12	1.014(10)
0.0062	0.031	0.09	1.000(12)
0.0031	0.031	0.09	0.996(10)

TABLE VI. Computed values of  $\mathcal{R}_{\text{val}}$ . The first four columns are the arguments of  $\mathcal{R}_{\text{val}}$  as defined in Eq. (37); they are the light valence quark mass  $m_x$ , the light sea quark mass  $\hat{m}'$ , the strange quark mass  $m_s'$ , and the approximate lattice spacing in fm. The fifth column is  $\mathcal{R}_{\text{val}}$ .

$am_x$	$a\hat{m}'$	$am_s'$	$a$ (fm)	$\mathcal{R}_{\text{val}}$
0.0097	0.0097	0.0484	0.15	1.0056(65)
0.01	0.01	0.05	0.12	0.9994(41)
0.007	0.007	0.05	0.12	0.9900(57)
0.005	0.005	0.05	0.12	1.0081(90)
0.0062	0.0062	0.031	0.09	1.0005(50)
0.0031	0.0031	0.031	0.09	1.0043(62)

## VII. SYSTEMATIC ERRORS

In the following subsections, we examine the uncertainties in our calculation due to fitting and excited states, the heavy-quark mass dependence, the chiral extrapolation of the light spectator quark mass, discretization errors, and perturbation theory. As mentioned in Sec. II, statistical uncertainties are computed with a single elimination jackknife and the full covariance matrix.

### A. Fitting and excited states

We have examined plateau fits to the time dependence of the double and quadruple ratios introduced in Secs I and V. The  $\chi^2$  in our fits is defined with the full covariance matrix. The fits to the ratios were done under a single elimination jackknife, after blocking the numerical data by 8 on the fine lattices and by 4 on the coarse and coarser lattices. The blocking procedure averages 4 (or 8) successive configurations before performing the single elimination jackknife. These values for the block size were chosen such that the statistical error on the double ratio fit did not increase when a larger block size was used. Statistical errors were determined in fits that included the full correlation matrix, which was remade for each jackknife fit. The jackknife data sets on different ensembles were then combined into a larger block-diagonal jackknife data set in order to perform the chiral fits. In this way, the fully correlated statistical errors were propagated through to the final result.

With our high statistics (several hundred lattice gauge field configurations for each ensemble), we are able to resolve the full covariance matrix well enough that we do not need to apply a singular value decomposition cut on the eigenvalues of the covariance matrix. The double ratio fit is needed to establish  $h_{A_1}(1)$  at the fiducial point (which was computed on the 0.0124/0.031 fine ensemble), while the quadruple ratios,  $\mathcal{R}_{\text{val}}$  and  $\mathcal{R}_{\text{sea}}$  are computed on the other ensembles in order to perform the chiral extrapolation and to remove taste-breaking nonanalytic terms. We find that the fit to the double ratio at the fiducial point on the 0.0124/0.031 ensemble is well described by a constant over a range of seven time slices. The excited-state con-

tamination in the quadruple ratios is even further suppressed, and we find that the correlated  $\chi^2$  values allow for a constant fit region of six to ten time slices, depending upon the lattice spacing. We take the good correlated  $\chi^2/\text{d.o.f.}$ , ranging from 0.15 to 1.00, in our constant plateau fits as evidence that the excited-state contamination in these fits is negligible as compared with other errors.

As an additional check of the jackknife fitting procedure, bootstrap fits were done to all of the double and quadruple ratios needed for this work. Close agreement was found for both central values and statistical errors. The statistical errors were typically the same size within 10%, and central values were well within  $1\sigma$ . The jackknife procedure had slightly larger errors than that of the bootstrap.

### B. Heavy-quark mass dependence

The value for  $h_{A_1}(1)$  depends on the heavy-quark masses, which are set by tuning the hopping parameters  $\kappa_b$  and  $\kappa_c$ . The principal method starts by fitting the lattice pole energy to  $E(\mathbf{p})$  to the dispersion relation

$$E(\mathbf{p}) = M_1 + \frac{\mathbf{p}^2}{2M_2} + b_1 \mathbf{p}^4 + b_2 \sum_{j=1}^3 |p_j|^4 + \dots, \quad (40)$$

in order to obtain the kinetic mass  $M_2$  (as well as  $b_1$  and  $b_2$ , which are unimportant here). In the Fermilab method [28,30,49],  $\kappa$  is adjusted so that the kinetic mass agrees with experiment. Here we take the spin-average of kinetic masses of pseudoscalar and vector heavy-strange mesons and obtain our central values for  $\kappa_b$  or  $\kappa_c$ , respectively, from the (spin-averaged)  $B_s^{(*)}$  and  $D_s^{(*)}$  masses. Applying this procedure we find statistical and fitting errors of 5.6% for  $\kappa_b$  and 1.2% for  $\kappa_c$  on the fine ( $a = 0.09$  fm) ensembles. There is an additional error in  $\kappa$  due to discretization effects. We determine this error by estimating the size of discretization effects for the Fermilab action (at  $a = 0.09$  fm) as in Ref. [72]. This error is 1.3% for  $\kappa_b$  and 0.3% for  $\kappa_c$ . Adding in quadrature the statistical and fitting error together with the discretization error leads to a total relative uncertainty of 5.7% for  $\kappa_b$  and 1.2% for  $\kappa_c$ . This error budget is summarized in Table VII. Note that these errors are conservative and are likely to decrease substantially with more sophisticated fitting methods and the higher statistics data set currently being generated.

We have computed  $h_{A_1}(1)$  at several different values of the bare charm and bottom quark masses, and these simu-

TABLE VII. Errors in the  $\kappa_{b,c}$  parameters. The first column labels the heavy quark, the second gives the statistical and fitting error for the  $\kappa$  parameter, the third gives the discretization error, and the fourth combines these in quadrature.

$\kappa$	Statistics + fitting	Discretization	Total
$\kappa_c$	1.2%	0.3%	1.2%
$\kappa_b$	5.6%	1.3%	5.7%

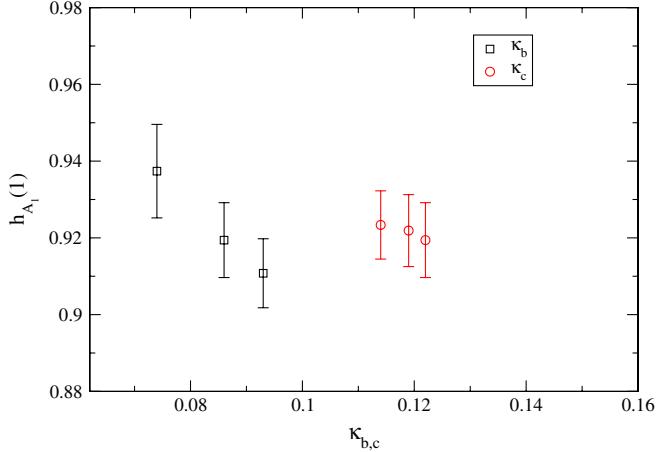


FIG. 4 (color online).  $h_{A_1}(1)$  for different  $\kappa_h$  values on the coarse  $\hat{m}' = 0.02$  ensemble (full QCD point). The points labeled  $\kappa_b$  show how  $h_{A_1}(1)$  depends on  $\kappa_b$  when  $\kappa_c$  is fixed to its tuned value. For the points labeled  $\kappa_c$  the roles of  $\kappa_b$  and  $\kappa_c$  are reversed.

lated points can be used to estimate the error in  $h_{A_1}(1)$  from the above uncertainties in the tuning of the heavy-quark  $\kappa$  values. Figure 4 illustrates the dependence of  $h_{A_1}(1)$  as a function of bottom and charm quark  $\kappa$  values on one of the coarse ( $a = 0.12$  fm) ensembles. The points labeled  $\kappa_b$  show  $h_{A_1}(1)$  where we have fixed  $\kappa_c$  to the tuned charm value, but vary the bare  $\kappa_b$  along the  $x$  axis. The points labeled  $\kappa_c$  are similar, where the value of  $\kappa_b$  is fixed at its tuned value, and the bare  $\kappa_c$  is varied. The above uncertainties in the  $\kappa$ 's, combined with the variation of  $h_{A_1}(1)$  with  $\kappa$ , lead to a systematic error of 0.7% in  $h_{A_1}(1)$ , labeled “kappa tuning” in Table X.

### C. Perturbation theory

The perturbative calculation of  $\rho_{A_1}$  is needed to match the heavy-quark lattice current, and the calculation has been carried out to one-loop order [ $O(\alpha_s)$ ]. As discussed in Sec. IV, much of the renormalization cancels when forming the ratios of  $Z$  factors that define  $\rho$  [Eq. (29)], and the coefficients of the perturbation series are small, by construction. The one-loop correction is quite small, only 0.3–0.4% on the different lattice spacings. We take the entire one-loop correction of 0.3% on the fine lattices as an estimate of the error introduced by neglecting higher orders in the perturbative expansion.

### D. Chiral extrapolation

We estimate our systematic error due to the chiral extrapolation by comparing fits with and without the additional terms with coefficients  $c_i$  in Eq. (33), i.e. analytic terms of higher order than NLO in rS $\chi$ PT, since the two-loop NNLO logarithms are unknown. We also compare with continuum  $\chi$ PT, both NLO and (partial) NNLO.

There are additional errors due to the uncertainties in the parameters that enter the NLO rS $\chi$ PT formulas. By far the largest uncertainty of this kind is that due to the uncertainty in  $g_{DD^*\pi}$ . Finally, there is an error due to a mistuning of the parameter  $u_0$  on the coarse lattices. All of these errors are discussed below in more detail. In the discussion of chiral extrapolation errors, it is important to keep in mind that the chiral logarithms (either rS $\chi$ PT or continuum) are tiny ( $\sim 3 \times 10^{-3}$ ) in the region where we have data. Nonanalytic behavior is important only near the physical pion mass where the  $\chi$ PT should be a very good description in the continuum. The main feature of the chiral extrapolation is a cusp that appears close to the physical pion mass (in the valence sector), due to the  $D\pi$  threshold and the fact that the  $D-D^*$  splitting is very close to the physical pion mass. This cusp represents real physics, and must be included in any version of the chiral extrapolation used to estimate systematic errors.

We extrapolate the light sea and light valence quark masses from the values used in the simulations, between  $m_s/2$  and  $m_s/10$ , to the average physical light-quark mass, around  $m_s/27$ . We use staggered chiral perturbation theory and the prescription introduced in Sec. VI to remove the nonanalytic taste-breaking discretization effects coming from the staggered light-quark sector. Separate fits are performed for the two ratios introduced in Eqs. (36) and (37),  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$ . The chiral extrapolation is performed on these ratios, and the staggered discretization errors appearing in the NLO chiral logarithms are removed by taking  $a \rightarrow 0$  in the rS $\chi$ PT expression. With the NNLO analytic terms given in Eq. (33) the chiral extrapolation formulas for the ratios are

$$\mathcal{R}_{\text{val}} = 1 + \text{NLO}_{\log} + c_1 m_{X_P}^2, \quad (41)$$

$$\mathcal{R}_{\text{sea}} = 1 + \text{NLO}_{\log} + c_2 (2m_{U_P}^2 + m_{S_P}^2), \quad (42)$$

where  $\text{NLO}_{\log}$  is a schematic notation representing the chiral logarithms coming from numerator and denominator. These terms are different for the two ratios, and can be obtained straightforwardly from the definitions of the ratios Eqs. (36) and (37), and the formula for the nonanalytic terms in Eq. (A1). The formula for  $\mathcal{R}_{\text{val}}$  in the continuum is given explicitly in Eq. (A6), for the purposes of illustration. The NNLO term  $c_3 a^2$  in Eq. (33) cancels in the ratios, and  $\mathcal{R}_{\text{sea}}$  and  $\mathcal{R}_{\text{val}}$  each determine one of the remaining two NNLO coefficients. Note that the factor of  $\eta_A$  in Eqs. (32) and (33) cancels in the chiral formulas for the two ratios. The only free parameters in our chiral fits are  $c_1$  and  $c_2$ ; the rest are determined from phenomenology or from rS $\chi$ PT fits to the pseudoscalar sector.

The ratios in Eqs. (36) and (37) are completely predicted through NLO in the continuum once  $f_\pi$ ,  $g_{DD^*\pi}$ , and the  $D-D^*$  splitting  $\Delta^{(c)}$  are taken from experiment. The constants  $f_\pi$  and  $g_{DD^*\pi}$  appear in an overall multiplicative

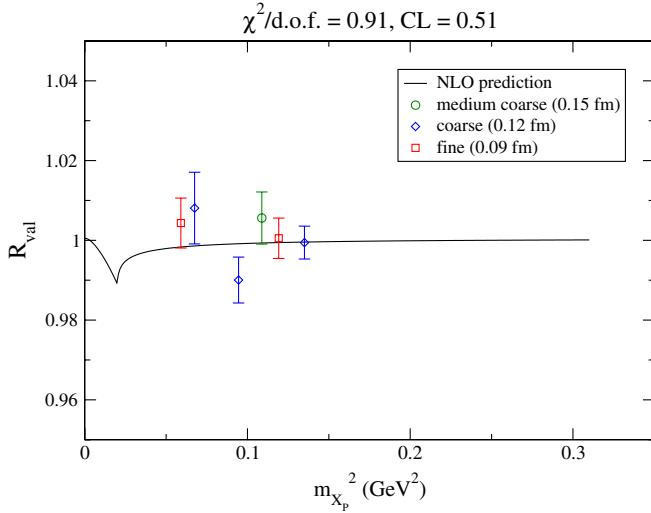


FIG. 5 (color online).  $\mathcal{R}_{\text{val}}$  ratio versus valence pion mass squared on all ensembles for the three different lattice spacings. The effects of staggering are negligible in the region where we have data. We include the term proportional to  $c_1$  in Eq. (41) in our fits used to obtain the central value for this quantity, as explained in Sec. V. (Since including a linear term proportional to  $c_1$  increases the statistical error in  $h_{A_1}$ , we take our central value and statistical error from this fit to be conservative.) This “partial NNLO” fit also has a good  $\chi^2/\text{d.o.f.} = 1.05$ , with a corresponding CL = 0.39. The constant linear term is small and consistent with zero [ $c_1 = -0.006(15)$ ]. Figure 6 shows the fit to  $\mathcal{R}_{\text{val}}$  versus  $m_{X_p}^2$  for all three lattice spacings using the rS $\chi$ PT formula, Eq. (41).

Although the data for  $\mathcal{R}_{\text{val}}$  is consistent with a constant, the cusp appearing close to the physical pion mass is a prediction of NLO  $\chi$ PT and has a physical origin, namely, the  $D$ - $\pi$  threshold, as we have remarked. Thus, any fits used to estimate systematic errors, even those that are somewhat *ad hoc*, such as those including higher-order polynomial terms, must include this cusp. Note that the cusp appears at the physical pion mass (in either SU(3) or SU(2)  $\chi$ PT), and is therefore in a region where  $\chi$ PT is

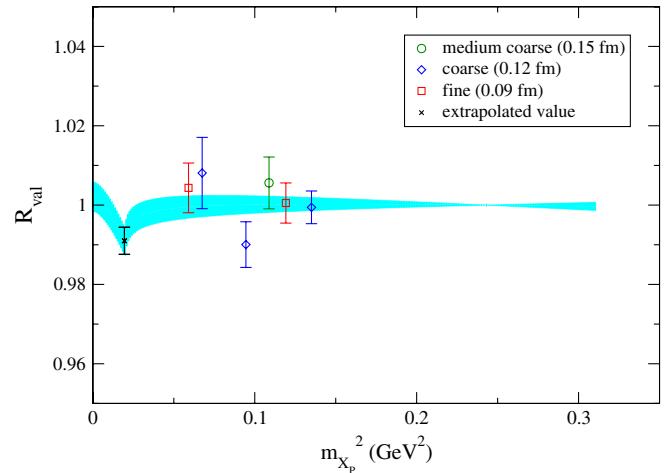


FIG. 6 (color online).  $\mathcal{R}_{\text{val}}$  ratio versus valence pion mass squared on all ensembles for the three different lattice spacings. The curve is the fit with 1 sigma error band to the ratio for all three lattice spacings using rS $\chi$ PT, extrapolated to the continuum by taking  $a \rightarrow 0$  in the NLO staggered chiral logarithms.

expected to be a reliable expansion. The cusp is a property of the function  $F(m, \Delta^{(c)}/m)$  given in Eq. (A2), and the position of the cusp as a function of  $m_{X_p}^2$  is determined by the  $D$ - $D^*$  splitting  $\Delta^{(c)}$  and the physical pion mass. We take these two quantities from experiment rather than from the lattice, since the experimental uncertainties are much smaller.

We find that with or without the NNLO analytic terms, the  $\chi$ PT (continuum or rooted staggered) describes the lattice data with  $\chi^2/\text{d.o.f.}$  close to 1 and correspondingly good confidence levels. We find a confidence level for the fit to  $\mathcal{R}_{\text{sea}}$  of 0.76 for the fit that includes NNLO analytic terms. The strictly NLO expression for the lattice ratio  $\mathcal{R}_{\text{sea}}$  has no free parameters, but it describes the data with a confidence level of 0.73. Similar fits to  $\mathcal{R}_{\text{val}}$  are described above and yield reasonable confidence levels for both types of fits. Since the lattice data do not distinguish between these model fit functions, and the fit using only the NNLO analytic terms is not systematic in the chiral expansion, we assign the difference between the two determinations, which is 0.9%, as the systematic error of leaving out higher-order terms when performing the chiral extrapolation. The final results for  $\mathcal{R}_{\text{sea}}$ ,  $\mathcal{R}_{\text{val}}$ , and  $\mathcal{R}_{\text{fid}}$  are given in Table VIII. The errors are statistical only; note that the strictly NLO values have no free parameters, and therefore no statistical errors. The final value of  $h_{A_1}$  still has statistical errors coming from the statistical errors in  $h_{A_1}^{\text{fid}}$ . The extrapolated results for  $\mathcal{R}_{\text{fid}}$  are consistent within the statistical errors of the NNLO fit. Again, we choose for our central value the result from the NNLO extrapolation with its larger errors to be conservative.

The cyan (gray) band in Fig. 6 is the continuum extrapolation with  $a \rightarrow 0$  in the rS $\chi$ PT formula. For this quantity,

TABLE VIII. Continuum extrapolated values of  $\mathcal{R}_{\text{sea}}$ ,  $\mathcal{R}_{\text{val}}$ ,  $\mathcal{R}_{\text{fid}}$ , and  $h_{A_1}(1)$  evaluated at the physical quark masses. The first column labels the quantity. The second is the computed value including NNLO analytic terms in the chiral fit. The third is the quantity evaluated in purely NLO  $\chi$ PT, and has no free parameters (once  $g_{DD^*}\pi$ ,  $f_\pi$  and  $\Delta^{(c)}$  are taken from phenomenology) in the chiral fit. The final row shows  $h_{A_1}(1)$ , which includes a statistical error coming from  $h_{A_1}^{\text{fid}}$ . The numbers are the same to the quoted precision using rS $\chi$ PT or continuum  $\chi$ PT.

	w/NNLO	Strictly NLO
$\mathcal{R}_{\text{sea}}$	1.0059(90)	0.9983
$\mathcal{R}_{\text{val}}$	0.9910(34)	0.9895
$\mathcal{R}_{\text{fid}}$	0.997(10)	0.9878
$h_{A_1}(1)$	0.921(13)	0.9124(84)

the staggered lattice artifacts affecting the chiral logarithms in  $h_{A_1}$  are negligible in the region where we have lattice data, which is due mainly to the small size of the chiral logarithms themselves. This is confirmed by the close agreement between the data points at each lattice spacing and the continuum curve. In fact, if we use continuum  $\chi$ PT to perform the chiral extrapolation, the result is unchanged. The primary difference between the rS $\chi$ PT expression and the continuum  $\chi$ PT expression is the reduction of the cusp near the physical pion mass in rS $\chi$ PT, though our lattice data are not near enough to the physical pion mass to demonstrate this effect.

Figure 7 shows the fit to  $\mathcal{R}_{\text{sea}}$ , extrapolated to the continuum and to the physical strange sea quark mass. Note that this ratio does not produce a cancellation of correlations between numerator and denominator and so has larger statistical errors than  $\mathcal{R}_{\text{val}}$ . Here again the discretization effects due to staggered logarithms are negligibly small. Since the effects of including staggered discretiza-

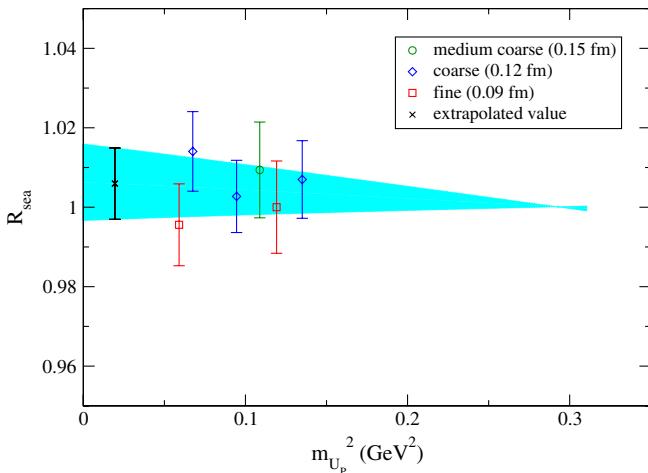


FIG. 7 (color online).  $\mathcal{R}_{\text{sea}}$  ratio versus  $m_{U_p}^2$  for all ensembles and lattice spacings. The curve is the fit to all of the lattice data, extrapolated to the continuum. The curve is also extrapolated to the physical strange sea quark mass.

tion effects in the chiral logarithms are negligible in the region where we have numerical data, and since the only nontrivial feature in the chiral extrapolation is the cusp near the physical pion mass, which we describe by continuum  $\chi$ PT (our extrapolated curve has  $a \rightarrow 0$  in the rS $\chi$ PT formula and thus reduces to the continuum form), we conclude that staggered taste-violating effects appearing in chiral logarithms are essentially removed in our ratio extrapolations.

Figure 8 shows all of the full QCD points on the three lattice spacings. The curve is the quantity

$$h_{A_1}^{\text{phys}}(\hat{m}') \approx h_{A_1}^{\text{fid}}(m_x^{\text{fid}}, \hat{m}'^{\text{fid}}, m_s^{\text{fid}}, a^{\text{fid}}) \\ \times [\mathcal{R}_{\text{sea}}(\hat{m}', m_s, 0) \times \mathcal{R}_{\text{val}}(\hat{m}', \hat{m}', m_s, 0)], \quad (43)$$

which again becomes an exact relation for the physical form factor when  $a^{\text{fid}} \rightarrow 0$ . The curve is thus the product of the two continuum extrapolated ratio fits shown in Figs. 6 and 7, times the fiducial point, which we take to be  $a\hat{m}^{\text{fid}} = 0.0124$  at the fine lattice spacing (the solid square in Fig. 8). Because this is a full QCD curve, the valence mass  $m_x$  equals the light sea mass  $\hat{m}'$ . The other full QCD points are shown as open symbols in Fig. 8 for comparison, though the fits were performed on the ratios and normalized by the fiducial point at  $a\hat{m}^{\text{fid}} = 0.0124$ . Note that the curve is already extrapolated in the strange sea quark mass, and so does not perfectly overlap with the  $a\hat{m}^{\text{fid}} = 0.0124$  point. As discussed above, when this quan-

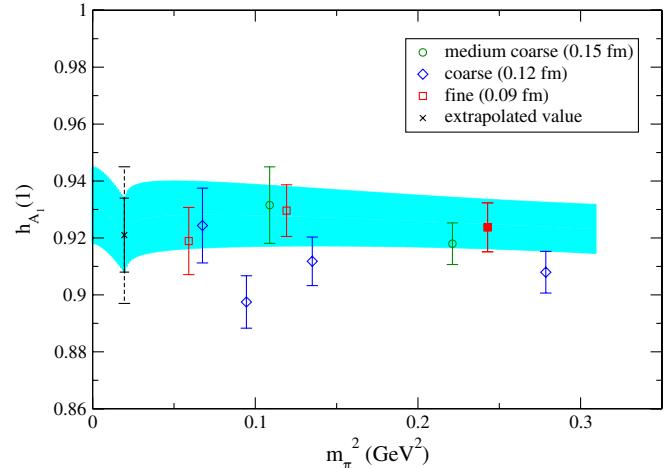


FIG. 8 (color online). The full QCD points versus  $m_\pi^2$  on the three lattice spacings are shown in comparison to the continuum curve. The curve is the product of the two continuum extrapolated ratio fits shown in Figs. 6 and 7, times the fiducial point, which we have chosen to be the  $\hat{m}' = 0.0124$  fine lattice point (the filled square). The curve is already extrapolated to the physical strange sea quark mass, and so does not perfectly overlap with the lattice data point at the fiducial value. The cross is the extrapolated value, where the solid line is the statistical error, and the dashed line is the total systematic error added to the statistical error in quadrature.

ity is evaluated at  $\hat{m}' = \hat{m}$  it yields the value of  $h_{A_1}$  at physical quark masses. The cross is the extrapolated value, where the solid line is the statistical error, and the dashed line is the total systematic error added to the statistical error in quadrature.

The low-energy constant  $g_{DD^*\pi}$  enters the chiral extrapolation formula and determines the size of the cusp near the physical pion mass. Our data do not constrain this constant, so we take a wide range for  $g_{DD^*\pi}$  that encompasses the range of values coming from phenomenology and lattice calculations: fits to a wide range of experimental data prior to the measurement of the  $D^*$  width by Stewart ( $g_{DD^*\pi} = 0.27^{+0.06}_{-0.03}$  [73]), an update of the Stewart analysis including the  $D^*$  width ( $g_{DD^*\pi} = 0.51$ ; no error quoted [74]), quark models ( $g_{DD^*\pi} \approx 0.38$  [75]), quenched lattice QCD ( $g_{DD^*\pi}^{N_f=0} = 0.67 \pm 0.08^{+0.04}_{-0.06}$  [76]), two flavor lattice QCD in the static limit ( $g_{\text{static}}^{N_f=2} = 0.516 \pm 0.051$  [77]), and the measurement of the  $D^*$  width ( $g_{DD^*\pi} = 0.59 \pm 0.07$  [78]). There are as of yet no 2 + 1 flavor lattice calculations of  $g_{DD^*\pi}$ . For this work we take  $g_{DD^*\pi} = 0.51 \pm 0.2$ , leading to a parametric uncertainty of 0.9% in  $h_{A_1}(1)$  that is included as a systematic error.

The additional low-energy constants that enter the chiral formulas are the tree-level continuum coefficients  $\mu_0$  and  $f$ , and the taste-violating parameters that vanish in the continuum. These are the taste splittings  $a^2 \Delta_\Xi$  with  $\Xi = P, A, T, V, I$ , and the taste-violating hairpin-coefficients  $a^2 \delta'_A$  and  $a^2 \delta'_V$ . We set  $f$  to the experimental value of the pion decay constant  $f_\pi = 0.1307$  GeV in the coefficient of the NLO logarithms. The pion masses used as inputs in the rS $\chi$ PT formulas are computed from the bare quark masses and converted into physical units using

$$m_{xy}^2 = (r_1/r_1^{\text{phys}})^2 \mu_{\text{tree}}(m_x + m_y), \quad (44)$$

where  $\mu_{\text{tree}}$  is obtained from fits to the light pseudoscalar mass squared to the tree-level form (in  $r_1$  units),  $r_1^2 \mu_{\text{tree}}(m_x + m_y)$ . This accounts for higher-order chiral corrections and is more accurate than using  $\mu$  obtained in the chiral limit, giving a better approximation to the pion mass squared at a given light-quark mass. Since the parameters in our lattice simulations at different lattice spacings are expressed in  $r_1$  units, we require the physical value of  $r_1$  to convert to physical units and take the physical pion mass and  $\Delta^{(c)}$  from experiment. Thus, the  $\approx 2.5\%$  uncertainty in  $r_1^{\text{phys}}$  gives a parametric error in the chiral extrapolation. Because the chiral extrapolation is so mild, however, this error turns out to be negligible compared with other systematic errors. Since we are taking the pion mass from experiment there is a negligible error due to the light-quark mass uncertainty in the chiral extrapolation. The strange sea quark mass enters the chiral extrapolation formulas, but the dependence is weak, and the error in the bare strange quark mass leads to a negligible parametric error in  $h_{A_1}$ . The taste splittings  $\Delta_\Xi$  have been determined

in Ref. [16], and their approximately 10% uncertainty also leads to a negligible error in  $h_{A_1}(1)$ . The taste-violating hairpin coefficients have much larger fractional uncertainties, but these too lead to a negligible uncertainty in  $h_{A_1}(1)$ . Even setting the rS $\chi$ PT parameters to zero does not change our result for  $h_{A_1}(1)$  significantly. As mentioned above, our result does not change if we use the continuum  $\chi$ PT formula in our chiral fits.

In the calculation of the form factor, the tadpole-improved coefficient  $c_{\text{SW}} = 1/u_0^3$  is obtained with  $u_0$  from the Landau link on the coarse lattices, but from the plaquette for  $u_0$  on the fine and coarser lattices. Though unintentional, there is nothing wrong with this, since it is not known *a priori* which provides the best estimate of the tadpole improvement factor. However, the  $u_0$  term for the spectator light (staggered) quark, which appears in the tadpole improvement of the Asqtad action, was taken from the Landau link on the coarse lattices, even though the sea quark sector used  $u_0$  from the plaquette. On the fine and coarser lattices,  $u_0$  was taken to be the same in the light valence and sea quark sectors. The estimates of  $u_0$  from plaquette versus Landau link differ only by 4% on the coarse lattices.

Although the effect of this mistuning is expected to be small (correcting  $u_0$  would lead to a slightly different valence propagator and different tuned  $\kappa$  values, thus leading to a small modification of the staggered chiral parameters in the valence sector for the coarse lattices used as inputs to the chiral fit), it is possible to study how much difference it makes using the  $h_{A_1}$  lattice data. Including all three lattice spacings and using our preferred chiral fit, we find  $h_{A_1}(1) = 0.921(13)$ , where the error here is statistical only. If we neglect the coarse data points, we find  $h_{A_1}(1) = 0.920(17)$ , almost unchanged except for a somewhat larger statistical error. We can also examine the ratios  $\mathcal{R}_{\text{val}}$  and  $\mathcal{R}_{\text{sea}}$ . In our preferred fit to all the lattice data these are 0.9910(34) and 1.0059(90), respectively, where the errors are again only statistical. If we drop the coarse lattice data, these become 0.9960(56) and 0.999(13), respectively. Since the ratio  $\mathcal{R}_{\text{sea}}$  has very little valence quark mass dependence, we can combine  $\mathcal{R}_{\text{sea}}$  from the fit to all of the lattice data with  $\mathcal{R}_{\text{val}}$  from the fit neglecting the coarse lattice data. This is useful, because  $\mathcal{R}_{\text{sea}}$  has the larger statistical error, so we would like to use the full lattice data set to determine this ratio, thus isolating the mistuning in the valence sector on the coarse lattices. When this is done we find that the central value of the final  $h_{A_1}(1)$  is shifted upward by 0.4%, well within statistical errors and smaller than our other systematic errors. We assign a systematic error of 0.4% due to the  $u_0$  mistuning.

## E. Finite volume effects

The finite volume corrections to the integrals which appear in heavy-light  $\chi$ PT formulas, including those for  $B \rightarrow D^*$  were given by Arndt and Lin [79]. There are no

new integrals appearing in the staggered case, and it is straightforward to use the results of Arndt and Lin in the rS $\chi$ PT for  $h_{A_1}(1)$ , as shown in Ref. [22]. We find that although the finite volume corrections in  $h_{A_1}(1)$  would be large near the cusp at the physical pion mass on the current MILC ensembles (ranging in size from 2.5–3.5 fm), for the less chiral data points at which we have actually simulated, the finite volume effects are negligible. For all data points in our simulations the finite volume corrections are less than 1 part in  $10^4$ . We therefore assign no error due to finite volume effects.

### F. Discretization errors

As shown in Refs. [28–30,49], the matching of lattice gauge theory to QCD is accomplished by normalizing the first few terms in the heavy-quark expansion. This is done by tuning the kinetic masses of the  $D_s$  and  $B_s$  mesons computed using the SW action (for the heavy quarks) to the experimental meson masses. Tree-level tadpole-improved perturbation theory is used to tune the coupling  $c_{\text{sw}}$  and the rotation coefficient  $d_1$  for the bottom and charm quarks. Once this matching is done, the discretization errors in  $h_{A_1}(1)$  are of order  $\alpha_s(\bar{\Lambda}/2m_Q)^2$  and  $(\bar{\Lambda}/2m_Q)^3$  [28], where the powers of two are combinatoric factors. The leading matching uncertainty is of the order  $\alpha_s(\bar{\Lambda}/2m_c)^2$ . We estimate the size of this error setting  $\alpha_s = 0.3$ ,  $\bar{\Lambda} = 500$  MeV, and  $m_c = 1.2$  GeV, which gives  $\alpha_s(\bar{\Lambda}/2m_c)^2 = 0.013$ .

Since we have numerical data at three lattice spacings we are able to study how well the power-counting estimate accounts for observed discretization effects. Making use of Eq. (43), but varying the fiducial lattice spacing from our lightest to coarsest lattices, we are able to obtain  $h_{A_1}(1)$  at physical quark masses, with discretization effects associated with the staggered chiral logarithms removed in the ratios appearing in Eq. (43). The discretization effects that remain are: taste violations in  $h_{A_1}^{\text{fid}}$ , taste violations at higher order than NLO in the ratios, the effect of the analytic term coming from light-quark discretization effects (proportional to  $\alpha_s a^2$ ), and the heavy-quark discretization effects. The taste violations in  $h_{A_1}^{\text{fid}}$  and the taste violations in the ratios appearing at higher order than NLO have been shown to be negligible. We now consider the remaining

TABLE IX.  $h_{A_1}(1)$  at physical quark masses at different lattice spacings, where taste-violating effects have been removed, or shown to be negligible. Discretization effects due to analytic terms associated with the light-quark sector and heavy-quark discretization effects remain in the lattice data.

$a$ (fm)	$h_{A_1}(1)$
0.15	0.914(11)
0.12	0.907(14)
0.09	0.921(13)

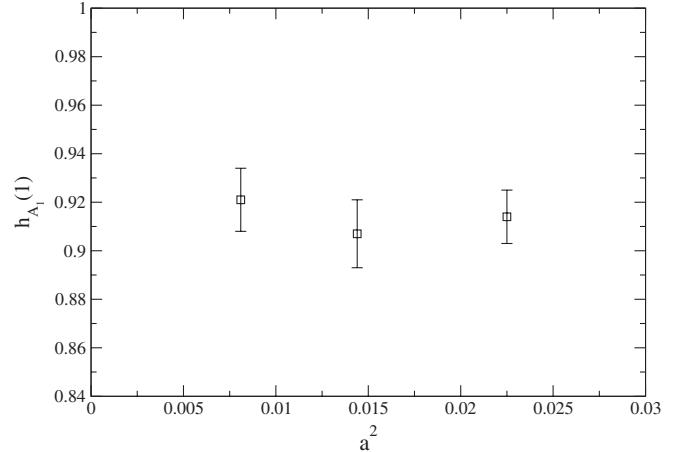


FIG. 9.  $h_{A_1}(1)$  at physical quark masses versus  $a^2$  ( $\text{fm}^2$ ), where taste-violating effects have been removed, or shown to be negligible. Discretization effects due to analytic terms associated with the light-quark sector and heavy-quark discretization effects remain in the lattice data.

discretization errors coming from the light-quark analytic term and the heavy-quark discretization effects. Table IX presents the results for  $h_{A_1}(1)$  as obtained from Eq. (43), and Fig. 9 shows them plotted as a function of lattice spacing squared. Although the Fermilab action and currents possess a smooth continuum limit, the MILC ensembles are not yet at small enough  $a$  to obtain simply  $O(a)$  or  $O(a^2)$  behavior. The spread of the lattice data points gives some indication of the size of the remaining discretization effects, however, and we find that the fine (0.09 fm) lattice data point and the coarse (0.12 fm) lattice data point differ by 1.5%. This is similar to our power-counting estimate, and we assign the larger of the two, 1.5%, as the systematic error due to residual discretization effects.

### G. Summary

Our final result, given the error budget in Table X, is

$$h_{A_1} = 0.921(13)(8)(8)(14)(6)(3)(4), \quad (45)$$

TABLE X. Final error budget for  $h_{A_1}(1)$  where each error is discussed in the text. Systematic errors are added in quadrature and combined in quadrature with the statistical error to obtain the total error.

Uncertainty	$h_{A_1}(1)$
Statistics	1.4%
$g_{DD^*\pi}$	0.9%
NLO vs NNLO $\chi$ PT fits	0.9%
Discretization errors	1.5%
Kappa tuning	0.7%
Perturbation theory	0.3%
$u_0$ tuning	0.4%
Total	2.6%

where the errors are statistical, parametric uncertainty in  $g_{DD^*\pi}$ , chiral extrapolation errors, discretization errors, parametric uncertainty in heavy-quark masses (kappa tuning), perturbative matching, and the  $u_0$  (mis)tuning on the coarse lattices. Adding all systematic errors in quadrature, we obtain

$$h_{A_1}(1) = 0.921(13)(20). \quad (46)$$

This final result differs slightly from that presented at Lattice 2007 [80], where a preliminary  $h_{A_1}(1) = 0.924(12)(19)$  was quoted. There are three main changes in the analysis from the preliminary result: our earlier result used a value of  $\alpha_s$  in the perturbative matching evaluated at the scale  $2/a$ , while the present result uses the HLM [64] prescription to fix the scale. This causes a change of 0.1%, well within the estimated systematic error due to the perturbative matching. In the previous result, the fine lattice data was blocked by 4 in the jackknife procedure; we now block by 8 to fully account for autocorrelation errors. This does not change the central value, but increases the statistical error slightly. Finally, we have chosen a value for  $g_{DD^*\pi} = 0.51 \pm 0.2$  instead of  $g_{DD^*\pi} = 0.45 \pm 0.15$  to be more consistent with the range of values quoted in the literature. This causes a decrease in  $h_{A_1}(1)$  of 0.2%.

### VIII. CONCLUSIONS

We have introduced a new method to calculate the zero-recoil form factor for the  $B \rightarrow D^* \ell \bar{\nu}$  decay. We include 2 + 1 flavors of sea quarks in the generation of the gauge ensembles, so the calculation is completely unquenched. We have introduced a new double ratio, which gives the form factor directly, and leads to a large savings in the computational cost. The simulation is performed in a regime where we expect rooted staggered chiral perturbation theory to apply; we therefore use the rS $\chi$ PT result for the  $B \rightarrow D^*$  form factor [22] to perform the chiral extrapolation and to remove taste-breaking effects. To aid the chiral and continuum extrapolations, we introduced a set of ratios that has allowed us to largely disentangle light and heavy-quark discretization effects. Our new result  $\mathcal{F}(1) = h_{A_1}(1) = 0.921(13)(20)$  is consistent with the previous quenched result  $\mathcal{F}(1) = 0.913^{+0.029}_{-0.034}$  [13], but our errors are both smaller and under better theoretical control. This result allows us to extract  $|V_{cb}|$  from the experimental measurement of the  $B \rightarrow D^* \ell \bar{\nu}$  form factor, which determines  $\mathcal{F}(1)|V_{cb}|$ . After applying a 0.7% electromagnetic correction to our value for  $\mathcal{F}(1)$  [81], and taking the most recent PDG average for  $|V_{cb}|$   $\mathcal{F}(1) = (35.9 \pm 0.8) \times 10^{-3}$  [82], we find

$$|V_{cb}| = (38.7 \pm 0.9_{\text{exp}} \pm 1.0_{\text{theo}}) \times 10^{-3}. \quad (47)$$

This differs by about  $2\sigma$  from the inclusive determination  $|V_{cb}| = (41.6 \pm 0.6) \times 10^{-3}$  [82]. Our new value supersedes the previous Fermilab quenched number [13], as it should other quenched numbers such as that in Ref. [83].<sup>3</sup>

Our largest error in  $\mathcal{F}(1)$  is the systematic error due to heavy-quark discretization effects, which we have estimated using HQET power counting and inspection of the numerical data at three lattice spacings. This error can be reduced by going to finer lattice spacings, or by using an improved Fermilab action [70]. When using this improved action, it would be necessary to improve the currents to the same order. We have introduced a method for separating the heavy and light-quark discretization errors, where the physical  $h_{A_1}$  can be factorized into two factors  $h_{A_1}^{\text{fid}} \times \mathcal{R}_{\text{fid}}$ , such that the heavy-quark discretization errors are largely isolated in  $h_{A_1}^{\text{fid}}$ . Combining our value of  $\mathcal{R}_{\text{fid}} = 0.997(10)(13)$  (where the first error is statistical, and the second is due to systematics that do not cancel in the ratio) with a determination of  $h_{A_1}^{\text{fid}}$  at finer lattice spacings and/or with an improved action would be a cost-effective way of reducing the heavy-quark discretization errors. The next largest error in our calculation of  $\mathcal{F}(1)$  is statistical, and this error drives many of the systematic errors. This is mostly a matter of computing. It would also be desirable to perform the matching of the heavy-quark current to higher order in perturbation theory, or by using nonperturbative matching. With these improvements, it would be possible to bring the error in  $\mathcal{F}(1)$  to or below 1%, allowing a very precise determination of  $|V_{cb}|$  from exclusive semileptonic decays.

### ACKNOWLEDGMENTS

We thank J. Bailey for a careful reading of the manuscript. Computations for this work were carried out in part on facilities of the USQCD Collaboration, which are funded by the Office of Science of the U.S. Department of Energy; and on facilities of the NSF Teragrid under allocation Contract No. TG-MCA93S002. This work was supported in part by the United States Department of Energy under Grant Nos. DE-FC02-06ER41446 (C.D., L.L.), DE-FG02-91ER40661 (S.G.), DE-FG02-91ER40677 (A.X.K.), DE-FG02-91ER40628 (C.B., J.L.), DE-FG02-04ER41298 (D.T.), and by the National Science Foundation under Grant Nos. PHY-0555243, PHY-0757333, PHY-0703296 (C.D., L.L.), PHY-0555235 (J.L.), and PHY-0456556 (R.S.). R.T.E. and E.G. thank Fermilab and URA for their hospitality. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

<sup>3</sup>Ref. [83] calculates the  $B \rightarrow D^* \ell \bar{\nu}$  form factor in the quenched approximation at zero and nonzero recoil momentum and uses a step-scaling method [84] to control the heavy-quark discretization errors.

## APPENDIX: CHIRAL PERTURBATION THEORY

Eq. (34) of Ref. [22] gives the expression needed for  $h_{A_1}(1)$  in partially quenched  $\chi$ PT with degenerate up- and down-quark masses (the  $2 + 1$  case) in the rooted staggered theory:

$$\begin{aligned} h_{A_1}^{(B_x)PQ,2+1}(1)/\eta_A = 1 + X_A(\Lambda_\chi) + \frac{g_{DD^*}^2 \pi}{48 \pi^2 f^2} & \left\{ \frac{1}{16} \sum_{\substack{j=x_u,x_d,x_s \\ \Xi=I,P,4V,4A,6T}} \bar{F}_{j\Xi} + \frac{1}{3} \left[ R_{X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \left( \frac{d\bar{F}_{X_I}}{dm_{X_I}^2} \right) \right. \right. \\ & - \sum_{j \in \{M_I^{(5)}\}} D_{j,X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \bar{F}_j \Big] + a^2 \delta'_V \left[ R_{X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) \left( \frac{d\bar{F}_{X_V}}{dm_{X_V}^2} \right) \right. \\ & \left. \left. - \sum_{j \in \{M_V^{(7)}\}} D_{j,X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) \bar{F}_j \right] \right. \\ & \left. + (V \rightarrow A) \right\}, \end{aligned} \quad (\text{A1})$$

where

$$\begin{aligned} F(m_j, z_j) &= \frac{m_j^2}{z_j} \left\{ z_j^3 \ln \frac{m_j^2}{\Lambda_\chi^2} + \frac{1}{3} z_j^3 - 4z_j + 2\pi - \sqrt{z_j^2 - 1} (z_j^2 + 2) (\ln[1 - 2z_j(z_j - \sqrt{z_j^2 - 1})] - i\pi) \right\} \\ &\rightarrow (\Delta^{(c)})^2 \ln \left( \frac{m_j^2}{\Lambda_\chi^2} \right) + \mathcal{O}[(\Delta^{(c)})^3], \end{aligned} \quad (\text{A2})$$

with  $\bar{F}(m_j, z_j) = F(m_j, -z_j)$ , and  $z_j = \Delta^{(c)}/m_j$ , where  $\Delta^{(c)}$  is the  $D$ - $D^*$  mass splitting. The residues  $R_j^{[n,k]}$  and  $D_{j,i}^{[n,k]}$  are defined in Refs. [18,19], and for completeness we quote them here:

$$R_j^{[n,k]}(\{M\}, \{\mu\}) \equiv \frac{\prod_{a=1}^k (\mu_a^2 - m_j^2)}{\prod_{i \neq j} (m_i^2 - m_j^2)}, \quad D_{j,l}^{[n,k]}(\{M\}, \{\mu\}) \equiv -\frac{d}{dm_l^2} R_j^{[n,k]}(\{M\}, \{\mu\}). \quad (\text{A3})$$

These residues are a function of two sets of masses, the numerator masses,  $\{M\} = \{m_1, m_2, \dots, m_n\}$  and the denominator masses,  $\{\mu\} = \{\mu_1, \mu_2, \dots, \mu_k\}$ . In our  $2 + 1$  flavor case, we have

$$\{M_X^{(5)}\} \equiv \{m_\eta, m_X\}, \quad \{M_X^{(7)}\} \equiv \{m_\eta, m_{\eta'}, m_X\}, \quad \{\mu\} \equiv \{m_U, m_S\}. \quad (\text{A4})$$

The masses  $m_{\eta_I}$ ,  $m_{\eta_V}$ ,  $m_{\eta'_V}$  are given by [18]

$$\begin{aligned} m_{\eta_I}^2 &= \frac{m_{U_I}^2}{3} + \frac{2m_{S_I}^2}{3}, \quad m_{\eta_V}^2 = \frac{1}{2} \left( m_{U_V}^2 + m_{S_V}^2 + \frac{3}{4} a^2 \delta'_V - Z \right), \quad m_{\eta'_V}^2 = \frac{1}{2} \left( m_{U_V}^2 + m_{S_V}^2 + \frac{3}{4} a^2 \delta'_V + Z \right), \\ Z &\equiv \sqrt{(m_{S_V}^2 - m_{U_V}^2)^2 - \frac{a^2 \delta'_V}{2} (m_{S_V}^2 - m_{U_V}^2) + \frac{9(a^2 \delta'_V)^2}{16}}. \end{aligned} \quad (\text{A5})$$

The ratio  $R_{\text{val}}^{\text{NLO}}$  in the continuum through NLO in  $\chi$ PT is

$$\begin{aligned} \mathcal{R}_{\text{val}}^{\text{NLO}} = 1 + \frac{g_{DD^*}^2 \pi}{48 \pi^2 f^2} & \left\{ \sum_{j=u,d,s} \bar{F}_{xj} + \frac{1}{3} \left[ R_X^{[2,2]}(\{M_X^{(5)}\}; \{\mu\}) \left( \frac{d\bar{F}_X}{dm_X^2} \right) \right. \right. \\ & - \sum_{j=u,d,s} \bar{F}_{x'j} - \frac{1}{3} \left[ R_{X'}^{[2,2]}(\{M_{X'}^{(5)}\}; \{\mu\}) \left( \frac{d\bar{F}_{X'}}{dm_{X'}^2} \right) \right. \\ & \left. \left. - \sum_{j \in \{M_{X'}^{(5)}\}} D_{j,X'}^{[2,2]}(\{M_{X'}^{(5)}\}; \{\mu\}) \bar{F}_j \right] \right\}, \end{aligned} \quad (\text{A6})$$

where

$$\{M_{X'}^{(5)}\} \equiv \{m_{\eta}, m_{X'}\}, \quad (\text{A7})$$

and where  $m_{X'}$  is a valence pion made of two quarks set to the fiducial valence quark mass, and the subscript  $x'$  refers to a valence quark at the fiducial mass. This ratio is one by construction when the valence quark mass equals the fiducial valence quark mass.

- [1] E. Barberio *et al.* (Heavy Flavor Averaging Group (HFAG)), arXiv:0704.3575.
- [2] E. Gamiz *et al.* (HPQCD Collaboration), Phys. Rev. D **73**, 114502 (2006).
- [3] T. Bae, J. Kim, and W. Lee, Proc. Sci., LAT2005 (2006) 335 [arXiv:hep-lat/0510008].
- [4] D. J. Antonio *et al.* (RBC Collaboration), Phys. Rev. Lett. **100**, 032001 (2008).
- [5] C. Aubin, J. Laiho, and R. S. Van de Water, Proc. Sci., LAT2007 (2007) 375 [arXiv:0710.1121].
- [6] J. Chay, H. Georgi, and B. Grinstein, Phys. Lett. B **247**, 399 (1990).
- [7] I. I. Y. Bigi, N. G. Uraltsev, and A. I. Vainshtein, Phys. Lett. B **293**, 430 (1992).
- [8] I. I. Y. Bigi, B. Blok, M. A. Shifman, N. G. Uraltsev, and A. I. Vainshtein, arXiv:hep-ph/9212227.
- [9] I. I. Y. Bigi, M. A. Shifman, N. G. Uraltsev, and A. I. Vainshtein, Phys. Rev. Lett. **71**, 496 (1993).
- [10] I. I. Y. Bigi, M. A. Shifman, and N. Uraltsev, Annu. Rev. Nucl. Part. Sci. **47**, 591 (1997).
- [11] O. Büchmuller and H. Flächer, Phys. Rev. D **73**, 073008 (2006).
- [12] C. W. Bauer, Z. Ligeti, M. Luke, A. V. Manohar, and M. Trott, Phys. Rev. D **70**, 094017 (2004).
- [13] S. Hashimoto, A. S. Kronfeld, P. B. Mackenzie, S. M. Ryan, and J. N. Simone, Phys. Rev. D **66**, 014503 (2002).
- [14] C. W. Bernard *et al.*, Phys. Rev. D **64**, 054506 (2001).
- [15] C. Aubin *et al.*, Phys. Rev. D **70**, 094505 (2004).
- [16] C. Aubin *et al.* (MILC Collaboration), Phys. Rev. D **70**, 114501 (2004).
- [17] W.-J. Lee and S. R. Sharpe, Phys. Rev. D **60**, 114503 (1999).
- [18] C. Aubin and C. Bernard, Phys. Rev. D **68**, 034014 (2003).
- [19] C. Aubin and C. Bernard, Phys. Rev. D **68**, 074011 (2003).
- [20] S. R. Sharpe and R. S. Van de Water, Phys. Rev. D **71**, 114505 (2005).
- [21] C. Aubin and C. Bernard, Nucl. Phys. B, Proc. Suppl. **140**, 491 (2005).
- [22] J. Laiho and R. S. Van de Water, Phys. Rev. D **73**, 054501 (2006).
- [23] A. F. Falk and M. Neubert, Phys. Rev. D **47**, 2965 (1993).
- [24] T. Mannel, Phys. Rev. D **50**, 428 (1994).
- [25] A. Czarnecki, Phys. Rev. Lett. **76**, 4124 (1996).
- [26] A. Czarnecki and K. Melnikov, Nucl. Phys. **B505**, 65 (1997).
- [27] M. E. Luke, Phys. Lett. B **252**, 447 (1990).
- [28] A. S. Kronfeld, Phys. Rev. D **62**, 014505 (2000).
- [29] J. Harada *et al.*, Phys. Rev. D **65**, 094513 (2002).
- [30] J. Harada, S. Hashimoto, A. S. Kronfeld, and T. Onogi, Phys. Rev. D **65**, 094514 (2002).
- [31] S. Prelovsek, Phys. Rev. D **73**, 014506 (2006).
- [32] C. W. Bernard, C. E. DeTar, Z. Fu, and S. Prelovsek, Proc. Sci., LAT2006 (2006) 173.
- [33] C. Bernard, Phys. Rev. D **73**, 114503 (2006).
- [34] C. Bernard, C. E. Detar, Z. Fu, and S. Prelovsek, Phys. Rev. D **76**, 094504 (2007).
- [35] C. Aubin, J. Laiho, and R. S. Van de Water, Phys. Rev. D **77**, 114501 (2008).
- [36] C. Bernard, M. Golterman, and Y. Shamir, Phys. Rev. D **73**, 114511 (2006).
- [37] Y. Shamir, Phys. Rev. D **75**, 054503 (2007).
- [38] Y. Shamir, Phys. Rev. D **71**, 034509 (2005).
- [39] S. Dürr, Proc. Sci., LAT2005 (2005) 021 [arXiv:hep-lat/0509026].
- [40] S. R. Sharpe, Proc. Sci., LAT2006 (2006) 022 [arXiv:hep-lat/0610094].
- [41] A. S. Kronfeld, Proc. Sci., LAT2007 (2007) 016 [arXiv:0711.0699].
- [42] T. Blum *et al.*, Phys. Rev. D **55**, R1133 (1997).
- [43] K. Orginos and D. Toussaint (MILC Collaboration), Phys. Rev. D **59**, 014501 (1998).
- [44] J. F. Lagae and D. K. Sinclair, Phys. Rev. D **59**, 014511 (1998).
- [45] G. P. Lepage, Phys. Rev. D **59**, 074502 (1999).
- [46] K. Orginos, D. Toussaint, and R. L. Sugar (MILC Collaboration), Phys. Rev. D **60**, 054503 (1999).
- [47] C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **61**, 111502 (2000).
- [48] B. Sheikholeslami and R. Wohlert, Nucl. Phys. **B259**, 572 (1985).
- [49] A. X. El-Khadra, A. S. Kronfeld, and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
- [50] G. P. Lepage and P. B. Mackenzie, Phys. Rev. D **48**, 2250 (1993).
- [51] R. Sommer, Nucl. Phys. **B411**, 839 (1994).
- [52] C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **62**, 034503 (2000).
- [53] A. Gray *et al.* (HPQCD Collaboration), Phys. Rev. Lett. **95**, 212001 (2005).
- [54] C. Bernard (MILC Collaboration), Proc. Sci., LAT2007 (2007) 090 [arXiv:0710.1118].
- [55] C. Bernard *et al.* (MILC Collaboration), Proc. Sci., LAT2005 (2006) 025 [arXiv:hep-lat/0509137].
- [56] M. Wingate, J. Shigemitsu, C. T. Davies, G. P. Lepage, and H. D. Trottier, Phys. Rev. D **67**, 054505 (2003).
- [57] M. Okamoto *et al.*, Nucl. Phys. B, Proc. Suppl. **140**, 461 (2005).
- [58] C. Aubin *et al.* (Fermilab Lattice), Phys. Rev. Lett. **94**, 011601 (2005).
- [59] C. Aubin *et al.*, Phys. Rev. Lett. **95**, 122002 (2005).
- [60] E. Dalgic *et al.*, Phys. Rev. D **73**, 074502 (2006).
- [61] R. T. Evans, A. X. El-Khadra, and M. Di Pierro (Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2006 (2006) 081.
- [62] S. J. Brodsky, G. P. Lepage, and P. B. Mackenzie, Phys. Rev. D **28**, 228 (1983).
- [63] Q. Mason *et al.* (HPQCD Collaboration), Phys. Rev. Lett. **95**, 052002 (2005).
- [64] K. Hornbostel, G. P. Lepage, and C. Morningstar, Phys. Rev. D **67**, 034023 (2003).
- [65] A. X. El-Khadra, E. Gamiz, A. S. Kronfeld, and M. A. Nobes, Proc. Sci., LAT2007 (2007) 242 [arXiv:0710.1437].
- [66] A. X. El-Khadra, E. Gamiz, A. S. Kronfeld, and M. A. Nobes (unpublished).
- [67] G. P. Lepage, J. Comput. Phys. **27**, 192 (1978).
- [68] C. Aubin and C. Bernard, Phys. Rev. D **73**, 014515 (2006).
- [69] C. Bernard, M. Golterman, and Y. Shamir, Phys. Rev. D **77**, 074505 (2008).
- [70] M. B. Oktay and A. S. Kronfeld, Phys. Rev. D **78**, 014504 (2008).

- [71] E. Follana *et al.* (HPQCD Collaboration), Phys. Rev. D **75**, 054502 (2007).
- [72] A. S. Kronfeld, Nucl. Phys. B, Proc. Suppl. **53**, 401 (1997).
- [73] I. W. Stewart, Nucl. Phys. **B529**, 62 (1998).
- [74] M. C. Arnesen, B. Grinstein, I. Z. Rothstein, and I. W. Stewart, Phys. Rev. Lett. **95**, 071802 (2005).
- [75] R. Casalbuoni *et al.*, Phys. Rep. **281**, 145 (1997).
- [76] A. Abada *et al.*, Nucl. Phys. B, Proc. Suppl. **119**, 641 (2003).
- [77] H. Ohki, H. Matsufuru, and T. Onogi, Phys. Rev. D **77**, 094509 (2008).
- [78] A. Anastassov *et al.* (CLEO Collaboration), Phys. Rev. D **65**, 032003 (2002).
- [79] D. Arndt and C. J. D. Lin, Phys. Rev. D **70**, 014503 (2004).
- [80] J. Laiho (Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2007 (2007) 358 [arXiv:0710.1111].
- [81] A. Sirlin, Nucl. Phys. **B196**, 83 (1982).
- [82] C. Amsler *et al.* (Particle Data Group), Phys. Lett. B **667**, 1 (2008).
- [83] G. M. de Divitiis, R. Petronzio, and N. Tantalo, Nucl. Phys. **B807**, 373 (2009).
- [84] M. Guagnelli, F. Palombi, R. Petronzio, and N. Tantalo, Phys. Lett. B **546**, 237 (2002).

# The $D_s$ and $D^+$ Leptonic Decay Constants from Lattice QCD

Fermilab Lattice and MILC Collaborations

A. Bazavov<sup>a</sup>, C. Bernard<sup>b\*</sup>, C. DeTar<sup>c</sup>, E.D. Freeland<sup>b</sup>, E. Gamiz<sup>d,e</sup>, Steven Gottlieb<sup>f,g</sup>, U.M. Heller<sup>h</sup>, J.E. Hetrick<sup>i</sup>, A.X. El-Khadra<sup>d</sup>, A.S. Kronfeld<sup>e</sup>, J. Laiho<sup>b</sup>, L. Levkova<sup>c</sup>, P.B. Mackenzie<sup>e</sup>, M.B. Oktay<sup>c</sup>, M. Di Pierro<sup>j</sup>, J.N. Simone<sup>te‡</sup>, R. Sugar<sup>k</sup>, D. Toussaint<sup>a</sup>, and R.S. Van de Water<sup>l</sup>

<sup>a</sup>Department of Physics, University of Arizona, Tucson, Arizona, USA

<sup>b</sup>Department of Physics, Washington University, St. Louis, Missouri, USA

<sup>c</sup>Physics Department, University of Utah, Salt Lake City, Utah, USA

<sup>d</sup>Physics Department, University of Illinois, Urbana, Illinois, USA

<sup>e</sup>Fermi National Accelerator Laboratory, Batavia, Illinois, USA

<sup>f</sup>Department of Physics, Indiana University, Bloomington, Indiana, USA

<sup>g</sup>National Center for Supercomputing Applications, University of Illinois, Urbana, Illinois, USA

<sup>h</sup>American Physical Society, One Research Road, Box 9000, Ridge, New York, USA

<sup>i</sup>Physics Department, University of the Pacific, Stockton, California, USA

<sup>j</sup>School of Computer Sci., Telecom. and Info. Systems, DePaul University, Chicago, Illinois, USA

<sup>k</sup>Department of Physics, University of California, Santa Barbara, California, USA

<sup>l</sup>Physics Department, Brookhaven National Laboratory, Upton New York, USA

PoS(LAT2009)249

We present the leptonic decay constants  $f_{D_s}$  and  $f_{D^+}$  computed on the MILC collaboration's 2 + 1 flavor asqtad gauge ensembles. We use clover heavy quarks with the Fermilab interpretation and improved staggered light quarks. The simultaneous chiral and continuum extrapolation, which determines both decay constants, includes partially-quenched lattice results at lattice spacings  $a \approx 0.09, 0.12$  and  $0.15$  fm. We have made several recent improvements in our analysis: a) we include terms in the fit describing leading order heavy-quark discretization effects, b) we have adopted a more precise input  $r_1$  value consistent with our other  $D$  and  $B$  meson studies, c) we have retuned the input bare charm masses based upon the new  $r_1$ . Our preliminary results are  $f_{D_s} = 260 \pm 10$  MeV and  $f_{D^+} = 217 \pm 10$  MeV.

*The XXVII International Symposium on Lattice Field Theory - LAT2009*

*July 26-31 2009*

*Peking University, Beijing, China*

---

\*cb@lump.wustl.edu

†Speaker.

‡simone@fnal.gov

## 1. Introduction

We report on progress in the Fermilab Lattice and MILC Collaboration calculation of the  $D$  meson decay constants. This work is a continuation of the program that predicted the decay constants:  $f_{D^+} = 201(3)(17)$  and  $f_{D_s} = 249(3)(16)$  MeV [1], in good agreement with the CLEO-c value of  $f_{D^+} = 205.8 \pm 8.5 \pm 2.5$  [2, 3]. We have since extended this calculation to two additional ensembles at our finest lattice spacing  $a \approx 0.09$  fm and we have replaced a limited set of very coarse ( $a \approx 0.18$  fm) ensembles with higher statistic ensembles at a somewhat finer spacing  $a \approx 0.15$  fm. In our last update [4] we reported:  $f_{D^+} = 207(11)$  and  $f_{D_s} = 249(11)$ , where  $f_{D_s}$  is about  $0.6\sigma$  lower than the recent experimental average [5]. The value of  $f_{D_s}$  remains an pressing issue given that experimental average is about  $2.1\sigma$  higher than the most precise lattice result from the HPQCD collaboration [6]. The apparent tension between experiment and lattice predictions has motivated suggestions of physics beyond the Standard Model [7].

Smaller statistical uncertainties and better control of systematic effects are key to resolving the  $f_{D_s}$  puzzle. In this report, we have doubled statistics on the most chiral of the  $a \approx 0.09$  fm lattices; otherwise, statistics have not changed. A new generation of calculations, now underway, aims to increase statistics by a factor of four overall. Our progress includes: a) a better method of accounting for heavy-quark discretization effects b) a more precise input value for the scale parameter  $r_1$ , consistent with our other heavy quark studies and c) more precisely tuned input charm kappa values.

## 2. Staggered chiral perturbation theory for heavy-light mesons

We use the asqtad improved staggered action for both sea and light valence quarks. Leading discretization effects split the light pseudoscalar meson masses,

$$M_{ab,\xi}^2 = (m_a + m_b)\mu + a^2 \Delta_\xi , \quad (2.1)$$

where there are sixteen tastes in representations  $\xi = P, A, T, V, I$ .

Staggered chiral perturbation theory for heavy-light mesons accounts for such taste breaking effects [8]. At NLO in the chiral expansion, for  $2 + 1$  flavors, and at leading order in the heavy quark expansion,

$$\phi_{H_q} = \Phi_0 [1 + \Delta f_H(m_q, m_l, m_h) + p_H(m_q, m_l, m_h) + c_L \alpha_V^2 a^2] , \quad (2.2)$$

where  $\phi_{H_q} = f_{H_q} \sqrt{m_{H_q}}$  and  $f_{H_q}$  is the decay constant of a heavy meson  $H_q$  consisting of a heavy quark and a light quark of mass  $m_q$ . The heavier sea quark has mass  $m_h$  and the two degenerate light sea quarks have mass  $m_l$ . The  $\phi_{H_q}$ , in general, are partially quenched:  $m_q \neq m_l$  and  $m_q \neq m_h$ . The chiral logarithm terms,  $\Delta f_H$ , are  $a$  dependent as a consequence of the mass splittings in Eq. (2.1) as well as from ‘‘hairpin’’ terms proportional to the low energy constants  $a^2 \delta'_A$  and  $a^2 \delta'_V$ . The  $a$  dependence of the analytic terms,  $p_H$ , ensures that  $\phi_{H_q}$  is unchanged by a change in the chiral scale,  $\Lambda_\chi$ , of the logarithms. The expression in Eq. (2.2) is used in our combined chiral and continuum extrapolations. In practice, we add the NNLO analytic terms to the fit function in order to extend the fit up to  $m_q \sim m_s$  and extract  $f_{D_s}$ . Priors for the parameters  $a^2 \delta'_A$  and  $a^2 \delta'_V$  as well as values of the physical light quark masses are obtained from the MILC analysis of  $f_\pi$  and  $f_K$  [9].

### 3. Discretization effects from clover heavy quarks

We use tadpole-improved clover charm quarks. At leading order, discretization errors are a combination of  $\mathcal{O}(a^2\Lambda_{HQ}^2)$  and  $\mathcal{O}(\alpha a \Lambda_{HQ})$  effects where  $\alpha$  is the QCD coupling and  $\Lambda_{HQ}$  is the scale in the heavy quark expansion. Our past studies have estimated heavy quark discretization effects using such power counting arguments to bound the error at the smallest lattice spacing, taking  $\Lambda_{HQ} \approx 700$  MeV. This rather crude method does not effectively use the data to guide the error estimate.

This study introduces a new procedure: the leading order heavy quark discretization errors are modeled to leading order as part of the combined chiral and continuum extrapolation. At tree-level, discretization effects arise from both the quark action and the (improved) current [10, 11, 12]. We add five extra terms to Equation 2.2:

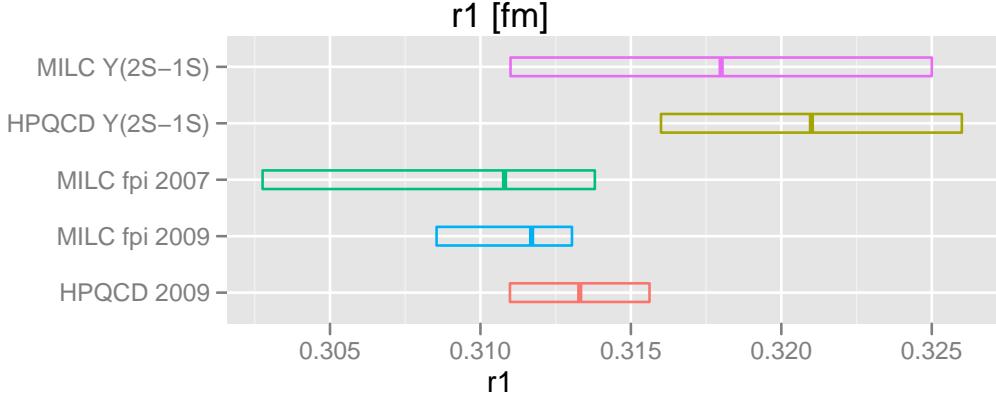
$$\Phi_0 [a^2 \Lambda_{HQ}^2 \{c_E f_E(am_Q) + c_X f_X(am_Q) + c_Y f_Y(am_Q)\} + \alpha_V a \Lambda_{HQ} \{c_B f_B(am_Q) + c_3 f_3(am_Q)\}] \quad (3.1)$$

The coefficients  $c_E$ ,  $c_X$ ,  $c_Y$ ,  $c_B$  and  $c_3$  are additional parameters determined in the fit while the  $f_i$  are (smooth) functions of the heavy quark mass,  $am_Q$ , known at tree level. We introduce priors for the coefficients constraining them to be  $\mathcal{O}(1)$  while setting  $\Lambda_{HQ} = 700$  MeV and  $m_c \sim 1.2$  GeV. Currently the data are too noisy and the shapes of the functions  $f_i$  are too similar for the fit to prefer a particular  $\Lambda_{HQ}$ . Including the heavy-quark discretization terms increases the decay constants by a few MeV and increases the error from  $\sim 1.8\%$  to  $\sim 3.8\%$ . The larger error now includes the residual heavy-quark discretization uncertainty in addition to residual light-quark discretization effects (encoded in Eq. (2.2)) as well as statistical errors. We quote a combined uncertainty from all the three sources of error.

### 4. Lattice spacing determination from $r_1$

The distance  $r_1$  is a property of the QCD potential between heavy quarks. The ratio  $r_1/a$ , for lattice spacing  $a$ , has been computed for all of the MILC gauge ensembles. At intermediate stages of the decay constant analysis quantities are converted from lattice units to  $r_1$  units using  $r_1/a$ . The value of  $r_1$  must then be input in order to convert results to physical units. The  $r_1$  value is also an input to the process of determining other quantities such as the bare charm quark masses as discussed in the next section.

Figure 1 depicts several  $r_1$  determinations. The first two determinations historically (circa 2004–2005) are labeled “HPQCD  $\Upsilon(2S-1S)$ ” [13] and “MILC  $\Upsilon(2S-1S)$ ” [14]. They are both based on the same analysis of the  $\Upsilon$  spectrum by the HPQCD Collaboration using a subset of the current MILC ensembles. The two determinations differ mainly in the details of the continuum extrapolation. The MILC Collaboration is also able to infer a value of  $r_1$  based on the value of  $f_\pi$  they find in their analysis of the light mesons. Recent light-meson analyses include results from finer lattice spacings than the earlier  $\Upsilon$  spectrum study and the resulting  $r_1$  values are known to better precision. The figure shows the result of two recent analyses labeled ‘MILC  $f_\pi$  2007’ [15] and “MILC  $f_\pi$  2009” [9]. The (preliminary) 2009 result agrees at the  $0.9\sigma$  level with the MILC  $\Upsilon$  value but differs from the HPQCD  $\Upsilon$  value at the  $1.8\sigma$  level. As these proceedings were being



**Figure 1:** Values of scale parameter  $r_1$  in fermi units. The “HPQCD  $\Upsilon(2S-1S)$ ” value uses the HPQCD collaboration  $\Upsilon$  spectrum results to set the physical value [13]. The “MILC  $\Upsilon(2S-1S)$ ” value derives from essentially the same spectrum analysis [14]. MILC determines  $r_1$  more precisely from their calculation of  $f_\pi$ : “MILC  $f_\pi$  2007” [15] and “MILC  $f_\pi$  2009” [9]. In a very recent update, “HPQCD 2009”, several physical quantities, including recent  $\Upsilon$  results, are used as inputs [16].

$r_1$ [fm]:		0.3108			0.318		
$a$	$\kappa$ run	$\kappa$ tune	$\delta\phi_s$	% $\phi_s$	$\kappa$ tune	$\delta\phi_s$	% $\phi_s$
0.09	0.127	0.1272	-0.0043	-0.56	0.1267	+0.0065	+0.84
0.12	0.122	0.1222	-0.0036	-0.50	0.1215	+0.0091	+1.26
0.15	0.122	0.1222	-0.0031	-0.42	0.1213	+0.0108	+1.47
$\delta f_{D_s}$ [MeV]		-1.8			+1.3		

**Table 1:** Tuning of  $\kappa$  charm at the three lattice spacings for two choices of  $r_1$ . The shift  $\delta\phi_s$  is the change in  $\phi$  at the strange quark mass when  $\kappa$  changes from the run value to tuned  $\kappa$  value. The corresponding change in extrapolated  $f_{D_s}$  is  $\delta f_{D_s}$ . In each case, all other extrapolation inputs are fixed to their appropriate ( $r_1$  dependent) values.

prepared, HPQCD published a new value for  $r_1$  [16], labeled “HPQCD 2009” in the figure, in much better agreement with MILC’s recent  $r_1$  values.

In this study, we use the MILC  $r_1$  determinations from  $f_\pi$  to set the physical scale. Our central value for  $r_1$  (the 2007 value) was also used in our studies of the semileptonic decays on the same lattices [17, 18]. The range of the 2009 MILC  $r_1$  determination is used to set a symmetric uncertainty around the central value. Hence, we take  $r_1 = 0.3108 \pm 0.0022$ . Our previous decay constant studies used the MILC  $\Upsilon$  value:  $r_1 = 0.318 \pm 0.007$  as an input which is about one  $\sigma$  higher.

## 5. Retuning kappa charm

We determine the value of  $\kappa$  for the charm quark by requiring that the spin-averaged kinetic masses of the lattice pseudoscalar and vector mesons made from a heavy clover quark and strange

source	$\phi_{D_s}$	$\phi_{D^+}$	$R_{D^+/D_s}$
statistics and discretization effects	2.6	3.4	1.3
chiral extrapolation	2.0	2.5	0.8
inputs $r_1$ , $m_s$ , $m_d$ and $m_u$	0.7	0.7	0.3
input $m_c$	1.0	1.2	0.2
$Z_V^{cc}$ and $Z_V^{qq}$	1.4	1.4	0
higher-order $\rho_{A_4}$	0.3	0.3	0.2
finite volume	0.2	0.6	0.6
total	3.8	4.7	1.7

**Table 2:** Uncertainties as a percentage of  $\phi$  and the ratio. The total combines all of the errors in quadrature.

asqtad valence quark equal the spin-averaged  $D_s$  meson mass. The tuning depends upon  $r_1$  in the conversion between lattice and physical masses.

In the past year, we have conducted new kappa-tuning runs with at least four times the statistics of our older tunings. At each lattice spacing, we simulated mesons for three values of  $\kappa$  around charm and three light-quark masses around strange allowing us to retune  $\kappa$  for a given  $r_1$ .

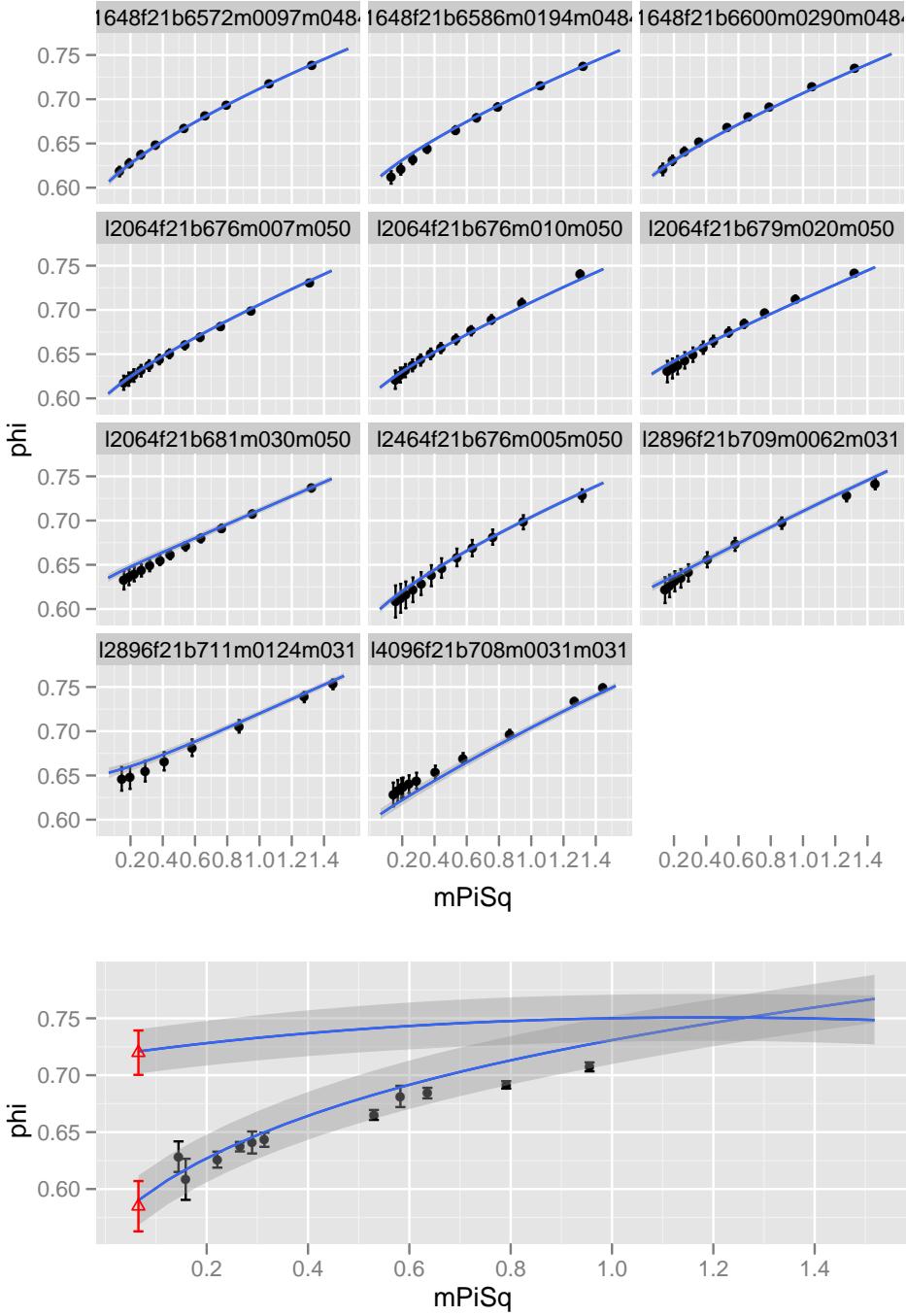
Table 1 shows preliminary tunings for  $\kappa$  charm based upon the two input values:  $r_1 = 0.3108$  fm (present value) and  $r_1 = 0.318$  fm (past studies). For each  $r_1$ , the (preliminary) tuned kappa and the corresponding change  $\delta\phi_s = \phi_s(\kappa \text{ tune}) - \phi_s(\kappa \text{ run})$  is listed by lattice spacing. The “run” kappa values are those used in the decay constant simulations. We adjust each  $\phi_q$  point by  $\delta\phi_s$  prior to the chiral extrapolation to correct for the mistuning of kappa. The bottom row of the table shows the resulting change in  $f_{D_s}$ . The opposite signs of the differences show that keeping kappa tuned partly compensates the change in  $r_1$ . We find that changing  $r_1$  from 0.318 fm to 0.3108 fm while keeping kappa charm tuned increases  $f_{D_s}$  by about 4.2 MeV.

## 6. The chiral and continuum extrapolation, results and uncertainty budget

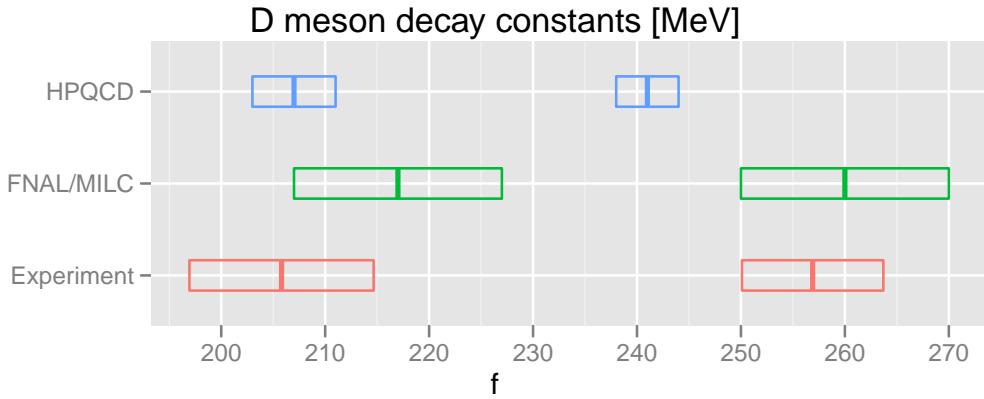
We fit  $\phi_{D_q}$  results from lattice simulations on eleven asqtad MILC ensembles [14] at the three lattice spacings:  $a \approx 0.09, 0.12$  and  $0.15$  fm. Our valence quark masses are in the range  $0.1m_s \leq m_q \lesssim m_s$ . Since our last report, we have doubled the statistics at the most chiral of the  $a \approx 0.09$  ensembles. The  $3 \times 4$  panel of plots at the top in Fig. 2 shows the  $\phi_{D_q}$  points and the fit where the fit function includes the lattice-spacing effects described in Sections 2 and 3. The plot at the bottom of Fig. 2 shows the extrapolations in the limit  $a = 0$ . The upper ( $D_s$ ) curve shows  $m_l \rightarrow \hat{m}$  setting  $m_q = m_h = m_s$ , while the lower ( $D^+$ ) curve shows  $m_q, m_l \rightarrow \hat{m}$  setting  $m_h = m_s$ . The physical quark mass inputs are from the MILC light meson analysis and  $\hat{m} = (m_u + m_d)/2$ . The points denoted by the red triangles correspond to physical  $f_{D_s}$  and  $f_{D^+}$ . Our preliminary results are:

$$f_{D^+} = 217 \pm 10 \text{ MeV}, \quad f_{D_s} = 260 \pm 10 \text{ MeV} \quad \text{and} \quad f_{D_s}/f_{D^+} = 1.20 \pm 0.02. \quad (6.1)$$

We have combined the statistical and the systematic uncertainties listed in Table 2 in quadrature. Our largest uncertainty is the combined uncertainty from statistical and residual discretization effects. The second largest uncertainty, chiral extrapolation, is an estimate of chiral expansion effects not included in the fit function and effects from variation in the extrapolation procedure. The



**Figure 2:** The preliminary  $D$  meson chiral extrapolation. The  $3 \times 4$  matrix of plots (top) show the  $\phi$  data and corresponding fit including  $a^2$  effects. Reading from left to right and top to bottom, plots correspond to  $(a, m_l/m_h)$  values of  $(0.15, 0.2)$ ,  $(0.15, 0.4)$ ,  $(0.15, 0.6)$ ,  $(0.12, 0.14)$ ,  $(0.12, 0.2)$ ,  $(0.12, 0.4)$ ,  $(0.12, 0.6)$ ,  $(0.12, 0.1)$ ,  $(0.09, 0.2)$ ,  $(0.09, 0.4)$  and  $(0.09, 0.1)$ . The larger plot (bottom) shows an overlay of the  $f_{D_s}$  and  $f_{D^+}$  extrapolations. The extrapolated curves are the fit (with error bands) taking  $a^2 \rightarrow 0$  and fixing/extrapolating the light quarks to their physical masses. The extrapolations are not expected to go through any of the points which are computed at finite  $a$ . None of the data points having  $m_q$  near  $m_s$  seen the upper panel are visible in the  $D_s$  extrapolation.



**Figure 3:** Comparisons of the Fermilab/MILC values of  $f_{D^+}$  and  $f_{D_s}$  to values from the HPQCD Collaboration [6] and recent experimental values [3][5].

third largest error is the statistical error in the nonperturbative calculation of the current renormalizations  $Z_V^{cc}$  and  $Z_V^{qq}$ . The value of  $f_{D_s}$  is about eleven MeV (one sigma) higher than our earlier value. Using nominal kappa values rather than tuned values at the previous  $r_1$  value accounts for about 1.3 MeV of the difference. Changing to the new  $r_1$  while keeping kappa tuned results in a 4.2 MeV increase. Incorporating heavy quark effects into the fit increases  $f_{D_s}$  by about 2 MeV. Higher statistics on the most chiral of the  $a \approx 0.09$  fm lattice increases  $f_{D_s}$  by about 1 MeV. These changes combine nonlinearly in the fit to yield the net increase.

Figure 3 compares the Fermilab and MILC Collaboration values for the decay constants with the HPQCD Collaboration [6] values and with the experimental results. The experimental result for  $f_{D^+}$  is from CLEO [3] while the  $f_{D_s}$  value is the Heavy Flavor Averaging Group average [5] of determinations by CLEO, BaBar and Belle. The Fermilab / MILC results remain in agreement with experiment. The total error on the experimental average for  $f_{D_s}$  is now smaller than our error providing a challenge for future lattice determinations. The apparent discrepancy between the HPQCD value of  $f_{D_s}$  and the other two  $f_{D_s}$  values is most striking. The HPQCD value is lower by about 1.8–2.1 $\sigma$ . The source of this difference may be clarified by further lattice simulations.

## 7. Summary and future plans

We have made several improvements in our analysis: a) discretization effects from both heavy and light quarks are modeled in our extrapolation function, b) we adopted a more precise  $r_1$  value which derived from the MILC  $f_\pi$  analysis rather than the  $r_1$  value related to early  $\Upsilon$  spectrum results c) we have improved the tuning of kappa charm. These improvements to the analysis will be more crucial in our next generation of decay constant study. We will increase statistics by a factor of four and extend the analysis to the finer lattice spacings  $a \approx 0.06$  and  $0.045$  fm which will reduce our combined statistical plus discretization error as well as help reduce uncertainties attributed to chiral extrapolation procedures. In addition, a new high-statistics computation of the nonperturbative part of the current renormalization aims for an error below the 0.5% level.

## References

- [1] C. Aubin *et al.*, *Charmed meson decay constants in three-flavor lattice QCD*, *Phys. Rev. Lett.* **95** (2005) 122002, [[hep-lat/0506030](#)].
- [2] **CLEO** Collaboration, M. Artuso *et al.*, *Improved Measurement of  $\mathcal{B}(D^+ \rightarrow \mu^+ \nu)$  and the Pseudoscalar Decay Constant f<sub>D<sub>+</sub></sub>*, *Phys. Rev. Lett.* **95** (2005) 251801, [[hep-ex/0508057](#)].
- [3] **CLEO** Collaboration, B. I. Eisenstein *et al.*, *Precision Measurement of  $\mathcal{B}(D^+ \rightarrow \mu^+ \nu)$  and the Pseudoscalar Decay Constant f<sub>D<sub>+</sub></sub>*, *Phys. Rev. D* **78** (2008) 052003, [[0806.2112](#)].
- [4] C. Bernard *et al.*, *B and D Meson Decay Constants*, *PoS LATTICE2008* (2008) 278, [[0904.1895](#)].
- [5] H. F. A. G. C. Physics), “f<sub>D<sub>s</sub></sub> world average.” [www.slac.stanford.edu/xorg/hfag/charm/PIC09/f\\_ds/results.html](http://www.slac.stanford.edu/xorg/hfag/charm/PIC09/f_ds/results.html), 2009.
- [6] **HPQCD** Collaboration, E. Follana, C. T. H. Davies, G. P. Lepage, and J. Shigemitsu, *High Precision determination of the  $\pi$ , K, D and D<sub>s</sub> decay constants from lattice QCD*, *Phys. Rev. Lett.* **100** (2008) 062002, [[0706.1726](#)].
- [7] B. A. Dobrescu and A. S. Kronfeld, *Accumulating evidence for nonstandard leptonic decays of D<sub>s</sub> mesons*, *Phys. Rev. Lett.* **100** (2008) 241802, [[0803.0512](#)].
- [8] C. Aubin and C. Bernard, *Staggered chiral perturbation theory for heavy-light mesons*, *Phys. Rev. D* **73** (2006) 014515, [[hep-lat/0510088](#)].
- [9] **The MILC** Collaboration, A. Bazavov *et al.*, *Results from the MILC collaboration’s SU(3) chiral perturbation theory analysis*, *PoS LAT2009* (2009) 079, [[0910.3618](#)].
- [10] A. S. Kronfeld, *Application of heavy-quark effective theory to lattice QCD. I: Power corrections*, *Phys. Rev. D* **62** (2000) 014505, [[hep-lat/0002008](#)].
- [11] J. Harada *et al.*, *Application of heavy-quark effective theory to lattice QCD. II: Radiative corrections to heavy-light currents*, *Phys. Rev. D* **65** (2002) 094513, [[hep-lat/0112044](#)].
- [12] M. B. Oktay and A. S. Kronfeld, *New lattice action for heavy quarks*, *Phys. Rev. D* **78** (2008) 014504, [[0803.0523](#)].
- [13] A. Gray *et al.*, *The Upsilon spectrum and m<sub>b</sub> from full lattice QCD*, *Phys. Rev. D* **72** (2005) 094507, [[hep-lat/0507013](#)].
- [14] C. Aubin *et al.*, *Light hadrons with improved staggered quarks: Approaching the continuum limit*, *Phys. Rev. D* **70** (2004) 094505, [[hep-lat/0402030](#)].
- [15] C. Bernard *et al.*, *Status of the MILC light pseudoscalar meson project*, *PoS LAT2007* (2007) 090, [[0710.1118](#)].
- [16] C. T. H. Davies, E. Follana, I. D. Kendall, G. P. Lepage, and C. McNeile, *Precise determination of the lattice spacing in full lattice QCD*, [0910.1229](#).
- [17] C. Bernard *et al.*, *The  $\bar{B} \rightarrow D^* \ell \bar{\nu}$  form factor at zero recoil from three-flavor lattice QCD: A Model independent determination of |V<sub>cb</sub>|*, *Phys. Rev. D* **79** (2009) 014506, [[0808.2519](#)].
- [18] J. A. Bailey *et al.*, *The  $B \rightarrow \pi \ell \bar{\nu}$  semileptonic form factor from three-flavor lattice QCD: A Model-independent determination of |V<sub>ub</sub>|*, *Phys. Rev. D* **79** (2009) 054507, [[0811.3640](#)].

## Visualization as a tool for understanding QCD evolution algorithms

This content has been downloaded from IOPscience. Please scroll down to see the full text.

2009 J. Phys.: Conf. Ser. 180 012068

(<http://iopscience.iop.org/1742-6596/180/1/012068>)

[View the table of contents for this issue](#), or go to the [journal homepage](#) for more

Download details:

IP Address: 140.192.180.252

This content was downloaded on 16/09/2014 at 14:51

Please note that [terms and conditions apply](#).

# Visualization as a tool for understanding QCD evolution algorithms

Massimo Di Pierro<sup>1</sup>, Michael Clark<sup>2</sup>, Chulwoo Jung<sup>3</sup>, James Osborn<sup>4</sup>, John Negele<sup>5</sup>, Richard Brower<sup>2</sup>, Steven Gottlieb<sup>6</sup>, Yaoqian Zhong<sup>1</sup>

<sup>1</sup> School of Computing, Depaul University; <sup>2</sup> Physics Department, Boston University;

<sup>3</sup> Physics Department, Brookhaven National Laboratory; <sup>4</sup> Argonne National Laboratory; <sup>5</sup> Center for Theoretical Physics, MIT; <sup>6</sup> Indiana University

**Abstract.** In this paper we present a project of visualizing topological charge in Lattice QCD in order to understand its evolution under the Markov Chain Monte Carlo algorithms and how to maximize its equilibration. Lattice QCD is a very computationally expensive technology for computing properties of composite particles from a few fundamental parameters such as quark masses. Our research plays an important role in validating these computations by showing that the autocorrelation length is small and it can eventually lead to even more efficient algorithms.

## 1. Introduction and Motivations

The Standard Model of Particle Physics provides a strikingly successful description of the structure of the matter down to scales of  $10^{-18}$ m. The basic degrees of freedom are elementary particles including quarks and leptons which interact with each other according to strong and electro-weak interactions. Quarks play a special role in this model because their interacting are mediated by the exchange of gluons; thus, they create bound states such as the proton, the neutron and the majority of particles that constitute the richness of structures that we observe in the physical world. The part of the Standard Model that describes quarks and gluons is called Quantum Chromo Dynamics (QCD). The ability to perform computations in QCD is of critical importance in Particle and Nuclear Physics. It enables us to determine fundamental parameters of the Standard Model from experiments, compare experimental results with model predictions, and eventually find discrepancies that can shed some light into yet unknown smaller structures and laws of physics at higher energy. It enables us to calculate the equation of state, transport coefficients, and other properties of hadronic matter and the quark gluon plasma explored in relativistic heavy ion collisions at RHIC the LHC. It also enables us to calculate the spectrum, structure, and interactions of mesons and baryons from first principles and lies at the heart of explorations at Jefferson Lab, RHIC spin, and a future electron-ion collider.

Since quarks are subject to strong interactions, standard perturbative techniques are inadequate to perform many interesting computations. For computing the spectrum of the model and computing the lifetime of composite particles, the purely numerical approach known as Lattice QCD that has been the most productive[2].

Lattice QCD consists of taking a small portion of space (a cube of  $\sim 10^{-15}$ m of size) for a brief time interval ( $\sim 10^{-23}$ secs), discretizing it on a four dimensional lattice, and simulating numerically the dynamics of QCD inside. The spatial fields do not evolve over time as in ordinary classical systems. Instead, the fields, that are defined on the four-dimensional Euclidean space-time, evolve over a fictitious MCMC time ( $\tau$ ). Lattice QCD computations consist of measuring correlation functions over the possible four dimensional configurations of the fields, each weighted by a number that encodes the dynamics of QCD[1]. Quantities like particles masses and lifetimes can be determined from these correlations.

In practice, the four dimensional configurations are created using Markov Chain Monte Carlo (MCMC) algorithms. Examples are the Molecular Dynamics algorithm or the Hybrid Monte Carlo algorithm. Each configuration is generated from the previous one using a transition in probability that encodes the laws of QCD dynamics.

In this paper, we describe how visualization techniques can be useful to provide information about these algorithms, about the data that is being generated.

The quark fields can be integrated out and, while they play a critical role in the inner workings of the algorithms, they are not part of the actual configurations. The latter only contain gauge degrees of freedom representing gluons, hence they are usually referred to as gauge configurations. Typical gauge configurations have sizes that range from  $32 \times 24^3$  to  $192 \times 64^3$  points; which correspond to a size ranging from 256 MBytes to 15 GBytes each (in single precision). A typical Lattice QCD computation generates and utilizes about 1000 of these gauge configurations, thus requiring multiple TBytes of storage. Algorithms are always parallel because of the size of the problem of Lattice QCD.

A typical problem consists in determining whether this small subset of the infinite space of all possible gauge configuration is sufficiently representative and whether the statistical uncertainty determined from the averages can be trusted. From a physical perspective the gauge configurations can be grouped into equivalence classes characterized by their topological charge density.

Our goal is to visualize this topological charge density and measure how fast/slow it evolved under the MCMC for different algorithms and different simulation parameters (the dynamic quark mass in particular). This would provide us with a proof that current Lattice QCD computation are adequately sampling the space of all possible configurations as opposed to getting stuck in one topological sector, and provide an effective tool for developing algorithms that maximize equilibration of the topological charge.

## 2. Visualization

Typical production gauge configurations appear to be separated by enough MCMC steps and there is no visible continuity in the topological charge. In order to measure autocorrelation in the topological charge we generated *ad hoc* gauge configurations with small trajectories for two different algorithms and different dynamical quark masses.

- **Wilson** Two sets of  $24^3 \times 32$  dynamical Wilson gauge configurations with  $\beta = 5.6$  and quark masses of  $m_1 = 66\text{MeV}$  and  $m_2 = 33\text{MeV}$  respectively. Each set consists of 1000 trajectories with a 0.1 molecular dynamics time units, saved every 2 time units ( $dt = 0.2$ ). To integrate Hamilton's equations a multi-step 2nd order symmetric symplectic integrator was used (Omelyan) [3].
- **DWF**: One set of  $24^3 \times 64$  dynamical gauge configurations generated with Domain Wall Fermions (two degenerate light ones,  $m_u = 14\text{MeV}$ , and one strange,  $m_s = 74\text{MeV}$ ) and Iwasaki gauge action at  $\beta = 2.13$ , saved every 1/6 of a trajectory ( $dt = 0.1\bar{6}$ ). Further details can be found in [4] and the references therein.

Each gauge configuration consists of a set of  $SU(3)$  matrices  $U_{x,y}$  for every pair of neighbor sites  $x, y$  on the lattice. Given a quark field  $q_y$ , the  $U_{x,y}$  matrix represent the change in the quark field under parallel transport from  $y \rightarrow x$ , *i.e.*,  $q_y \rightarrow q_x = U_{x,y}q_y$ . Therefore  $U_{y,x} = U_{x,y}^H$ .

The topological charge is computed in three steps according to the definition:

- The gauge field was smoothed by  $n$ -iterations of a “cooling step”

$$U_{x,y} \rightarrow U_{x,y} = P_{SU(3)}(\alpha U_{x,y} + (1 - \alpha)S_{x,y}) \quad (1)$$

where  $P_{SU(3)}$  is a projection operator into  $SU(3)$  and  $S_{x,y}$  is the average of all products of three consecutive links connecting  $x$  with  $y$ .

- We computed the chromo-electro-magnetic field defined by

$$F_{x,\mu,\nu} = \frac{1}{2}\text{Re}(A_{x,\mu,\nu} - A_{x,\mu,\nu}^H) \quad (2)$$

where  $A_{x,\mu,\nu}$  is the average of the product of four consecutive links in the  $\mu, \nu$  plane starting and ending at point  $x$ .

- We computed the topological charge density, defined by

$$Q_x = \frac{1}{32\pi^2} \sum_{\mu\nu\rho\sigma} \epsilon_{\mu\nu\rho\sigma} \text{Tr}(F_{x,\mu,\nu} F_{x,\rho,\sigma}) \quad (3)$$

In fig. 1 we show three images of iso-surfaces of topological charge density for one gauge configuration for different numbers of cooling steps (0,10,15). In fig. 2 we show three images of iso-surfaces of topological charge density for the same number of cooling steps (20) for different consecutive gauge configurations separated by 6 trajectories.

### 3. Autocorrelation and Hurst Exponent

In addition to the visualization of the topological charge density we also computed the average autocorrelation  $R(\tau)$  and average Hurst exponent  $H$  for each independent Markov Chain. First we computed the autocorrelation and the Hurst exponent over the MCMC time ( $\tau$ ) for each lattice point, then we averaged the results over all lattice points. The Hurst exponent is computed using the standard R/S technique described in ref. [5]. In fractal geometry, the Hurst exponent is also known as “index of dependence”, it measures the relative tendency of a time series to either strongly regress to the mean or cluster in a direction.  $H$  is 0.5 for white noise.  $H$  is related to the auto-correlation length via  $R(\tau) \simeq \tau^{2H-2}$  and  $2 - H$  is known as the “fractal dimension” of the problem.

Our results are summarized in the following table.

Action	quark masses	H
Wilson	66MeV	0.966
Wilson	33MeV	0.973
DWF	69,17MeV	0.908

The autocorrelation is plotted in fig. 3

Notice how the autocorrelation becomes negative at around 100 MCMC steps. This may indicate an alternating pattern in the data, but a larger MCMC chain will be required to validate this hypothesis.

### 4. Conclusion

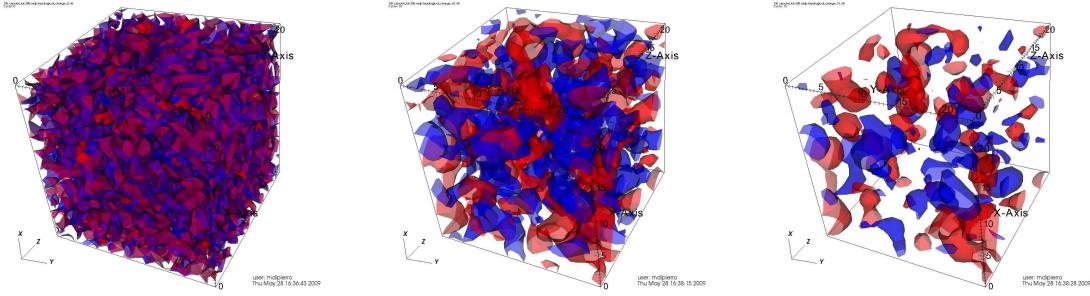
Our analysis demonstrates that the autocorrelation length of a topological charge is very small compared with the trajectory length of typical production grade Lattice QCD computations. Our work is still in progress and, as more data is analyzed, we hope to derive an empirical interpolating formula for the autocorrelation length as function of the simulation parameters.

### Acknowledgments

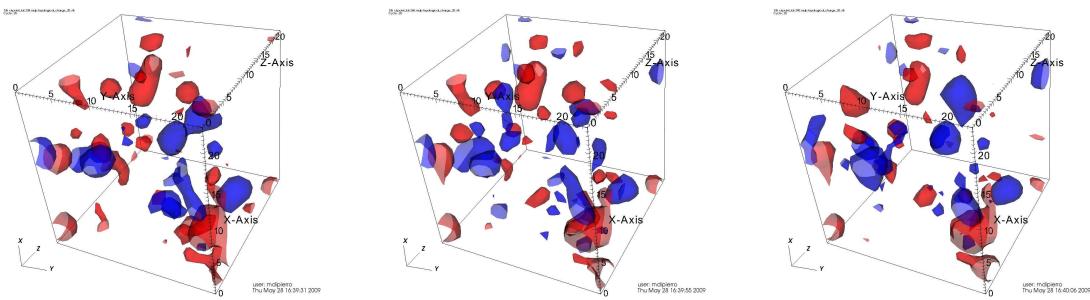
This research is funded by the Department of Energy grant DE-FC02-06ER41441. We thank Argonne National Laboratory computing resources, the DOE Advanced Simulation and Computing Initiative (ASCI) for the VisIt software, and Brian Fox at DePaul Univ.

### Bibliography

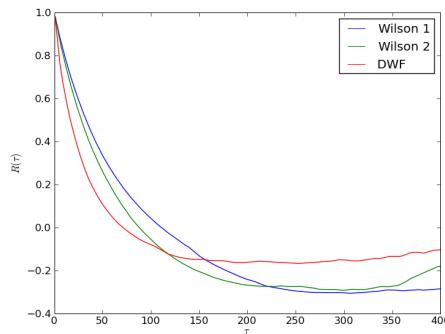
- [1] Pierro M D 2008 *Modern Physics A* **125:012067**
- [2] Kronfeld A 2006 *J. Phys. Conf. Ser.* **21-3**
- [3] Takaishi T and de Forcrand P 2006 *Phys. Rev. E* **73** 036706 (*Preprint hep-lat/0505020*)
- [4] Jung C 2007 *POS(LATTICE 2007)* **37** (*Preprint arXiv:0710.5337*)
- [5] Hurst E 1951 *Trans. Am. Soc. Civil Eng.* **116** 770



**Figure 1.** Iso-surfaces for a 3D slice of a topological charge density for one gauge configuration for different numbers of cooling steps (0,10, 15).



**Figure 2.** Iso-surfaces for a 3D slice of topological charge density for the same number of cooling steps (20) for different consecutive gauge configurations separated by 6 trajectories.



**Figure 3.** Autocorrelation length for the three sets of gauge configurations considered in the paper.

# Visualization of semileptonic form factors from lattice QCD

C. Bernard,<sup>1</sup> C. DeTar,<sup>2</sup> M. Di Pierro,<sup>3</sup> A. X. El-Khadra,<sup>4</sup> R. T. Evans,<sup>4</sup> E. D. Freeland,<sup>5</sup> E. Gamiz,<sup>4</sup> Steven Gottlieb,<sup>6</sup> U. M. Heller,<sup>7</sup> J. E. Hetrick,<sup>8</sup> A. S. Kronfeld,<sup>9</sup> J. Laiho,<sup>1</sup> L. Levkova,<sup>2</sup> P. B. Mackenzie,<sup>9</sup> M. Okamoto,<sup>9</sup> M. B. Oktay,<sup>2</sup> J. N. Simone,<sup>9</sup> R. Sugar,<sup>10</sup> D. Toussaint,<sup>11</sup> and R. S. Van de Water<sup>12</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>Department of Physics, Washington University, St. Louis, Missouri, USA

<sup>2</sup>Physics Department, University of Utah, Salt Lake City, Utah, USA

<sup>3</sup>School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois, USA

<sup>4</sup>Physics Department, University of Illinois, Urbana, Illinois, USA

<sup>5</sup>Liberal Arts Department, The School of the Art Institute of Chicago, Chicago, Illinois, USA

<sup>6</sup>Department of Physics, Indiana University, Bloomington, Indiana, USA

<sup>7</sup>American Physical Society, Ridge, New York, USA

<sup>8</sup>Physics Department, University of the Pacific, Stockton, California, USA

<sup>9</sup>Fermi National Accelerator Laboratory, Batavia, Illinois, USA

<sup>10</sup>Department of Physics, University of California, Santa Barbara, California, USA

<sup>11</sup>Department of Physics, University of Arizona, Tucson, Arizona, USA

<sup>12</sup>Physics Department, Brookhaven National Laboratory, Upton, New York, USA

(Received 16 June 2009; published 21 August 2009)

Comparisons of lattice-QCD calculations of semileptonic form factors with experimental measurements often display two sets of points, one each for lattice QCD and experiment. Here we propose to display the output of a lattice-QCD analysis as a curve and error band. This is justified, because lattice-QCD results rely in part on fitting, both for the chiral extrapolation and to extend lattice-QCD data over the full physically allowed kinematic domain. To display an error band, correlations in the fit parameters must be taken into account. For the statistical error, the correlation comes from the fit. To illustrate how to address correlations in the systematic errors, we use the Bećirević-Kaidalov parametrization of the  $D \rightarrow \pi l\nu$  and  $D \rightarrow K l\nu$  form factors, and an analyticity-based fit for the  $B \rightarrow \pi l\nu$  form factor  $f_+$ .

DOI: 10.1103/PhysRevD.80.034026

PACS numbers: 13.20.Fc, 12.38.Gc, 13.20.He

The past several years have witnessed considerable improvement in our understanding of semileptonic decays of  $D$  and  $B$  mesons. Measurements have advanced in accuracy from 6%–20% on the normalization [1–3] and  $\sim 10\%$  on the shape [4] to  $\sim 1\%$  on both [5]. Meanwhile, *ab initio* calculations in QCD with lattice gauge theory have become realistic [6–9], now incorporating the effects of sea quarks that were omitted in earlier work [10–14]. In this article, we discuss how to present both together, so that the agreement (or, in principle, lack thereof) is easy to assess.

We focus on reactions mediated by electroweak vector currents, leading to pseudoscalar mesons,  $\pi$  or  $K$ , in the final state. At the quark level, a heavy quark  $h$  decays into a daughter quark  $d$  (not necessarily the down quark), with a spectator antiquark  $\bar{q}$ . Writing the decay  $H \rightarrow Pl\nu$ , the form factors are defined by

$$\langle P|V^\mu|H\rangle = f_+(q^2)(p_H + p_P - \Delta)^\mu + f_0(q^2)\Delta^\mu, \quad (1)$$

where  $q = p_H - p_P$  is the 4-momentum of the lepton system, and  $\Delta^\mu = (p_H + p_P) \cdot q q^\mu / q^2 = (m_H^2 - m_P^2)q^\mu / q^2$ . Equation (1) is general, applying to  $K \rightarrow \pi l\nu$  as well as to  $D$  and  $B$  decays. For lattice QCD, it is more convenient to express the transition matrix element as

$$\langle P|V^\mu|H\rangle = \sqrt{2m_H}[\nu^\mu f_{||}(E) + p_\perp^\mu f_\perp(E)], \quad (2)$$

where  $\nu = p_H/m_H$ , and  $p_\perp = p_P - Ev$  and  $E = \nu \cdot p_P$  denote the 3-momentum and energy of the final-state meson in the rest frame of the initial state. The energy  $E$  is related to  $q^2$  via

$$q^2 = m_H^2 + m_P^2 - 2m_H E. \quad (3)$$

Neglecting the lepton mass,  $0 \leq q^2 \leq q_{\max}^2 = (m_H - m_P)^2$  is kinematically allowed in the semileptonic decay.

The form factors  $f_+(q^2)$  and  $f_0(q^2)$  are related to  $f_{||}(E)$  and  $f_\perp(E)$  by

$$f_+(q^2) = (2m_H)^{-1/2}[f_{||}(E) + (m_H - E)f_\perp(E)], \quad (4)$$

$$f_0(q^2) = \frac{\sqrt{2m_H}}{m_H^2 - m_P^2}[(m_H - E)f_{||}(E) - p_\perp^2 f_\perp(E)], \quad (5)$$

with Eq. (3) understood. Equations (4) and (5) imply  $f_+(0) = f_0(0)$ , as required in Eq. (1).

Two aspects of lattice-QCD calculations are important here. First (as in all lattice-QCD calculations), it is computationally demanding to have a spectator quark with mass as small as those of the up and down quarks;

for  $P = \pi$  the same applies to the daughter quark. In recent unquenched calculations, the mass of the  $\bar{q}q$  pseudoscalar  $P_{\bar{q}q}$  lies in the range  $0.1m_K^2 \lesssim m_{P_{\bar{q}q}}^2 \lesssim m_K^2$ . Second (of special importance in semileptonic decays), the calculations take place in a finite spatial volume, so the 3-momentum takes discrete values. In typical cases the box size  $L \approx 2.5$  fm, so the smallest nonzero momentum  $\mathbf{p}_{(1,0,0)} = 2\pi(1, 0, 0)/L$  satisfies  $|\mathbf{p}_{(1,0,0)}| \approx 500$  MeV.

After generating numerical data at several values of  $(E, m_{P_{\bar{q}q}}^2)$ , the next step for lattice-QCD calculations is to carry out a chiral extrapolation,  $m_{P_{\bar{q}q}}^2 \rightarrow m_\pi^2$ , of the data for  $f_\perp$  and  $f_\parallel$  [15,16]. The chiral extrapolation must reflect the fact that the form factors are analytic in  $E = \sqrt{\mathbf{p}^2 + m_P^2}$ , not  $\mathbf{p}$  [17]. Note also that  $f_\perp$  and  $f_+$  can be computed only with  $\mathbf{p} \neq \mathbf{0}$ , hence  $E > m_P$  or, equivalently,  $q^2 < q_{\max}^2$ . The statistical and discretization uncertainties in  $f_+(q^2, m_{P_{\bar{q}q}}^2)$  start out smallest at  $q_{(1,0,0)}^2$ , corresponding to  $\mathbf{p}_{(1,0,0)}$ . A sensible chiral extrapolation will propagate this feature to  $f_+(q^2, m_\pi^2)$ . Similarly, the statistical and discretization uncertainties in  $f_0(q^2, m_\pi^2)$  are smallest near  $q_{\max}^2$ .

When  $|\mathbf{p}|a$  becomes too large, discretization effects grow out of control. Therefore, the kinematic domain of lattice-QCD calculations is limited to, these days,  $|\mathbf{p}| \lesssim 1$  GeV, with a corresponding upper limit on  $E$  and lower limit on  $q^2$ . To extend the form factor over the full physical kinematic domain, a parametrization of the  $q^2$  dependence is needed.

One choice is the Bećirević-Kaidalov (BK) ansatz [18]

$$f_+(q^2) = \frac{F}{(1 - \tilde{q}^2)(1 - \alpha\tilde{q}^2)}, \quad (6)$$

$$f_0(q^2) = \frac{F}{1 - \tilde{q}^2/\beta}, \quad (7)$$

where  $\tilde{q}^2 = q^2/m_{H^*}^2$  ( $H^*$  is the vector meson of flavor  $h\bar{d}$ ), and  $F$ ,  $\alpha$ , and  $\beta$  are free parameters to be fitted. A key feature of Eq. (6) is the built-in pole at  $q^2 = m_{H^*}^2$ , or  $E = -(m_{H^*}^2 - m_H^2 - m_P^2)/2m_H < 0$ , an indisputable feature of the physical  $f_+$ . Further singularities at higher negative energy are modeled by the BK parameters  $\alpha$  and  $\beta$ . A similar possibility is the Ball-Zwicky (BZ) ansatz [8,19], which has one more parameter for  $f_+$  than BK. A shortcoming of these parametrizations is that comparisons of lattice-QCD and experimental slope parameters can be misleading [20,21], because lattice-QCD slopes are determined near  $q^2 = q_{\max}^2$ , whereas experimental slopes are determined near  $q^2 = 0$ .

Another approach based on analyticity and unitarity is to write the form factors as

$$f_+(q^2) = \frac{1}{(1 - \tilde{q}^2)\phi_+(q^2)} \sum_{k=0}^N a_k z^k, \quad (8)$$

$$f_0(q^2) = \frac{1}{\phi_0(q^2)} \sum_{k=0}^N b_k z^k, \quad (9)$$

where  $\phi_{+,0}$  are arbitrary, but suitable, functions, and the series coefficients are fit parameters. The variable

$$z = \frac{\sqrt{1 - q^2/t_+} - \sqrt{1 - t_0/t_+}}{\sqrt{1 - q^2/t_+} + \sqrt{1 - t_0/t_+}}, \quad (10)$$

where  $t_+ = (m_H + m_P)^2$  and  $t_0$  can be chosen to make  $|z|$  small for all kinematically allowed  $q^2$ . Like BK and BZ, Eq. (8) builds the  $H^*$  pole into  $f_+$ , but this approach is model independent because unitarity [17,22,23] and heavy-quark physics [21] impose bounds on  $\sum_k |a_k|^2$ ,  $\sum_k |b_k|^2$ , and because kinematics set  $|z| < 1$ . Consequently, the series can be truncated safely, once additional terms are negligible compared to other uncertainties in the analysis.

In all approaches the output of an analysis of lattice-QCD form factors is a fit, usually a two-stage fit of chiral extrapolation followed by  $q^2$  parametrization. Clearly, the final fit describes a curve, and the error matrix of the fit parameters describes an error band. Nevertheless, lattice-QCD results usually have been plotted as a set of points with error bars at fiducial values of  $q^2$  (or  $E$ ). These points evoke the underlying discrete nature of the 3-momentum  $\mathbf{p}$  but, in general, the chosen values of  $q^2$  (or  $E$ ) have nothing to do with the original discrete values of  $\mathbf{p}$ . A plot with a curve plus error band exhibits the same information, while giving a visually superior sense of the correlations between points on the curve.

The experimental measurements of  $f_+(q^2)$  come from counting events in bins of  $q^2$  and removing coupling and kinematic factors. The analysis inevitably entails some fitting, to correct for acceptance, etc., but the postfit bins of  $q^2$  faithfully mirror the input to such fits.

If one would like to compare the calculations with the measurements, it is appealing to represent one as a curve with error band, and the other as points with error bars. Bearing the foregoing remarks in mind, it seems natural to draw the curve for lattice-QCD calculations. A few years ago, we prepared illustrative plots for  $D \rightarrow K l \nu$  with the Fermilab-MILC [6,7] lattice-QCD calculations and FOCUS [4] and Belle [24] measurements. The intent was pedagogical, and we showed the plots at seminars and conferences [25].

Unfortunately, the error band in that effort was impressionistic, not rigorous. With the prospect of yet-more-precise results based on CLEO-*c*'s full accumulation of  $818 \text{ pb}^{-1}$  [5], we now present a version that treats the error band as rigorously as possible. We also prepare plots for  $D \rightarrow \pi l \nu$  and  $B \rightarrow \pi l \nu$ .

As before we shall base the plots for  $D$  decays on Ref. [6]. The final result of this analysis consists of the BK parameters ( $F, \alpha, \beta$ ) and the  $3 \times 3$  error matrix. The full statistical error matrix is contained in a detailed, un-

TABLE I. Best-fit values of BK parameters with statistical and systematic errors, successively, in parentheses [6,7,26].

Decay	$F$	$\alpha$	$\beta$
$D \rightarrow K l \nu$	0.73(3)(7)	0.50(4)(7)	1.31(7)(13)
$D \rightarrow \pi l \nu$	0.64(3)(6)	0.44(4)(7)	1.41(6)(7)

TABLE II. Statistical error correlation matrices  $\rho_{ij} = \sigma_{ij}^2 / (\sigma_{ii}^2 \sigma_{jj}^2)^{1/2}$  of the BK parameters [26].

$D \rightarrow K$	$F$	$\alpha$	$\beta$
$F$	1.000	-0.597	0.530
$\alpha$	-0.597	1.000	-0.316
$\beta$	0.530	-0.316	1.000
$D \rightarrow \pi$	$F$	$\alpha$	$\beta$
$F$	1.000	-0.583	0.535
$\alpha$	-0.583	1.000	-0.312
$\beta$	0.535	-0.312	1.000

published description of a BK-based analysis of  $B \rightarrow \pi l \nu$  form factors [26]. The best fit, statistical errors, and systematic errors are tabulated in Table I. The statistical correlation matrices  $\rho_{ij} = \sigma_{ij}^2 / (\sigma_{ii}^2 \sigma_{jj}^2)^{1/2}$  are tabulated in Table II. The correlations among systematic errors are discussed below.

Propagating (correlated) fluctuations in  $F$ ,  $\alpha$ , and  $\beta$  to the form factors, one finds relative squared errors

$$\frac{\sigma_{++}^2}{f_+^2} = \frac{\sigma_{FF}^2}{F^2} + 2 \frac{\sigma_{F\alpha}^2}{F} \frac{\tilde{q}^2}{1 - \alpha \tilde{q}^2} + \sigma_{\alpha\alpha}^2 \left( \frac{\tilde{q}^2}{1 - \alpha \tilde{q}^2} \right)^2, \quad (11)$$

$$\frac{\sigma_{00}^2}{f_0^2} = \frac{\sigma_{FF}^2}{F^2} - 2 \frac{\sigma_{F\beta}^2}{F\beta} \frac{\tilde{q}^2}{\beta - \tilde{q}^2} + \frac{\sigma_{\beta\beta}^2}{\beta^2} \left( \frac{\tilde{q}^2}{\beta - \tilde{q}^2} \right)^2. \quad (12)$$

These errors are plotted as a function of  $q^2$  in Fig. 1 as solid curves.

The relative statistical errors are smallest for  $q^2$  such that

$$\sigma_{F\alpha}^2 = -F \sigma_{\alpha\alpha}^2 \tilde{q}^2 / (1 - \alpha \tilde{q}^2), \quad (13)$$

$$\sigma_{F\beta}^2 = F \sigma_{\beta\beta}^2 \tilde{q}^2 / \beta(\beta - \tilde{q}^2). \quad (14)$$

It is illustrative to take  $\sigma_{F\alpha}^2$  and  $\sigma_{F\beta}^2$  from Tables I and II and solve Eqs. (13) and (14) for  $\tilde{q}^2$ . We call these values  $\tilde{q}_\alpha^2$  and  $\tilde{q}_\beta^2$  and tabulate them, as well as  $\tilde{q}_{(1,0,0)}^2$  and  $\tilde{q}_{\max}^2$ , in Table III.

As one can see from Fig. 1 and Table III, the statistical error is smallest between  $\tilde{q}_{(1,0,0)}^2$  and  $\tilde{q}_{\max}^2$ , as expected. One may view this outcome as a check on the fitting procedures.

One can reverse this strategy to determine the correlation between the systematic errors of  $F$  and the slope parameters. In the error budget of Ref. [6] the largest

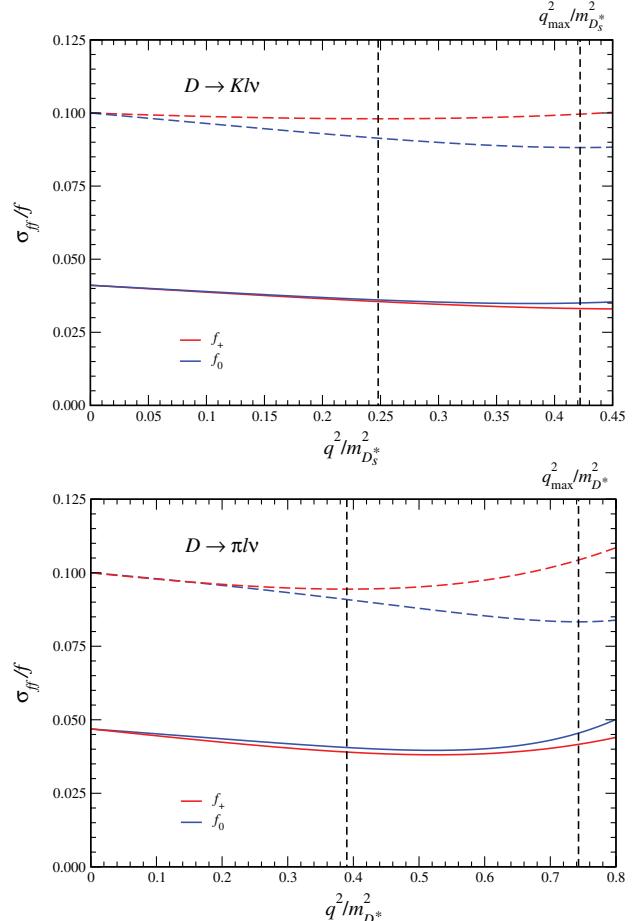


FIG. 1 (color). Relative errors vs  $q^2$ . Solid (dashed) curves show the fitted statistical (estimated systematic) error for  $f_+$  (red curves) and  $f_0$  (blue curves). Vertical lines show  $q_{(1,0,0)}^2$  and  $q_{\max}^2$ .

systematic effect comes from discretization errors. These should be smallest around  $q_{(1,0,0)}^2$  and  $q_{\max}^2$  for  $f_+$  and  $f_0$ , respectively, because those correspond to the smallest  $|p|$  yielding the respective matrix elements. This yields

$$\rho_{F\alpha}^{\text{syst}} = -0.198(D \rightarrow K), \quad -0.329(D \rightarrow \pi), \quad (15)$$

$$\rho_{F\beta}^{\text{syst}} = +0.471(D \rightarrow K), \quad +0.533(D \rightarrow \pi), \quad (16)$$

and the dashed curves in Fig. 1.

It is customary to combine statistical and systematic uncertainties by adding the two  $\sigma^2$  (matrices). Carrying out this procedure leads to the curves and bands in Fig. 2. The error bands seem to contradict the conventional wisdom that the lattice-QCD uncertainties are smallest near  $q_{\max}^2$ . This is not entirely the case for the *relative* error, as seen in Fig. 1. As  $q^2$  increases, the relative errors decrease until hitting a minimum somewhere between  $q_{(1,0,0)}^2$  and  $q_{\max}^2$ , as is reasonable. The form factors rise faster than the relative errors drop, leading to the increasing absolute error seen in Fig. 2. These features are not an artifact of the BK

TABLE III. Useful quantities for generating and assessing Figs. 1–3.

Decay	$H^*$	$m_{H^*}$ (MeV)	$E_{(1,0,0)}$ (MeV)	$q_{(1,0,0)}^2$ (GeV $^2$ )	$q_{\max}^2$ (GeV $^2$ )	$\tilde{q}_{(1,0,0)}^2$	$\tilde{q}_{\max}^2$	$\tilde{q}_{\alpha}^2$	$\tilde{q}_{\beta}^2$
$D \rightarrow K l \nu$	$D_s^*$	2112	704	1.10	1.88	0.25	0.42	0.47	0.38
$D \rightarrow \pi l \nu$	$D^*$	2008	518	1.57	3.00	0.39	0.74	0.53	0.52
$B \rightarrow \pi l \nu$	$B^*$	5325	518	22.4	26.4	0.79	0.93	...	...

parametrization, as we shall see below with the  $B \rightarrow \pi l \nu$  form factor  $f_+$ .

Figure 2 is the first main result of this article. It shows the form factors  $f_+$  and  $f_0$  for  $D \rightarrow K l \nu$  and  $D \rightarrow \pi l \nu$ . The lattice-QCD results are shown as curves (red for  $f_+$ , blue for  $f_0$ ) with two error bands, one statistical (orange for  $f_+$ , gray for  $f_0$ ), the other systematic and statistical combined (yellow for  $f_+$ , light blue for  $f_0$ ). Experimental measurements for  $f_+$  [24,27–29] are overlaid as points with error bars. It may require careful scrutiny to see which

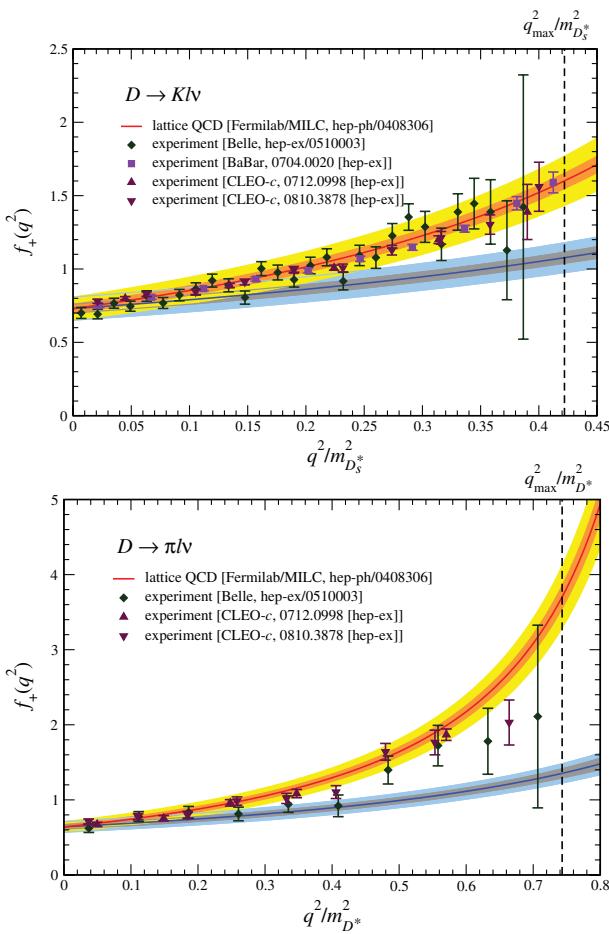


FIG. 2 (color). Form factors  $f_+$  (and  $f_0$ ) for semileptonic  $D$  decays, from lattice QCD [6,26], expressed as a red (blue) curve with an orange (gray) statistical error band and a yellow (light blue) combined error band. Error bands take correlations into account. Measurements of  $f_+$  are from Belle (green diamonds) [24], BABAR (magenta squares) [27], and CLEO- $c$  (maroon triangles) [28,29]. The vertical line shows  $q_{\max}^2$ .

experiment is which, but a glance reveals how well the points and curves agree. The agreement is good for  $D \rightarrow \pi l \nu$  and very good for  $D \rightarrow K l \nu$ .

For the  $z$  expansion the propagation of errors is even simpler. Focusing on  $f_+$ , one has from Eq. (8)

$$\frac{\sigma_{++}^2}{f_+^2} = \frac{\sum_{k,l=0}^N \sigma_{kl}^2 z^{k+l}}{[\sum_{k=0}^N a_k z^k]^2}, \quad (17)$$

where the indices on  $\sigma^2$  correspond to those on the series coefficients. The coefficients and error matrix for the  $N = 3$  fit ( $t_0 = 0.65 q_{\max}^2$ ) are tabulated in Table IV.

The  $z$ -series fit was carried out after assigning  $q^2$ -dependent systematic uncertainties, so Table IV refers

TABLE IV. Best-fit values  $a_k$  and correlation matrix  $\rho_{kl}$  of the 3-term  $z$  expansion of  $f_+$  for  $B \rightarrow \pi l \nu$ , with statistical and systematic errors combined [9].

Fit:	0.0216(27)	-0.0378(191)	-0.113(27)
$\rho$	$a_0$	$a_1$	$a_2$
$a_0$	1.000	0.640	0.475
$a_1$	0.640	1.000	0.964
$a_2$	0.474	0.964	1.000

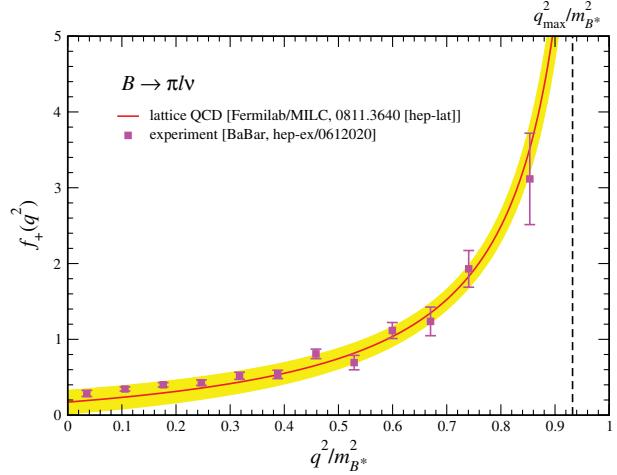


FIG. 3 (color). Form factor  $f_+$  for  $B \rightarrow \pi l \nu$  expressed as a curve (red) from the best fit with a total error band (yellow) from taking correlations in the fit parameters into account [9], overlaid with measurements of  $|V_{ub}| f_+ / (3.38 \times 10^{-3})$  from BABAR (magenta squares) [30]. The vertical line shows  $q_{\max}^2$ .

to the combined statistical and systematic errors of this analysis.

This information, combined with the outer function  $\phi_+$  [9], is used to produce Fig. 3, the second main result of this paper. Now the curve and error band conform with preconceptions, for several reasons. First,  $q_{(1,0,0)}^2$  is close to  $q_{\max}^2$ , rather than in the middle of the kinematic range. Second, the chiral extrapolation in Ref. [9] is less aggressive than that in Ref. [6], leading to a larger but more realistic error at  $q^2 = 0$ . The most striking aspect is that even though the absolute error in  $f_+$  is increasing for  $\tilde{q}^2 \geq 0.8$ , the band remains narrow. The band simply conveys the point-to-point correlations better.

This paper shows in detail how to compare semileptonic form factors from lattice QCD and from experiments. For illustration we use the BK parametrization for  $D \rightarrow Kl\nu$  and  $D \rightarrow \pi l\nu$ , and the  $z$  expansion for  $B \rightarrow \pi l\nu$ . Clearly, the idea is more general. For example, an interesting prospect relevant to semileptonic form factors is to inject 3-momenta smaller than  $\mathbf{p}_{(1,0,0)}$  using “twisted” boundary conditions [31–33]. That strategy should improve the accuracy of parameters in the chiral extrapolation and, hence, the BK, BZ, or  $z$  fits. The output of any fit could still be exhibited as outlined here, although one should bear in mind that superior visualization of a fitting procedure does not repair any shortcomings of the fit itself.

We would like to thank Ian Shipsey for encouraging us to think carefully about the correlations in the systematic errors. We would like to thank Laurenz Widhalm for providing the Belle data in numerical form [24], and Shipsey for the *BABAR* and CLEO  $D$ -decay data [27–29]. Computations for this work were carried out in part on facilities of the USQCD Collaboration, which are funded by the Office of Science of the United States Department of Energy. This work was supported in part by the U.S. Department of Energy under Grants No. DE-FC02-06ER41446 (C. D., L. L., M. B. O.), No. DE-FG02-91ER40661 (S. G.), No. DE-FG02-91ER40677 (A. X. E. K., R. T. E., E. G.), No. DE-FG02-91ER40628 (C. B., J. L.), and No. DE-FG02-04ER41298 (D. T.); by the National Science Foundation under Grants No. PHY-0555243, No. PHY-0757333, No. PHY-0703296 (C. D., L. L., M. B. O.), No. PHY-0555235 (J. L.), and No. PHY-0757035 (R. S.); and by Universities Research Associates (R. T. E., E. G.). This manuscript has been coauthored by an employee of Brookhaven Science Associates, LLC, under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

- 
- [1] M. Ablikim *et al.* (BES Collaboration), Phys. Lett. B **597**, 39 (2004).
  - [2] G. S. Huang *et al.* (CLEO Collaboration), Phys. Rev. Lett. **94**, 011802 (2005).
  - [3] J. M. Link *et al.* (FOCUS Collaboration), Phys. Lett. B **607**, 51 (2005).
  - [4] J. M. Link *et al.* (FOCUS Collaboration), Phys. Lett. B **607**, 233 (2005).
  - [5] D. Besson *et al.* (CLEO Collaboration), Phys. Rev. D **80**, 032005 (2009).
  - [6] C. Aubin *et al.* (Fermilab Lattice, MILC, and HPQCD Collaborations), Phys. Rev. Lett. **94**, 011601 (2005).
  - [7] M. Okamoto *et al.* (Fermilab Lattice, MILC, and HPQCD Collaborations), Nucl. Phys. B, Proc. Suppl. **140**, 461 (2005).
  - [8] E. Dalgic néé Gulez *et al.* (HPQCD Collaboration), Phys. Rev. D **73**, 074502 (2006); **75**, 119906(E) (2007).
  - [9] J. A. Bailey *et al.*, Phys. Rev. D **79**, 054507 (2009).
  - [10] K. C. Bowler *et al.* (UKQCD Collaboration), Phys. Lett. B **486**, 111 (2000).
  - [11] A. Abada *et al.*, Nucl. Phys. **B619**, 565 (2001).
  - [12] A. X. El-Khadra *et al.*, Phys. Rev. D **64**, 014502 (2001).
  - [13] S. Aoki *et al.* (JLQCD Collaboration), Phys. Rev. D **64**, 114505 (2001).
  - [14] J. Shigemitsu *et al.*, Phys. Rev. D **66**, 074506 (2002).
  - [15] D. Bećirević, S. Prelovšek, and J. Zupan, Phys. Rev. D **67**, 054010 (2003).
  - [16] C. Aubin and C. Bernard, Phys. Rev. D **76**, 014002 (2007).
  - [17] L. Lellouch, Nucl. Phys. **B479**, 353 (1996).
  - [18] D. Bećirević and A. B. Kaidalov, Phys. Lett. B **478**, 417 (2000).
  - [19] P. Ball and R. Zwicky, Phys. Rev. D **71**, 014015 (2005).
  - [20] R. J. Hill, Phys. Rev. D **73**, 014012 (2006).
  - [21] T. Becher and R. J. Hill, Phys. Lett. B **633**, 61 (2006).
  - [22] C. Bourrely, B. Machet, and E. de Rafael, Nucl. Phys. **B189**, 157 (1981).
  - [23] C. G. Boyd, B. Grinstein, and R. F. Lebed, Phys. Rev. Lett. **74**, 4603 (1995); C. G. Boyd and M. J. Savage, Phys. Rev. D **56**, 303 (1997).
  - [24] L. Widhalm *et al.* (Belle Collaboration), Phys. Rev. Lett. **97**, 061804 (2006); K. Abe *et al.* (Belle Collaboration), arXiv:hep-ex/0510003.
  - [25] A. S. Kronfeld *et al.* (Fermilab Lattice, MILC, and HPQCD Collaborations), Proc. Sci., LAT2005 (2006) 206; Int. J. Mod. Phys. A **21**, 713 (2006); A. S. Kronfeld (Fermilab Lattice, MILC, and HPQCD Collaborations), J. Phys. Conf. Ser. **46**, 147 (2006).
  - [26] M. Okamoto (Fermilab Lattice and MILC Collaborations) (unpublished). A public report of this work is contained in Ref. [7]. This line of analysis was superseded by Ref. [9].

- [27] B. Aubert *et al.* (*BABAR* Collaboration), Phys. Rev. D **76**, 052005 (2007).
- [28] D. Cronin-Hennessy *et al.* (*CLEO* Collaboration), Phys. Rev. Lett. **100**, 251802 (2008); S. Dobbs *et al.* (*CLEO* Collaboration), Phys. Rev. D **77**, 112005 (2008).
- [29] J. Y. Ge *et al.* (*CLEO* Collaboration), Phys. Rev. D **79**, 052010 (2009).
- [30] B. Aubert *et al.* (*BABAR* Collaboration), Phys. Rev. Lett. **98**, 091801 (2007).
- [31] P. F. Bedaque, Phys. Lett. B **593**, 82 (2004).
- [32] C. T. Sachrajda and G. Villadoro, Phys. Lett. B **609**, 73 (2005).
- [33] J. M. Flynn, A. Jüttner, and C. T. Sachrajda (UKQCD Collaboration), Phys. Lett. B **632**, 313 (2006).

# Quarkonium mass splittings in three-flavor lattice QCD

T. Burch,<sup>1</sup> C. DeTar,<sup>1</sup> M. Di Pierro,<sup>2</sup> A. X. El-Khadra,<sup>3</sup> E. D. Freeland,<sup>4</sup> Steven Gottlieb,<sup>5,6</sup> A. S. Kronfeld,<sup>7</sup> L. Levkova,<sup>1</sup> P. B. Mackenzie,<sup>7</sup> and J. N. Simone<sup>7</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>*Department of Physics, University of Utah, Salt Lake City, Utah, USA*

<sup>2</sup>*School of Computing, DePaul University, Chicago, Illinois, USA*

<sup>3</sup>*Physics Department, University of Illinois, Urbana, Illinois, USA*

<sup>4</sup>*Department of Physics, Washington University, St. Louis, Missouri, USA*

<sup>5</sup>*Department of Physics, Indiana University, Bloomington, Indiana, USA*

<sup>6</sup>*National Center for Supercomputing Applications, University of Illinois, Urbana, Illinois, USA*

<sup>7</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois, USA*

(Received 17 December 2009; published 12 February 2010)

We report on calculations of the charmonium and bottomonium spectrum in lattice QCD. We use ensembles of gauge fields with three flavors of sea quarks, simulated with the asqtad improved action for staggered fermions. For the heavy quarks we employ the Fermilab interpretation of the clover action for Wilson fermions. These calculations provide a test of lattice QCD, including the theory of discretization errors for heavy quarks. We provide, therefore, a careful discussion of the results in light of the heavy-quark effective Lagrangian. By and large, we find that the computed results are in agreement with experiment, once parametric and discretization errors are taken into account.

DOI: 10.1103/PhysRevD.81.034508

PACS numbers: 12.38.Gc, 14.40.Pq

## I. INTRODUCTION

Quarkonium plays an important role in the application of QCD to hadronic physics. Early calculations of charmonium based on potential models gave strong support to the interpretation of these states as bound states of a new heavy quark [1,2]. Although physically the charmonium state is analogous to positronium, its historical role as a model system of QCD proved to be analogous to that of the hydrogen atom in quantum mechanics. The charmonium spectrum provided a simple example of how QCD works, made even more compelling with the subsequent observation of the bottomonium states.

Although the analysis of the quarkonium spectrum based on potential models is a significant triumph of QCD, an *ab initio* calculation based on lattice QCD, an approach that can simultaneously deal with light quarks would be even more satisfying. However, in lattice QCD, the continuum limit requires that  $am$  approaches zero, where  $a$  is the lattice spacing and  $m$  is the mass of the state. Quarkonium states and their constituent quarks are so heavy that it is often impractical to use so small a lattice spacing that  $am$  is small. Both lattice nonrelativistic QCD (NRQCD) [3,4] and the Fermilab action [5] have been developed to treat heavy quarks in lattice gauge theory. Thus, the successful calculation of the spectrum of charmonium and bottomonium becomes a significant test of these techniques.

This paper describes the current state of the quarkonium spectrum based on the Fermilab approach to heavy quarks, using gauge configurations provided by the MILC

Collaboration [6] that incorporate the effects of three light quarks: up, down, and strange. We have been studying the quarkonium spectrum using this formalism for some time [7], starting on ensembles with two flavors of sea quark [8], and new ensembles of configurations have become available during the course of the project. In this paper, we report on results with four lattice spacings from  $a \approx 0.18$  fm to  $\approx 0.09$  fm. We do not yet consider this work a definitive calculation using our approach. Results from two finer lattice spacings should become available in the future. As we detail below, the tuning of the bare valence heavy-quark masses, via the heavy-light meson spectrum [9], is not yet precise enough to give satisfactory answers to all the questions that have arisen. In the future, we expect to have better control of the heavy-quark masses. Nevertheless, already we can successfully reproduce important features of the quarkonium spectrum, and we consider this another important test bed in which to assess the errors that arise from our treatment of the heavy quarks. Knowledge of these errors is also important for calculations of properties of heavy-light mesons, for example, those pertaining to semileptonic decays [10], leptonic decays [11], and  $B$ - $\bar{B}$  mixing [12], as well as the heavy-light spectrum [9].

Prior lattice QCD work on quarkonium with different sea-quark content has been reviewed by Bali [13]. More recently, Dudek and collaborators have used the quenched approximation to explore decays [14], radiative transitions [15], and the excited-state spectrum [16] of charmonium. Meinel has used lattice NRQCD to compute the bottomo-

nium spectrum at one lattice spacing on four ensembles with  $2 + 1$  domain-wall sea quarks [17]. The HPQCD Collaboration has calculated the quarkonium spectrum on many of the same MILC ensembles used in our work, using lattice NRQCD for bottomonium [18,19], and using highly improved staggered quarks (HISQ) for charmonium [20]. Lattice NRQCD is not very accurate for charmonium, and HISQ requires very small lattice spacings for  $b$  quarks [21]. An advantage of the Fermilab method is that it allows the same treatment for both the charmed and bottom quarks. Predictions of the  $c\bar{b}$  spectrum have also been made, for the pseudoscalar  $B_c$  using NRQCD  $b$  quarks and the charmed quark propagators from this project [22], and also for the vector  $B_c^*$  using NRQCD  $b$  quarks and HISQ charmed quarks [23].

The plan of this paper is as follows. In Sec. II, we describe our methodology, explaining the actions used for gluons, light sea quarks, and the heavy valence quarks. We describe, in detail, how to use the heavy-quark effective Lagrangian to understand heavy-quark discretization errors in quarkonium masses. We also discuss the construction of the hadronic correlators and how they are fit to determine meson masses. Several broad issues inform the uncertainties (statistical and systematic), and they are discussed in Sec. III. In Sec. IV, we show our results for the splittings between various states or combinations of states. Where possible, we have organized the presentation of the mass splittings around individual terms in the heavy-quark effective action, which clarifies the approach to the continuum limit at available lattice spacings. Section V contains our conclusions and suggestions on ways to improve on this calculation.

## II. METHODOLOGY

In this section, we collect several sets of information needed to understand the results that follow. Section II A defines the notation for different quarkonium states and splittings. Then we provide details of the lattice gauge configurations that we have used in Sec. II B. Next, in Sec. II C, we review the Fermilab method, discussing in detail how it can be understood via an effective Lagrangian. This discussion provides a link between the lattice fermion action and the computed mass splittings; an important theme in this paper is to scrutinize our numerical results according to these theoretical expectations. Last, we explain how we form correlation functions in Sec. II D, and how we fit them to obtain masses in Sec. II E.

### A. Notation

In this paper, we use two notations for hadrons and their masses, both the standard names from the Particle Data Group [24] and the spectroscopic notation  $n^{2S+1}L_J$ , where  $S$ ,  $L$ , and  $J$  are the spin, orbital, and total angular momentum, respectively, of the  $n$ th radial excitation. As usual,  $L = 0, 1, 2, \dots$  are denoted  $S, P, D, \dots$

It is often convenient to discuss spin-averaged masses (and mass splittings), which we indicate with a horizontal line, such as  $\overline{1S}$  or  $\overline{1^3P}$ . In particular,

$$M(\overline{1S}) = \frac{1}{4}(M_{\eta_c} + 3M_{J/\psi}), \quad (2.1)$$

$$M(\overline{1^3P}) = \frac{1}{9}(M_{\chi_{c0}} + 3M_{\chi_{c1}} + 5M_{\chi_{c2}}), \quad (2.2)$$

using charmonium for illustration. For brevity we usually write  $\overline{1P}$  for  $\overline{1^3P}$ . These spin-averages are sensitive to the leading term in a nonrelativistic expansion. Note that the  $\overline{1P}$  and  $1^1P_1$  (also denoted  $h_c$  and  $h_b$ ) levels are nearly the same in nature, which can be explained by the spin-spin interaction's short range— $\delta(\mathbf{r})$  in the context of potential models [2].

Complementary to the spin-averaged masses are spin-splittings that hone in on spin-dependent corrections [25,26]. Below, we examine the hyperfine splittings (HFS)

$$M(nS_{\text{HFS}}) = M_{J/\psi} - M_{\eta_c}, \quad (2.3)$$

using charmonium  $1S$  notation on the right-hand side. For the  $P$  states two combinations are of interest:

$$M(nP_{\text{spin-orbit}}) = \frac{1}{9}(5M_{\chi_{c2}} - 2M_{\chi_{c0}} - 3M_{\chi_{c1}}), \quad (2.4)$$

$$M(nP_{\text{tensor}}) = \frac{1}{9}(3M_{\chi_{c1}} - M_{\chi_{c2}} - 2M_{\chi_{c0}}), \quad (2.5)$$

again using charmonium  $1S$  notation on the right-hand side.  $M(nS_{\text{HFS}})$  and  $M(nP_{\text{tensor}})$  are sensitive to spin-spin interactions, and  $M(nP_{\text{spin-orbit}})$  to spin-orbit interactions.

### B. Configuration details

These calculations have been carried out on lattice gauge configurations provided by the MILC Collaboration [6], listed in Table I. They were generated via the  $R$  algorithm [27] with the one-loop Symanzik-improved Lüscher-Weisz gluon action [28] combined with  $2 + 1$  flavors of sea quarks simulated with the asqtad action [29]. The  $n_f$ -dependent part of the one-loop couplings [30] became available only after the ensembles were generated. We have used ensembles at four lattice spacings:  $a \approx 0.18, 0.15, 0.12$ , and  $0.09$  fm (also called in the text “extra-coarse,” “medium-coarse,” “coarse,” and “fine” ensembles, respectively). The first four columns of Table I list the parameters of these ensembles, including the masses of the sea quarks, denoting the pair as  $am_l/am_s$ . The lattice scale of each ensemble with different sea-quark masses was kept approximately fixed using the length  $r_1$  [31,32] from the static quark potential. The absolute scale from the  $Y\ 2S-1S$  splitting was determined on most of our ensembles by the HPQCD Collaboration [18,19]. Details on the  $r_1$  determinations can be found in review of other work on the MILC ensembles [33]. Combining this deter-

TABLE I. Run parameters and configuration numbers for the ensembles used to study charmonium and bottomonium  $\eta$ ,  $J/\psi$ ,  $\Upsilon$ ,  $h$ ,  $\chi_0$ , and  $\chi_1$  states with relativistic operators, and  $h$  and all  $\chi_J$  states with nonrelativistic operators. The labels  $a$  and  $b$  are used to distinguish between the two runs with the same  $\beta = 6.76$  but different  $am_l/am_s$ .

$a$ (fm)	$\beta$	$am_l/am_s$	$N_s^3 \times N_t$	$\kappa_c$	Relativistic		Nonrelativistic		
					$N_{\text{conf}}^c$	$\kappa_b$	$N_{\text{conf}}^b$	$\kappa_c$	$N_{\text{conf}}^c$
$\approx 0.18$	6.503	0.0492/0.082	$16^3 \times 48$	0.120	401	...	...	0.120	400
	6.485	0.0328/0.082	$16^3 \times 48$	0.120	331			0.120	501
	6.467	0.0164/0.082	$16^3 \times 48$	0.120	645			0.120	647
	6.458	0.0082/0.082	$16^3 \times 48$	0.120	400			0.120	601
$\approx 0.15$	6.600	0.0290/0.0484	$16^3 \times 48$	...	...	...	...	0.122	580
	6.586	0.0194/0.0484	$16^3 \times 48$	0.122	631	0.076	631	0.122	580
	6.572	0.0097/0.0484	$16^3 \times 48$	0.122	631	0.076	631	0.122	629
	6.566	0.00484/0.0484	$20^3 \times 48$	...	...	...	...	0.122	601
$\approx 0.12$	6.81	0.03/0.05	$20^3 \times 64$	0.122	549	0.086	549	...	...
	6.79	0.02/0.05	$20^3 \times 64$	0.122	460	0.086	460		
	6.76, $a$	0.01/0.05	$20^3 \times 64$	0.122	593	0.086	539		
	6.76, $b$	0.007/0.05	$20^3 \times 64$	0.122	403	...	...		
$\approx 0.09$	7.11	0.0124/0.031	$28^3 \times 96$	0.127	517	0.0923	517	0.127	518
	7.09	0.0062/0.031	$28^3 \times 96$	0.127	557	0.0923	557	0.127	557
	7.08	0.0031/0.031	$40^3 \times 96$	0.127	504	0.0923	504	0.127	504

mination with more recent work [34,35], leads us to take the range  $r_1 = 0.318^{+0.009}_{-0.007}$  fm in this paper.

### C. Heavy-quark formulation

In this work, the charmed and bottom quarks are simulated with the Fermilab action [5]

$$\begin{aligned} S = & \sum_n \bar{\psi}_n \psi_n - \kappa \sum_n [\bar{\psi}_n (1 - \gamma_4) U_{n,4} \psi_{n+\hat{4}} \\ & + \bar{\psi}_{n+\hat{4}} (1 + \gamma_4) U_{n,4}^\dagger \psi_n] - \kappa \zeta \sum_{n,i} [\bar{\psi}_n (r_s - \gamma_i) U_{n,i} \psi_{n+i} \\ & + \bar{\psi}_{n+i} (r_s + \gamma_i) U_{n,i}^\dagger \psi_n] - c_B \kappa \zeta \sum_n \bar{\psi}_n i \boldsymbol{\Sigma} \cdot \mathbf{B}_n \psi_n \\ & - c_E \kappa \zeta \sum_n \bar{\psi}_n \boldsymbol{\alpha} \cdot \mathbf{E}_n \psi_n, \end{aligned} \quad (2.6)$$

where  $U$  denotes the gluon field, and  $\psi$  and  $\bar{\psi}$  denote the quark and antiquark fields. The clover definitions of the chromomagnetic and chromoelectric fields  $\mathbf{B}$  and  $\mathbf{E}$  are standard and given, for example, in Ref. [5]. When  $\zeta = r_s = 1$  and  $c_B = c_E = 0$ ,  $S$  reduces to the Wilson action [36]; when  $\zeta = r_s = 1$  and  $c_B = c_E = c_{\text{SW}}$ , it reduces to the Sheikholeslami-Wohlert action [37]. The relation between the hopping parameter  $\kappa$  and the bare mass is

$$m_0 a = \frac{1}{2\kappa} - 1 - 3r_s \zeta \quad (2.7)$$

in four space-time dimensions.

To motivate our choices of the input parameters  $\kappa$ ,  $\zeta$ ,  $r_s$ ,  $c_B$ , and  $c_E$ , let us review the nonrelativistic interpretation of Wilson fermions [5]. The pole energy of a single quark of spatial momentum  $\mathbf{p}$  is

$$E(\mathbf{p}) = m_1 + \frac{\mathbf{p}^2}{2m_2} + O(p^4), \quad (2.8)$$

where the quark rest mass  $m_1$  and the kinetic mass  $m_2$  are defined at all orders of perturbation theory via the self energy [38]. At the tree level,

$$m_1 = a^{-1} \ln(1 + m_0 a), \quad (2.9)$$

$$\frac{1}{m_2} = \frac{2\zeta^2}{m_0(2 + m_0 a)} + \frac{ar_s \zeta}{1 + m_0 a}. \quad (2.10)$$

In general,  $m_1 \neq m_2$  unless  $m_0 a \ll 1$ ; for charmed and bottom quarks on the ensembles listed in Table I one has  $m_{0c} a \lesssim 1$ ,  $m_{0b} a \gtrsim 1$ . One could tune  $\zeta$  so that  $m_2 = m_1$ , and we shall revisit that strategy below.

Equation (2.8) is the simplest example of a nonrelativistic interpretation of physical quantities computed with the action in Eq. (2.6). This is justified, because in quarkonium the relative momentum of the heavy quarks is small compared with the heavy-quark mass. This is the basis of the phenomenological success of potential models, which yield estimates of the relative velocity and, equivalently, internal momentum. For charmonium

$$v \approx 0.55, \quad p \approx 840 \text{ MeV}, \quad (2.11)$$

and for bottomonium

$$v \approx 0.31, \quad p \approx 1475 \text{ MeV}. \quad (2.12)$$

For both systems the typical kinetic energy is 450 MeV, as seen, for example, in the  $\overline{1P}-\overline{1S}$  splitting. The kinetic energies  $\frac{1}{2}m_2 v^2$  are small on our lattices, and the momenta  $m_2 v$  are marginally small (especially for bottomonium).

The nonrelativistic interpretation can be extended beyond the tree level and to higher order in the nonrelativistic expansion using effective field theories. This has been pursued in detail emphasizing heavy-light hadrons [39–41], and here we explain the ideas in the context of quarkonium. As in the Symanzik effective theory, one introduces a continuum effective Lagrangian, but here it is an effective Lagrangian valid for heavy quarks. With quarkonium, the appropriate power-counting rule for the effective Lagrangian is that of nonrelativistic QED [42] and nonrelativistic QCD [3,4,43]. So one has

$$S \doteq - \sum_s \int d^4x \mathcal{L}_{\text{HQ}}^{(s)}, \quad (2.13)$$

where  $s$  counts the powers of velocity. Here  $\doteq$  means that the lattice gauge theory on the left-hand side, defined in our case by Eq. (2.6), is given an effective description by the right-hand side. The first several terms of the effective Lagrangian are

$$\begin{aligned} \mathcal{L}_{\text{HQ}}^{(2)} = & -\bar{h}^{(+)}(D_4 + m_1)h^{(+)} + \frac{\bar{h}^{(+)}\mathbf{D}^2h^{(+)}}{2m_2} \\ & -\bar{h}^{(-)}(D_4 + m_1)h^{(-)} + \frac{\bar{h}^{(-)}\mathbf{D}^2h^{(-)}}{2m_2}, \end{aligned} \quad (2.14)$$

$$\begin{aligned} \mathcal{L}_{\text{HQ}}^{(4)} = & \frac{\bar{h}^{(+)}i\boldsymbol{\sigma} \cdot \mathbf{B}h^{(+)}}{2m_B} + \frac{\bar{h}^{(+)}i\boldsymbol{\sigma} \cdot (\mathbf{D} \times \mathbf{E})h^{(+)}}{8m_E^2} \\ & + \frac{\bar{h}^{(+)}(\mathbf{D} \cdot \mathbf{E})h^{(+)}}{8m_{E'}^2} + \frac{\bar{h}^{(+)}(\mathbf{D}^2)^2h^{(+)}}{8m_4^3} \\ & + \frac{1}{6}a^3w_4\bar{h}^{(+)}D_i^4h^{(+)} + \frac{\bar{h}^{(-)}i\boldsymbol{\sigma} \cdot \mathbf{B}h^{(-)}}{2m_B} \\ & - \frac{\bar{h}^{(-)}i\boldsymbol{\sigma} \cdot (\mathbf{D} \times \mathbf{E})h^{(-)}}{8m_E^2} - \frac{\bar{h}^{(-)}(\mathbf{D} \cdot \mathbf{E})h^{(-)}}{8m_{E'}^2} \\ & + \frac{\bar{h}^{(-)}(\mathbf{D}^2)^2h^{(-)}}{8m_4^3} + \frac{1}{6}a^3w_4\bar{h}^{(-)}D_i^4h^{(-)}, \end{aligned} \quad (2.15)$$

where  $h^{(+)}$  is a two-component field describing the quark, and  $h^{(-)}$  is a two-component field describing the antiquark. The short-distance coefficients  $m_1$ ,  $m_2^{-1}$ ,  $m_B^{-1}$ ,  $m_E^{-2}$ ,  $m_{E'}^{-2}$ ,  $m_4^{-3}$ , and  $w_4$  depend on the bare quark masses, the bare gauge coupling, and all other couplings of the (improved) lattice action. The terms in  $\mathcal{L}_{\text{HQ}}^{(s)}$  scale with the heavy quark's velocity as  $v^s$ , with the rules [4]  $\mathbf{D} \sim m_2 v$ ,  $\mathbf{E} \sim m_2^2 v^3$ , and  $\mathbf{B} \sim m_2^2 v^4$ . In particular, the nonrelativistic kinetic energy,  $\mathbf{D}^2/2m_2 \sim \frac{1}{2}m_2 v^2$ , is an essential part of quarkonium dynamics, which is why  $m_2$  appears with  $v$  in the power counting. The short-distance coefficients  $m_B^{-1}$ ,  $m_E^{-2}$ , etc., can be expanded in perturbation theory, with  $\alpha_s \sim v$  [4]. We have put the rest mass  $m_1 \bar{h}^{(\pm)}h^{(\pm)}$  and temporal kinetic energy  $\bar{h}^{(\pm)}D_4h^{(\pm)}$  into  $\mathcal{L}_{\text{HQ}}^{(2)}$ , because by

the equation of motion  $D_4 + m_1 \sim \mathbf{D}^2/2m_2 \sim \frac{1}{2}m_2 v^2$ . The next set of terms,  $\mathcal{L}_{\text{HQ}}^{(6)}$ , are not written out, because they are numerous yet merely describe subleading contributions to the splittings examined below.

One would like to adjust  $\kappa$ ,  $\zeta$ ,  $r_s$ ,  $c_B$ , and  $c_E$  so that the lattice gauge theory matches continuum QCD with controllable uncertainty. One would also like to reduce the number of input parameters as much as possible, to make the simulation easier to carry out. The coupling  $r_s$  is redundant: any choice is allowed as long as the doubling problem is solved. We take

$$r_s = 1. \quad (2.16)$$

To derive tuning criteria for the others, one refers to the NRQCD description of continuum QCD, which takes the same form as Eqs. (2.14) and (2.15), but with the following substitutions:

$$m_1 \mapsto m, \quad (2.17)$$

$$m_2 \mapsto m, \quad (2.18)$$

$$\frac{1}{m_B} \mapsto \frac{Z_B}{m}, \quad (2.19)$$

$$\frac{1}{m_E^2} \mapsto \frac{Z_E}{m^2}, \quad (2.20)$$

$$\frac{1}{m_{E'}^2} \mapsto \frac{Z_{E'}}{m^2}, \quad (2.21)$$

$$\frac{1}{m_4^3} \mapsto \frac{Z_4}{m^3}, \quad (2.22)$$

$$w_4 \mapsto 0, \quad (2.23)$$

where the last is a consequence of Lorentz invariance, as is the exact equality of the rest and kinetic masses. The matching factors  $Z_i$  are unity at the tree level and have a perturbative expansion. To bring the lattice field theory in line with continuum QCD, one must then simply adjust the lattice couplings so that the lattice quantities on the left in (2.17), (2.18), (2.19), (2.20), (2.21), (2.22), and (2.23) become, to some accuracy, the continuum quantities on the right. In principle, this matching could be carried out non-perturbatively [44], although we do not pursue that strategy here.

If one restricts one's attention to mass splittings and matrix elements, it is not necessary to adjust a coupling to tune  $m_1$ . The operators  $\bar{h}^{(\pm)}h^{(\pm)}$  are number operators, commuting with everything else in the Hamiltonian [39]. It is therefore acceptable to tolerate a large discretization error in the rest mass and, consequently, one does not need to adjust  $\zeta$ . We take

$$\zeta = 1. \quad (2.24)$$

To obtain the correct dynamics, one must adjust  $\kappa$  so that the rest of  $\mathcal{L}_{\text{HQ}}^{(2)}$  is correctly tuned. In other words, one must identify the kinetic quark mass  $m_2$  with the physical quark mass.

The adjustment of  $c_B$  stems from a concrete realization of (2.19). At the tree level

$$\frac{1}{m_B} = \frac{2\zeta^2}{m_0(2+m_0a)} + \frac{ac_B\zeta}{1+m_0a}, \quad (2.25)$$

so to ensure  $m_B = m_2$  (as desired at the tree level where  $Z_B = 1$ ), one needs  $c_B = r_s$ . In practice, we take [recalling Eq. (2.16)]

$$c_B = u_0^{-3} \quad (2.26)$$

to account for tadpole diagrams at higher orders in perturbation theory [45]. On the coarse ensembles, we set  $u_0$  from the Landau link; on the other ensembles, we set it from the plaquette.

In principle, the adjustment of  $c_E$  should stem from (2.20). These simulations have been carried out, however, in concert with calculations of heavy-light masses [9], for which the adjustment of  $c_E$  is a subleading effect [39,46]. Thus, we have taken

$$c_E = c_B. \quad (2.27)$$

Using formulas in Ref. [47], we can estimate the error stemming from  $1/m_E^2$ , finding a tree-level mismatch of

$$\frac{1}{4m_E^2} - \frac{1}{4m_2^2} = \frac{a^2}{(2+m_0a)(1+m_0a)} - \frac{a^2}{4(1+m_0a)^2}, \quad (2.28)$$

where the right-hand side holds for  $\zeta = r_s = c_B = c_E = 1$ . At the tree level  $m_E = m_{E'}$ , so the same error is made in the Darwin terms  $\bar{h}^{(\pm)}\mathbf{D} \cdot \mathbf{E} h^{(\pm)}$ .

An advantage of using Eqs. (2.13), (2.14), and (2.15) to describe our lattice calculation is that it clarifies which parameters in  $S$  play a key role in various splittings defined in Sec. II A. The spin-averaged masses receive energy (beyond  $2m_1$ ) from the balance between the kinetic energies  $\bar{h}^{(\pm)}\mathbf{D}^2 h^{(\pm)}$  and the exchange of temporal gluons between  $\bar{h}^{(+)}A_4 h^{(+)}$  and  $\bar{h}^{(-)}A_4 h^{(-)}$ . As discussed above, they are sensitive to  $m_2$ , motivating the tuning of  $\kappa$  (and the fixed choice for  $\zeta$ ). The hyperfine splittings  $M(nS_{\text{HFS}})$  arise from exchange of spatial gluons between  $\bar{h}^{(+)}i\boldsymbol{\sigma} \cdot \mathbf{B} h^{(+)}$  and  $\bar{h}^{(-)}i\boldsymbol{\sigma} \cdot \mathbf{B} h^{(-)}$ . Hence they are proportional to  $1/m_B^2$  and, drilling further back to  $S$ , sensitive to the coupling  $c_B$ . The same line of dependency holds for the tensor splittings  $M(nP_{\text{tensor}})$ . Similarly, the spin-orbit parts of the  $\chi_{cJ}$  and  $\chi_{bJ}$  levels arise from exchange of a temporal gluon between  $\bar{h}^{(\pm)}i\boldsymbol{\sigma} \cdot (\mathbf{D} \times \mathbf{E}) h^{(\pm)}$  and  $\bar{h}^{(\mp)}A_4 h^{(\mp)}$ . Hence they are proportional to  $1/m_E^2$  and, referring back to  $S$ , sensitive to  $c_E$ .

With the tree-level adjustment of  $c_B$ , the hyperfine splittings should be expected to have errors of order  $\alpha_s m v^4$  from radiative corrections to  $m_B^{-1}$  [relative error:  $O(\alpha_s) \sim O(v)$ ], and of order  $v^6$  from the terms  $\bar{h}^{(\pm)}\{\mathbf{D}^2, i\boldsymbol{\sigma} \cdot \mathbf{B}\}h^{(\pm)}$  in  $\mathcal{L}_{\text{HQ}}^{(6)}$  [relative error:  $O(v^2)$ ]. Similarly, with  $c_E = c_B$ , we expect leading errors of order  $a^2 m^3 v^4$  in the spin-orbit part of the  $\chi$  splittings [relative error:  $O(m^2 a^2)$ ], as well as radiative corrections to  $m_E^{-2}$  [relative error: again  $O(\alpha_s) \sim O(v)$ ]. On the MILC ensembles both relative errors are expected to be a few to several percent [47] and, perhaps counterintuitively, smaller for bottomonium than charmonium [47].

The lattice action in Eq. (2.6) does not contain parameters to tune the two terms proportional to  $p^4$  in Eq. (2.15). The mismatches

$$\begin{aligned} \frac{1}{8m_4^3} - \frac{1}{8m_2^3} &= \frac{a^2}{2m_0(2+m_0a)^2(1+m_0a)} \\ &+ \frac{a^2(1+4m_0a)}{4m_0(2+m_0a)(1+m_0a)^2} \\ &+ \frac{m_0a^4}{8(1+m_0a)^3} \end{aligned} \quad (2.29)$$

and

$$a^3 w_4 = \frac{2a^2}{m_0(2+m_0a)} + \frac{a^3}{4(1+m_0a)} \quad (2.30)$$

(given again for  $\zeta = r_s = c_B = c_E = 1$ ) cause errors of order  $a^2 m^3 v^4$  in the spin-averaged splittings. The relative errors,  $O(m^2 a^2)$ , are again expected to be a few to several percent, but in this case larger for bottomonium than for charmonium [47]. For plots of the  $a$  dependence of discretization effects caused by Eqs. (2.28), (2.29), and (2.30), see Figs. 2 and 3 of Ref. [47].

To tune  $\kappa$  nonperturbatively, one adjusts it so that a hadron mass agrees with the experimentally measured value. Let us define  $M_1$  and  $M_2$  for a hadron analogously to Eq. (2.8).<sup>1</sup> From the effective Lagrangian description for quarkonium, Eqs. (2.13), (2.14), and (2.15), it follows that

$$M_1 = 2m_1 + B_1, \quad (2.31)$$

$$M_2 = 2m_2 + B_2, \quad (2.32)$$

where the binding energy  $B_1$  is determined by terms of order  $v^2$  and higher, but  $B_2$  by terms of order  $v^4$  and higher [48]. In the splittings of rest masses,  $m_1$  drops out, so we can obtain well-tuned results for  $B_1$  (and their differences) by adjusting  $\kappa$  so that  $m_2$  corresponds to a physical quark. That suggests tuning  $\kappa$  so that, say,  $M_2(\overline{1S})$  agrees with experiment. The spin average is useful, because it elimi-

<sup>1</sup>In this paper, we use  $m_1, m_2, \dots$  for quark masses, and  $M_1, M_2, \dots$  for hadron masses.

nates the leading effect of a mistuned chromomagnetic coupling  $c_B$ .

A better approach, still using a hadron's kinetic mass, is as follows. Reference [48] analyzes the Breit equation to show how higher-order potentials and the  $p^4$  terms generate  $B_2$ , tracing how the mismatches noted in Eqs. (2.29) and (2.30) propagate to  $B_2$ . This analysis reveals that the discretization error in  $B_2$  is smaller for heavy-light hadrons than for quarkonium states. For heavy-light hadrons, the largest part of the kinetic binding energy comes from the light quarks and gluons, and, since the light quark has mass  $ma \ll 1$ , its contribution to the kinetic binding energy of the meson has only a small discretization error. To tune  $\kappa$  for charmed and bottom quarks, it is therefore better to use heavy-light states, such as  $D_s^{(*)}$  and  $B_s^{(*)}$ , whose kinetic masses have the smallest statistical, discretization, and chiral extrapolation errors. In fact, the leading discretization error, from the chromomagnetic energy, can again be removed by taking the spin-averaged mass of the pseudoscalar and vector mesons.

It is sometimes thought that the tuning inaccuracy of the kinetic binding energy  $B_2$  can be circumvented by adjusting  $\zeta$  so that (a hadron's)  $M_1 = M_2$ , and then fixing  $M_1$  to experiment. But any discretization error in  $B_2$  is then propagated to  $\zeta$  and, hence, throughout the rest of the simulation. It is, therefore, just as clean to leave  $\zeta = 1$  and tune  $M_2$  to a target meson mass, as we have done here.

At this stage, it may be helpful to compare and contrast the Fermilab approach [5,47] with lattice NRQCD [3,4]. The construction of lattice NRQCD starts with the (dimensionally regulated and  $\overline{\text{MS}}$ -renormalized) NRQCD effective Lagrangian for continuum QCD [42,43], and then discretizes it. This process can be repeated order-by-order in perturbation theory. In the Fermilab method, a version of the Wilson-Sheikholeslami-Wohlert lattice action is used, but the results are interpreted with (dimensionally regulated and  $\overline{\text{MS}}$ -renormalized) NRQCD with modified short-distance coefficients. This is possible because Wilson fermions possess heavy-quark symmetry, and the proposed improvements preserve this feature. Then the parallel structure of the NRQCD descriptions of QCD and lattice gauge theory are used to match the latter to the former. In both frameworks, the lattice action can be systematically improved via the nonrelativistic expansion [4,47].

At a practical level, early spectrum calculations [49] use a lattice-NRQCD action [4] that adjusts, at the tree level, the full  $v^4$  Lagrangian and the spin-dependent  $v^6$  Lagrangian.<sup>2</sup> The  $p^4$  terms are, thus, correctly normalized at the tree level, so the quarkonium and heavy-light kinetic mass tunings are comparably accurate. On the other hand,

<sup>2</sup>The HPQCD Collaboration's most recently published unquenched calculations [19] of the bottomonium spectrum with lattice NRQCD are obtained from an action without the spin-dependent  $v^6$  corrections.

the Fermilab action has tree-level errors in the  $v^6$  and even some of the  $v^4$  terms. The errors diminish monotonically as  $a$  is reduced, however. This is especially important for charmonium: here the nonrelativistic expansion is not especially good, but it is needed only to organize the matching of the most important couplings in  $S$ , knowing that further errors, such as those described by  $\mathcal{L}_{\text{HQ}}^{(6)}$ , are of the form  $(mv^2a)^2$  and smaller.

In summary, the pattern of discretization effects leads us to tune  $\kappa$  via kinetic masses corresponding to the  $\overline{1S}$ ,  $D_s$ , and  $B_s$  mesons. The main spectroscopic results, presented in Sec. IV, are for mass splittings, in which case the uncertainties are minimized by quoting differences of our computed rest masses.

## D. Correlator construction

The meson correlator at a given spatial momentum  $\mathbf{p}$  and time  $t$  is defined as

$$C_{ab}(\mathbf{p}, t) = \sum_x e^{-i\mathbf{p}\cdot\mathbf{x}} \langle 0 | O_a(x, t) O_b^\dagger(\mathbf{0}, 0) | 0 \rangle, \quad (2.33)$$

where  $x$  is the spatial coordinate. The source and sink meson operators  $O_b$  and  $O_a$  have the form

$$O_c(x, t) = \sum_y \bar{\psi}(x, t) \Gamma \phi_c(x - y) \psi(y, t), \quad (2.34)$$

where  $\Gamma$  is a product of Dirac matrices appropriate for the meson spin structure, and  $\phi_c(x - y)$  is a smearing function. Neglecting the disconnected piece, the meson correlator can be rewritten with the quark propagators

$$G(x, t; \mathbf{0}, 0) = \int [d\psi][d\bar{\psi}] \psi(x, t) \bar{\psi}(\mathbf{0}, 0) e^{-S}, \quad (2.35)$$

with  $S$  from Eq. (2.6), yielding

$$C_{ab}(\mathbf{p}, t) = \sum_x e^{-i\mathbf{p}\cdot\mathbf{x}} \times \text{Tr}[G(\mathbf{0}, 0; x, t) \Gamma G_{ab}(x, t; \mathbf{0}, 0) \Gamma^\dagger], \quad (2.36)$$

where

$$G_{ab}(x, t; \mathbf{0}, 0) = \sum_{y, z} \phi_a(x - y) G(y, t; z, 0) \phi_b^\dagger(z) \quad (2.37)$$

is the smeared quark propagator.

For the  $P$  states, we use two types of quarkonium correlators, which we call “relativistic” and “nonrelativistic.” In the relativistic case, all four spin components of the quark propagators were used to construct the two-point functions. We used point and smeared sources and sinks. The smearing functions  $\phi_c(x)$  are  $1S$  and  $2S$  wave functions of the QCD-motivated Richardson potential [50]. At the sink, spatial momentum  $\mathbf{p} = 2\pi(n_1, n_2, n_3)/L$  is given to the quarkonium state. We restrict the range of  $\mathbf{p}$  such that  $\sum n_i^2 \leq 9$ . Using this approach, we computed correlation functions for the  $1S$  and  $2S$  states for the pseudoscalar and the vector to study both the kinetic and rest masses. For

the  $1P$  states  $h$ ,  $\chi_0$ , and  $\chi_1$  we computed only the rest masses.

In the nonrelativistic approach to constructing the two-point functions, the meson operators project onto two of the Dirac components of the quark fields. Table II gives the explicit form of these operators. At the source and sink we smear the quark propagators with a  $P$ -type wave function  $\phi_c(\mathbf{r}) = \phi_{1S}(|\mathbf{r}|)\hat{r}_i$ , where  $\phi_{1S}(|\mathbf{r}|)$  is a Richardson  $1S$  wave function [50] and  $i = 1, 2, 3$ . At the origin we set  $\phi_c(\mathbf{0}) = 0$ . The relativistic interpolating operators include extra lower Dirac components that increase the overlap with excited states. Therefore, one should expect that the overlap of the nonrelativistic meson operators with the  $1P$  ground states would be better than in the relativistic case. We used these nonrelativistic operators at  $\mathbf{p} = \mathbf{0}$  for the  $h$ ,  $\chi_0$ ,  $\chi_1$ , and  $\chi_2$  states. In Sec. III A, we compare the results for the first three states with the corresponding results from relativistic operators.

For both correlator constructions, we use several time-slice positions for the source vectors. In the case of the coarse  $\beta = 6.76$ ,  $am_l/am_s = 0.005/0.05$  ensemble and all medium-coarse ensembles, we use eight sources for the relativistic operators; in all other cases, we use four.

### E. Fitting methods

To determine the mass spectrum, we fit our correlator data with a Bayesian procedure, taking priors guided by potential models [50,51]. The priors, listed in Table III, are the same for both relativistic and nonrelativistic correlators. To find the quarkonium masses from relativistic correlators, we use a delta function and a  $1S$  smearing wave function as the source and sink. We fit simultaneously two or three source-sink combinations for the zero-momentum states, including the ground state and up to two excited states. The minimum and maximum source-sink separation is varied, and the best fit is selected based on the confidence level and the size of the errors in the ground state and first excited-state masses. After choosing the fit range, 250 bootstrap samples are generated to provide an error estimate.

The fitting method in the case of nonrelativistic operators is similar except we use the same  $P$ -type wave function, described above, for both source and sink. In this case,

TABLE II. Nonrelativistic meson operators for the  $1P$  states. The smearing operator in spatial direction  $i$  is denoted by  $p_i$ . The indices  $j$  and  $k$  are different from  $i$  and each other, and repeated indices on the last line are not summed over.

Meson	$2S+1L_J$	Irrep.	Operator	
$h$	$^1P_1$	$T_1$	$p_i$ ,	$i = 1, 2, 3$
$\chi_0$	$^3P_0$	$A_1$	$\sum_{i=1}^3 \sigma_i p_i$	
$\chi_1$	$^3P_1$	$T_1$	$\sigma_j \times p_k$ ,	$i = 1, 2, 3$
$\chi_2$	$^3P_2$	$T_2$	$\sigma_j p_k + \sigma_k p_j$ ,	$i = 1, 2, 3$
$\chi_2$	$^3P_2$	$E_2$	$\sigma_j p_j - \sigma_k p_k$ ,	$i = 1, 2$

TABLE III. Prior central values for the ground-state masses. The priors' widths are all fixed to 0.5.

$a$ (fm)	$\kappa$	$M_{q\bar{q}}a$
$\approx 0.18$	0.120	1.932 386
$\approx 0.15$	0.122 0.076	1.841 549 3.818 718
$\approx 0.12$	0.122 0.086	1.539 279 3.187 431
$\approx 0.09$	0.127 0.0923	1.184 840 2.818 421

we use no more than a ground state plus one excited state in the fitting form. The quality of data in the nonrelativistic case is such that often a fit with just the ground state is enough, provided the fitting range is appropriately chosen.

## III. GENERAL RESULTS

Before presenting results for mass splittings (in Sec. IV) we discuss three general issues: a comparison of the statistical quality of relativistic and nonrelativistic operators (Sec. III A); a numerical comparison of tuning  $\kappa$  via  $M_2$  in heavy-light and quarkonium (Sec. III B); and a discussion of how uncertainties from tuning  $\kappa$  are propagated to the mass splitting (Sec. III C).

### A. Relativistic vs nonrelativistic operators

The statistical quality of our data can be judged from Fig. 1, which shows examples of typical two-point functions for the  $1S$  pseudoscalar and vector states and their corresponding effective masses, calculated with relativistic operators. The data are from the coarse ensemble with  $am_l/am_s = 0.01/0.05$ . We have a clear signal and the effective masses have well-established plateaus. As already mentioned, for the  $1P$  states we used both relativistic and nonrelativistic types of operators. Figure 2 compares the effective masses of the  $h_c(1P)$  and  $h_b(1P)$  states, calculated with both types of operators. The data show that the effective masses obtained with nonrelativistic operators plateau at an earlier  $t_{\min}$ . Despite the fact that the statistics in the nonrelativistic case are, in this example, 3 times lower than in the relativistic case, the errors on the fitted  $h_c$  and  $h_b$  masses are smaller than the ones calculated with relativistic operators. This finding holds for all of the  $1P$  states studied here. Our statistics with the nonrelativistic operators are 2–3 times lower than with the relativistic ones (except for the medium-coarse case where they are 6 times lower), yet the errors on the masses are up to 50% smaller, and in some cases smaller still—see Tables IV, V, VI, and VII for numerical comparisons. The nonrelativistic operators couple much more weakly to the excited states and, thus, yield effective mass plateaus of better quality and fitted masses with smaller errors. All of our results for

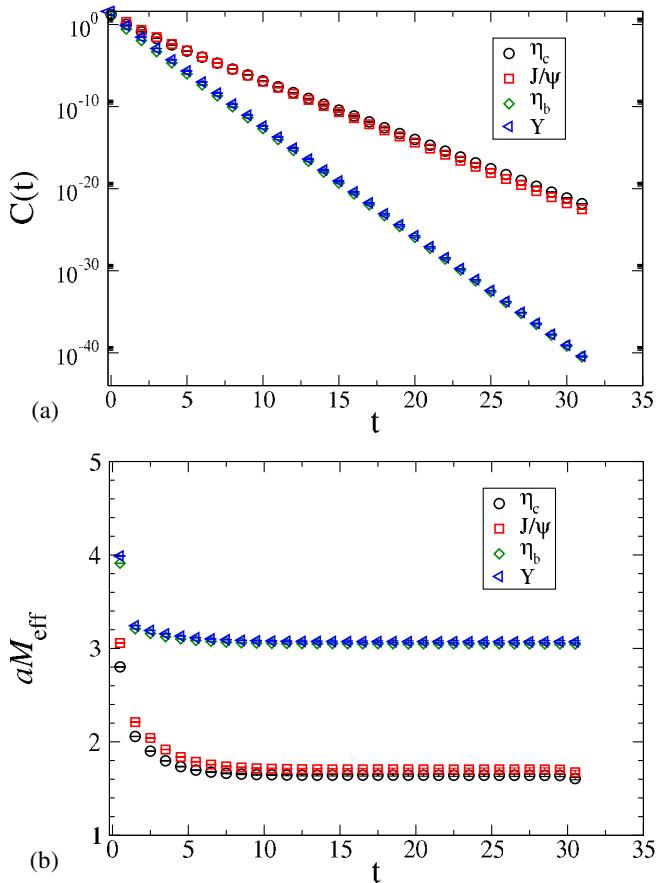


FIG. 1 (color online). Propagators (a) and effective masses (b) for the  $\eta_c$ ,  $J/\psi$ ,  $\eta_b$ , and  $Y$  states, with delta-function sources and sinks, from the coarse ensemble with  $am_l/am_s = 0.01/0.05$ .

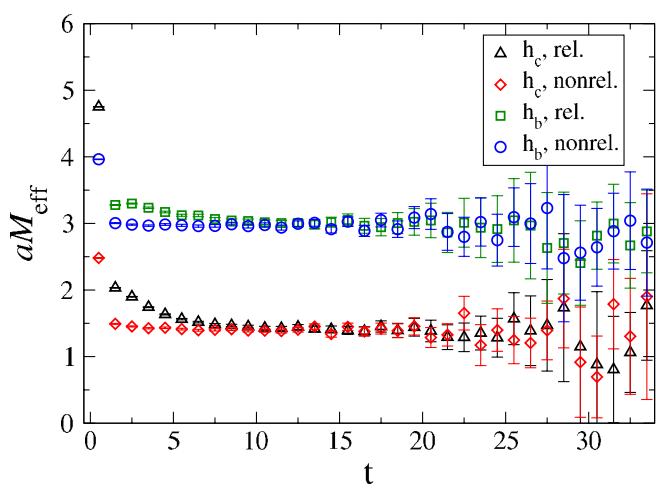


FIG. 2 (color online). Comparison between effective masses for  $h_c$  and  $h_b$  calculated with relativistic and nonrelativistic operators, on the fine ensemble with  $am_l/am_s = 0.0124/0.031$ .

quarkonium masses are listed in Tables IV, V, VI, and VII with statistical errors calculated with the bootstrap method and symmetrized.

The central values of the  $1P$  states calculated with relativistic and nonrelativistic operators occasionally differ by more than 1.5 uncorrelated  $\sigma$ . This difference arises more often than expected, especially once correlations are considered. In the tables, these cases are labeled with a star. To check whether this difference is due to statistics, in some cases we carried out simultaneous fits to both the relativistic and nonrelativistic correlators. The masses extracted this way turned out to be indistinguishable from the masses from nonrelativistic data alone. This was not surprising, because the data from relativistic sources had larger fluctuations than that from nonrelativistic sources. Thus, in our further analysis of the chiral extrapolation and  $a$  dependence, we use the nonrelativistic results for the  $1P$  states wherever they are available.

## B. $\kappa$ tuning in quarkonium and heavy-strange mesons

In Sec. II C, we argued that the best way to tune the hopping parameter  $\kappa$  is to use the spin-averaged kinetic mass of heavy-strange hadrons. If instead one would tune to the kinetic mass of the (spin-averaged) quarkonium ground state, the resulting tuned  $\kappa$  could be different at nonzero lattice spacing. To study this discrepancy we have computed the quarkonium  $\overline{\text{TS}}$  kinetic mass for a wide range of  $\kappa$  on the medium-coarse ensemble with  $am_l/am_s = 0.0290/0.0484$ . Figure 3 shows the results and also shows the physical  $aM(\overline{\text{TS}})$  and  $aM_Y$ .<sup>3</sup> From a polynomial fit to the data, we get  $\kappa_c \approx 0.122$  for the charmed quark, which is the same value as the one we obtain from matching to  $D_s$ . However, because the relevant discretization effects are larger in bottomonium than in  $B$  mesons, the tuned values of the hopping parameter differ substantially:  $\kappa_b \approx 0.094$  from  $Y$  vs  $\kappa_b \approx 0.076$  from  $B_s$ .

When we tune to the  $D_s$ , some uncertainty in  $\kappa$  arises. We take the tuning error in  $\kappa_c$  to be 0.0015 and in  $\kappa_b$  to be 0.006. Reference [9] finds uncertainties (statistical and fitting) in this range on the medium-coarse, coarse, and fine ensembles, and here we assume the same for the extra-coarse ensembles. We discuss in the next subsection how to propagate these errors to our computed splittings.

Above we mentioned a small difference in tuning the clover coupling for the coarse ensembles. The value of the tadpole coefficient  $u_0$  used in that analysis was determined from mean Landau gauge link whereas the coefficient used in the others was determined from the plaquette. This difference means that our bare quark mass, i.e.  $\kappa$ , has a slightly different definition on the coarse ensembles. Discrepancies in mass splittings caused by this choice should be eliminated via the nonperturbative tuning.

<sup>3</sup>When this tuning was carried out, the  $\eta_b$  had not yet been observed by experiment.

TABLE IV. Rest masses of the charmonium states  $\eta_c$ ,  $J/\psi$ ,  $h_c$ ,  $\chi_{c0}$ , and  $\chi_{c1}$  calculated with relativistic operators. All masses in units of  $r_1 = 0.318$  fm. The star denotes masses that differ from their counterparts in Table V by more than  $1.5\sigma$ .

$a$ (fm)	$\beta$	$\kappa_c$	$\eta_c(1^1S_0)$	$\eta_c(2^1S_0)$	$J/\psi(1^3S_1)$	$\psi(2^3S_1)$	$h_c(1^1P_1)$	$\chi_{c0}(1^3P_0)$	$\chi_{c1}(1^3P_1)$
$\approx 0.18$	6.503	0.120	3.2924(9)	4.24(6)	3.4452(16)	4.35(7)	4.185(17)	$\star 4.079(12)$	$\star 4.052(89)$
	6.485	0.120	3.3071(14)	4.42(4)	3.4581(18)	4.48(3)	4.214(26)	4.117(15)	$\star 4.173(13)$
	6.467	0.120	3.3327(7)	4.39(27)	3.4862(11)	4.45(11)	4.213(29)	4.109(12)	4.200(25)
	6.458	0.120	3.3481(13)	4.47(6)	3.5004(16)	4.43(10)	$\star 4.217(18)$	4.106(18)	$\star 4.181(19)$
$\approx 0.15$	6.586	0.122	3.5688(8)	4.66(3)	3.7317(13)	4.75(3)	$\star 4.476(8)$	4.341(6)	4.450(7)
	6.572	0.122	3.5883(9)	4.64(5)	3.7501(14)	4.79(2)	$\star 4.495(8)$	4.368(6)	4.471(17)
$\approx 0.12$	6.81	0.122	3.8721(11)	5.16(4)	4.0594(18)	5.25(3)	4.807(17)	4.626(10)	4.755(19)
	6.79	0.122	3.8876(12)	5.14(3)	4.0747(18)	5.22(4)	4.821(12)	4.657(10)	4.791(10)
	6.76, <i>a</i>	0.122	3.8824(9)	5.09(4)	4.0677(15)	5.10(5)	4.800(13)	4.658(8)	4.758(15)
	6.76, <i>b</i>	0.122	3.9009(8)	5.12(3)	4.0864(11)	5.27(3)	4.817(14)	4.650(30)	4.785(15)
$\approx 0.09$	7.11	0.127	4.2740(26)	5.33(13)	4.4460(22)	5.55(6)	5.159(29)	5.027(16)	5.185(10)
	7.09	0.127	4.2885(15)	5.52(4)	4.4596(15)	5.66(4)	$\star 5.149(24)$	$\star 4.986(15)$	$\star 5.123(19)$
	7.08	0.127	4.2889(26)	5.51(5)	4.4613(33)	5.65(7)	5.167(33)	$\star 4.986(26)$	5.133(48)

TABLE V. Rest masses of the charmonium states  $h_c$ ,  $\chi_{c0}$ ,  $\chi_{c1}$ , and  $\chi_{c2}$  calculated with nonrelativistic operators. All masses in units of  $r_1 = 0.318$  fm. The star denotes masses that differ from their counterparts in Table IV by more than  $1.5\sigma$ .

$a$ (fm)	$\beta$	$\kappa_c$	$h_c(1^1P_1)$	$\chi_{c0}(1^3P_0)$	$\chi_{c1}(1^3P_1)$	$\chi_{c2}(1^3P_2)$
$\approx 0.18$	6.503	0.120	4.213(1)	$\star 4.111(9)$	$\star 4.210(9)$	4.272(15)
	6.485	0.120	4.227(7)	4.105(8)	$\star 4.200(7)$	4.286(10)
	6.467	0.120	4.223(12)	4.127(7)	4.227(8)	4.278(15)
	6.458	0.120	$\star 4.253(9)$	4.128(9)	$\star 4.222(9)$	4.310(11)
$\approx 0.15$	6.600	0.122	4.492(7)	4.344(6)	4.458(7)	4.537(11)
	6.586	0.122	$\star 4.493(7)$	4.349(7)	4.462(7)	4.536(13)
	6.572	0.122	$\star 4.516(9)$	4.375(9)	4.488(9)	4.574(10)
	6.566	0.122	4.548(10)	4.405(6)	4.526(7)	4.614(10)
$\approx 0.09$	7.11	0.127	5.199(11)	5.030(8)	5.170(12)	5.257(12)
	7.09	0.127	$\star 5.198(13)$	$\star 5.034(11)$	$\star 5.168(13)$	5.257(14)
	7.08	0.127	5.178(15)	$\star 5.047(8)$	5.167(13)	5.232(18)

### C. $\kappa$ -tuning uncertainties

Tables IV, V, VI, and VII and most of the plots in Sec. IV show statistical errors only, because the foremost aim of this paper is to understand the pattern of discretization

errors. A systematic error also arises from inaccuracies in tuning  $\kappa_c$  and  $\kappa_b$ , and to study the continuum limit it is necessary to propagate this error to the mass splittings. We discuss here how we treat these uncertainties.

TABLE VI. Rest masses of the bottomonium states  $\eta_b$ ,  $\Upsilon$ ,  $h_b$ ,  $\chi_{b0}$ , and  $\chi_{b1}$  calculated with relativistic operators. All masses in units of  $r_1 = 0.318$  fm. The star denotes masses that differ from their counterparts in Table VII by more than  $1.5\sigma$ .

$a$ (fm)	$\beta$	$\kappa_b$	$\eta_b(1^1S_0)$	$\eta_b(2^1S_0)$	$\Upsilon(1^3S_1)$	$\Upsilon(2^3S_1)$	$h_b(1^1P_1)$	$\chi_{b0}(1^3P_0)$	$\chi_{b1}(1^3P_1)$
$\approx 0.15$	6.586	0.076	7.3776(8)	8.202(5)	7.4100(9)	8.209(5)	8.269(120)	$\star 8.162(35)$	$\star 8.147(40)$
	6.572	0.076	7.4061(9)	8.241(63)	7.4386(9)	8.248(7)	8.321(13)	8.292(11)	8.318(11)
$\approx 0.12$	6.81	0.086	8.0690(10)	8.933(12)	8.1299(12)	8.957(12)	8.919(15)	8.855(13)	8.898(13)
	6.79	0.086	8.0563(17)	8.910(14)	8.1167(19)	8.929(14)	8.902(19)	8.850(21)	8.874(38)
	6.76, <i>a</i>	0.086	7.9815(9)	8.870(11)	8.0426(13)	8.890(10)	8.860(20)	8.796(13)	8.839(14)
$\approx 0.09$	7.11	0.0923	10.2040(15)	11.130(78)	10.2627(19)	11.160(32)	11.050(14)	$\star 11.006(13)$	$\star 11.049(10)$
	7.09	0.0923	10.1861(11)	11.142(20)	10.2468(15)	11.161(18)	$\star 11.056(19)$	10.992(14)	11.034(13)
	7.08	0.0923	10.1795(26)	11.112(30)	10.2397(33)	11.137(56)	$\star 11.066(19)$	$\star 11.017(12)$	$\star 11.048(14)$

TABLE VII. Rest masses of the charmonium states  $h_b$ ,  $\chi_{b0}$ ,  $\chi_{b1}$ , and  $\chi_{b2}$  calculated with nonrelativistic operators. All masses in units of  $r_1 = 0.318$  fm. The star denotes masses that differ from their counterparts in Table VI by more than  $1.5\sigma$ .

$a$ (fm)	$\beta$	$\kappa_b$	$h_b(1^1P_1)$	$\chi_{b0}(1^3P_0)$	$\chi_{b1}(1^3P_1)$	$\chi_{b2}(1^3P_2)$
$\approx 0.15$	6.600	0.076	8.254(8)	8.220(8)	8.243(8)	8.274(9)
	6.586	0.076	8.252(10)	*8.216(10)	*8.242(10)	8.276(10)
	6.572	0.076	8.321(11)	8.288(10)	8.312(11)	8.341(11)
	6.566	0.076	8.369(9)	8.335(9)	8.359(9)	8.391(10)
$\approx 0.09$	7.11	0.0923	11.046(9)	*10.981(10)	*11.022(11)	11.077(8)
	7.09	0.0923	*11.020(10)	10.973(10)	11.006(12)	11.040(10)
	7.08	0.0923	*11.014(10)	*10.964(11)	*11.008(10)	11.045(9)

Several pieces of evidence show that the spin-averaged splittings depend very little on  $\kappa$ . These splittings vary little from charmonium to bottomonium [24], a feature understood to be a consequence of both systems lying between the confining and Coulombic part of the potential [1,2]. This feature is, in fact, reproduced in our lattice-QCD data. Moreover, earlier work in the quenched approximation [52] and with  $n_f = 2$  [8] show negligible  $\kappa$  dependence for spin-averaged splittings. Thus, we shall assume that the  $\kappa$ -tuning error for these splittings can be neglected.

For spin-dependent splittings, we compute the  $1S$  hyperfine splitting as a function of  $\kappa$ , on the medium-coarse ensemble with  $am_l/am_s = 0.0290/0.0484$ , the same ensemble as in Fig. 3. The data are summarized in Table VIII. We fit the data to the form

$$\mu = 1/\kappa - 1/\kappa_{cr}, \quad (3.1)$$

$$HFS = b_0/\mu^2 + b_1/\mu^3 + b_2/\mu^4 + b_3/\mu^5 + b_4/\mu^6, \quad (3.2)$$

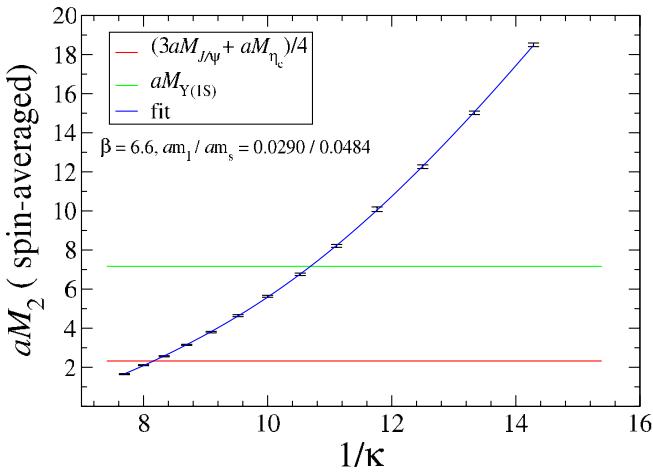


FIG. 3 (color online). Spin-averaged kinetic mass  $aM_2$  as a function of  $\kappa$ , over a wide range, on the medium-coarse ensemble with  $am_l/am_s = 0.0290/0.0484$ . With a polynomial fit to the data, we can read off  $\frac{1}{4}(aM_{\eta_c} + 3aM_{J/\psi})$  and  $aM_Y$ , finding  $\kappa_c \approx 0.122$  and  $\kappa_b \approx 0.094$ .

for  $\kappa_{cr} = 0.145$ , which enforces the requirement that, at large heavy-quark mass  $m_0 = \mu/2a$ , the splitting goes as  $1/m_0^2$ . The fit gives  $\chi^2/\text{dof} = 0.6/8$ , where dof is degrees of freedom. From the fit result we estimate that an error of 0.0015 in the determination of  $\kappa_c$  results in a 6% error in the charmonium hyperfine splitting, and an error of 0.006 in the determination of  $\kappa_b$ , a 22% error in the bottomonium hyperfine splitting. We expect that these errors are characteristic of all splittings driven by the spin-spin and tensor terms in the quarkonium effective potential, since in the nonrelativistic treatment, they all stem from the same term in the heavy-quark effective Lagrangian.

The spin-orbit splitting remains to be considered. In our data and in experiment, it decreases from charmonium to bottomonium similarly to the hyperfine and tensor splittings. Therefore, we shall assume the same relative error from the uncertainty in tuning  $\kappa$ .

Below we also present results for the splittings between twice the spin-averaged mass of  $D_s$  and  $D_s^*$ , and of  $B_s$  and  $B_s^*$ , and the corresponding  $\bar{1}\bar{S}$  quarkonium mass. To estimate their  $\kappa$ -tuning errors we have calculated these spin-averaged masses for several values of  $\kappa$  near  $\kappa_c$  and  $\kappa_b$  on the coarse ensemble with  $am_l/am_s = 0.01/0.05$ . These

TABLE VIII.  $1^3S_1$ - $1^1S_0$  hyperfine splittings in  $r_1$  units as a function of the valence  $\kappa$  calculated for the medium-coarse ensemble with  $am_l/am_s = 0.0290/0.0484$ .

$\kappa$	$r_1[M(1^3S_1) - M(1^1S_0)]$
0.070	0.0247(9)
0.075	0.0299(11)
0.080	0.0369(12)
0.085	0.0442(13)
0.090	0.0531(14)
0.095	0.0631(15)
0.100	0.0749(17)
0.105	0.0885(18)
0.110	0.1029(23)
0.115	0.1244(27)
0.120	0.1499(33)
0.125	0.1836(40)
0.130	0.2335(49)

direct measurements allow us to propagate the  $\kappa$ -tuning errors from the masses to the mass splittings. We obtain an error of 1.3% for charm and 13% for bottom. We assume the same error for these splittings at other lattice spacings.

#### IV. SPECTRUM RESULTS

We now present plots of quarkonium mass splittings as a function of the square of the sea-quark pion mass. The splittings and their errors are calculated using the bootstrap method. In most cases, we expect the dependence on the sea-quark mass to be mild, so we perform on our results a chiral extrapolation linear in  $M_\pi^2$  down to the physical pion. The extrapolated values are denoted in each plot with filled symbols. The error bars come from symmetrizing the  $1\sigma$  (68%) interval of the bootstrap distribution.

Where possible, we compare our results to experimental measurements. As a rule we take the average values from the compilation of the Particle Data Group [24]. The exception is the mass of the  $\eta_b(1S)$  meson, which has only recently been observed. We take  $M_{\eta_b} = 9390.9 \pm 2.8$  MeV, based on our average of two measurements by the *BABAR* Collaboration [53,54] and one by the *CLEO* Collaboration [55].

In examining the results, we are interested in seeing how well we can understand discretization errors via the non-relativistic description of Eqs. (2.13), (2.14), and (2.15). We therefore carry out separate chiral extrapolations at each lattice spacing, and discuss whether the  $a$  dependence, and any deviations from experiment, make sense.

From the effective Lagrangian discussion, we expect different discretization errors to affect spin-averaged and spin-dependent splittings. Errors in the spin-averaged splittings stem from the Darwin ( $D \cdot E$ ) term and the two  $p^4$  terms. Errors in the spin-dependent splittings stem from the chromomagnetic ( $i\sigma \cdot B$ ) and spin-orbit ( $i\sigma \cdot D \times E$ ) terms. Moreover, from the general structure of potentials arising from QCD [25,26], we learn that  $i\sigma \cdot B$  predominantly affects  $M(nS_{\text{HFS}})$  and  $M(nP_{\text{tensor}})$ , while  $i\sigma \cdot D \times E$  affects  $M(nP_{\text{spin-orbit}})$ .

#### A. Spin-averaged splittings

Let us start with  $\overline{1P}-\overline{1S}$  and  $1^1P_1-\overline{1S}$  splittings, plotted in Figs. 4 and 5 vs  $(r_1 M_\pi)^2$ . In the nonrelativistic picture, they arise predominantly at order  $v^2$  via the kinetic energy, which our tuning of  $\kappa$  should normalize correctly. The spin-dependent terms in  $\mathcal{L}_{\text{HQ}}^{(4)}$  [cf. Eq. (2.15)] do not contribute to spin averages ( $\overline{1S}, \overline{1P}$ ) or to a spin singlet ( $1^1P_1$ ). Discretization errors remain, however, at order  $v^4$  via the mismatches in Eqs. (2.28), (2.29), and (2.30). We assess these results using the error estimates in Ref. [47], which account for both the  $a$  dependence and the relative  $v^2$  suppression.

Our results for charmonium are shown in Fig. 4. Our results for both splittings approach the continuum physical point as the lattice spacing decreases, and the size of the discretization effects is about what one expects: 5–6% from  $m_E \neq m_2$  and 3–6% from  $m_4 \neq m_2$  [47].

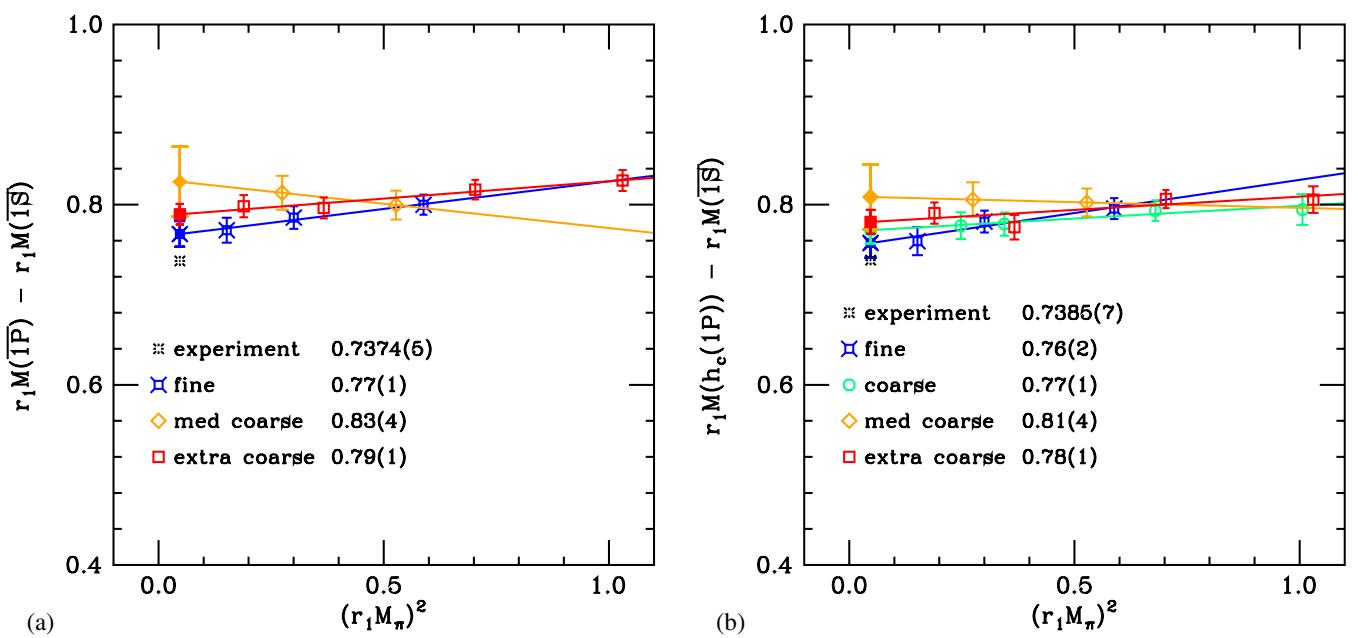


FIG. 4 (color online). The (a)  $\overline{1P}-\overline{1S}$  and (b)  $1^1P_1-\overline{1S}$  splittings in charmonium. The fine ensemble data are in blue fancy squares, the coarse in green circles, the medium-coarse in orange diamonds and the extra-coarse in red squares. The chirally extrapolated values are given in the legend and plotted with filled symbols.

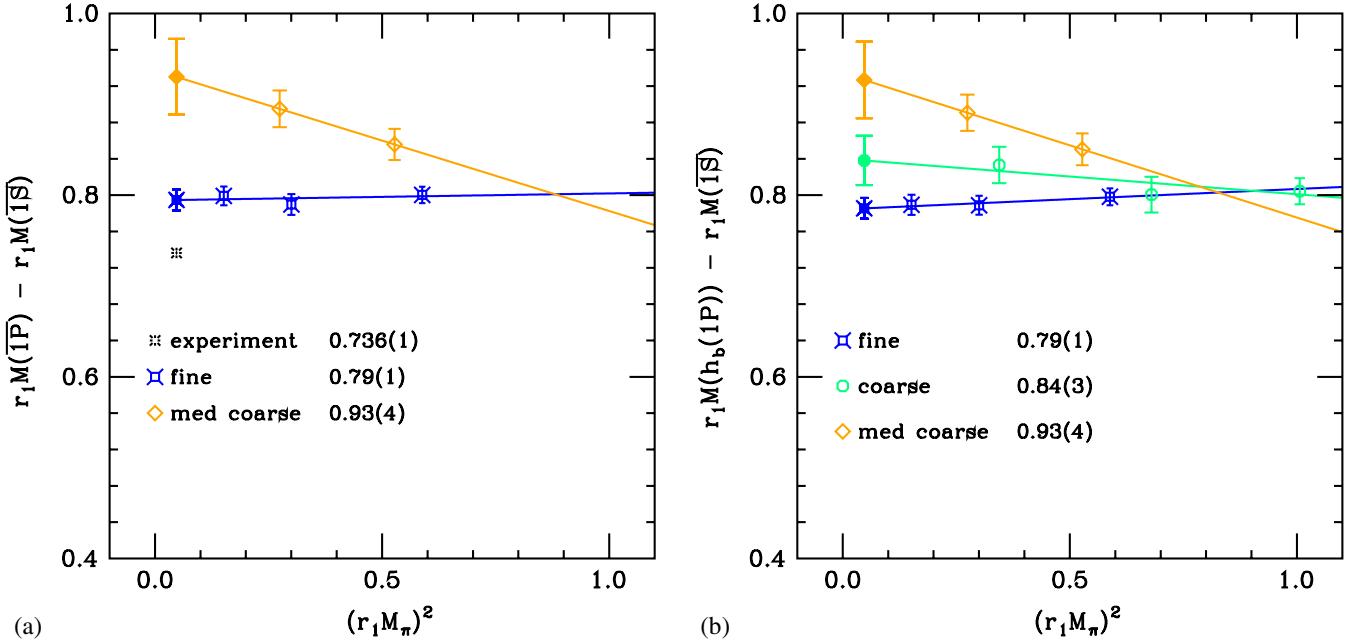


FIG. 5 (color online). The (a)  $\overline{1P}-\overline{1S}$  and (b)  $1^1P_1-\overline{1S}$  splittings in bottomonium. Color and symbol code as in Fig. 4.

The  $\overline{1P}-\overline{1S}$  and  $1^1P_1-\overline{1S}$  splittings in bottomonium are given in Fig. 5. These splittings agree acceptably with experiment, given the estimated discretization errors, 2–3% from  $m_E \neq m_2$  and 2–5% from  $m_4 \neq m_2$  [47]. We cannot compare the  $h_b(1^1P_1)$  mass with experiment, because that state has not been observed [24], but our results for the  $1^1P_1$  level agree very well with the  $\overline{1^3P_J}$  average.

Next let us examine the  $\overline{2S}-\overline{1S}$  splitting. We fit a correlator matrix constructed from two interpolating operators, local and smeared, to three or more states (i.e., two or more excited states). The error we assign to the mass determination estimates the uncertainties in our method. The results for charmonium as a function of  $(r_1 M_\pi)^2$  are shown in Fig. 6(a). The lattice data appear to lie significantly

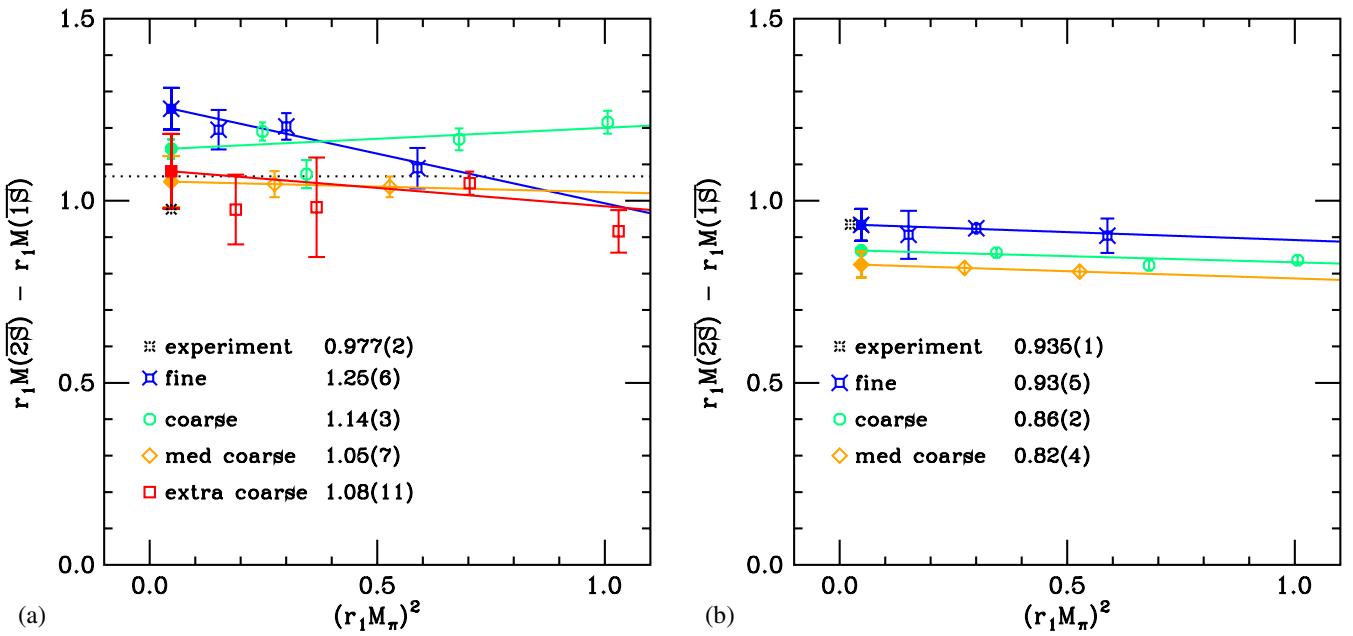
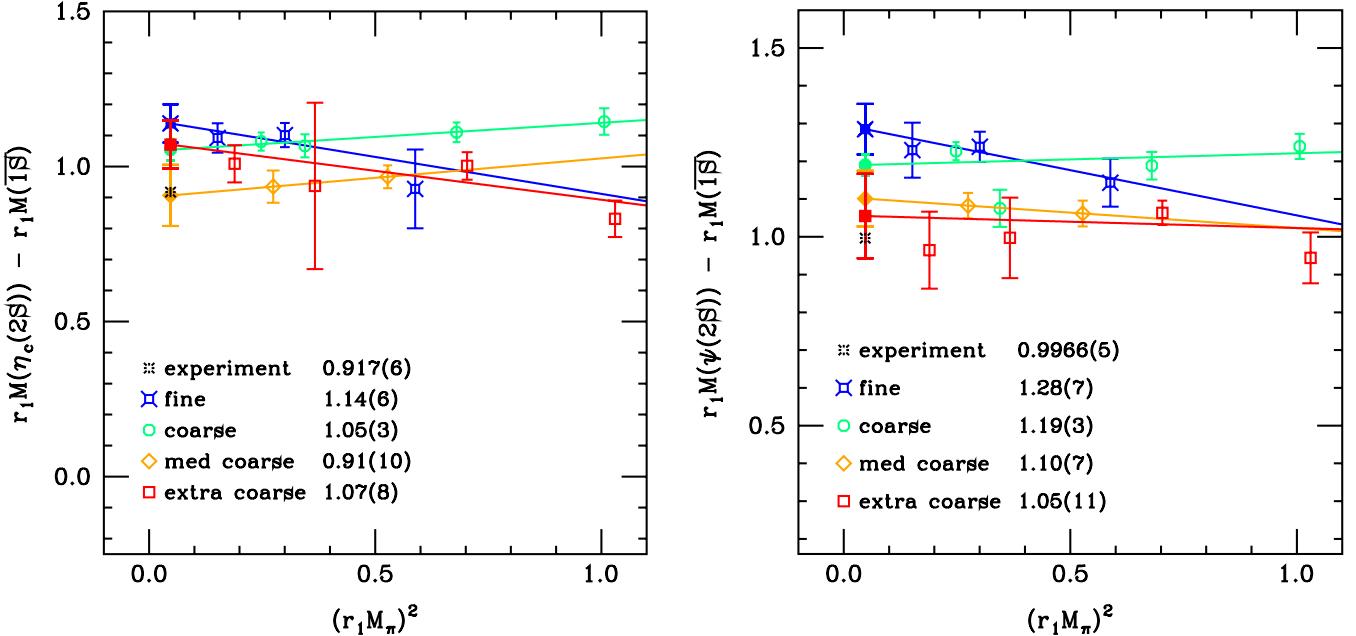


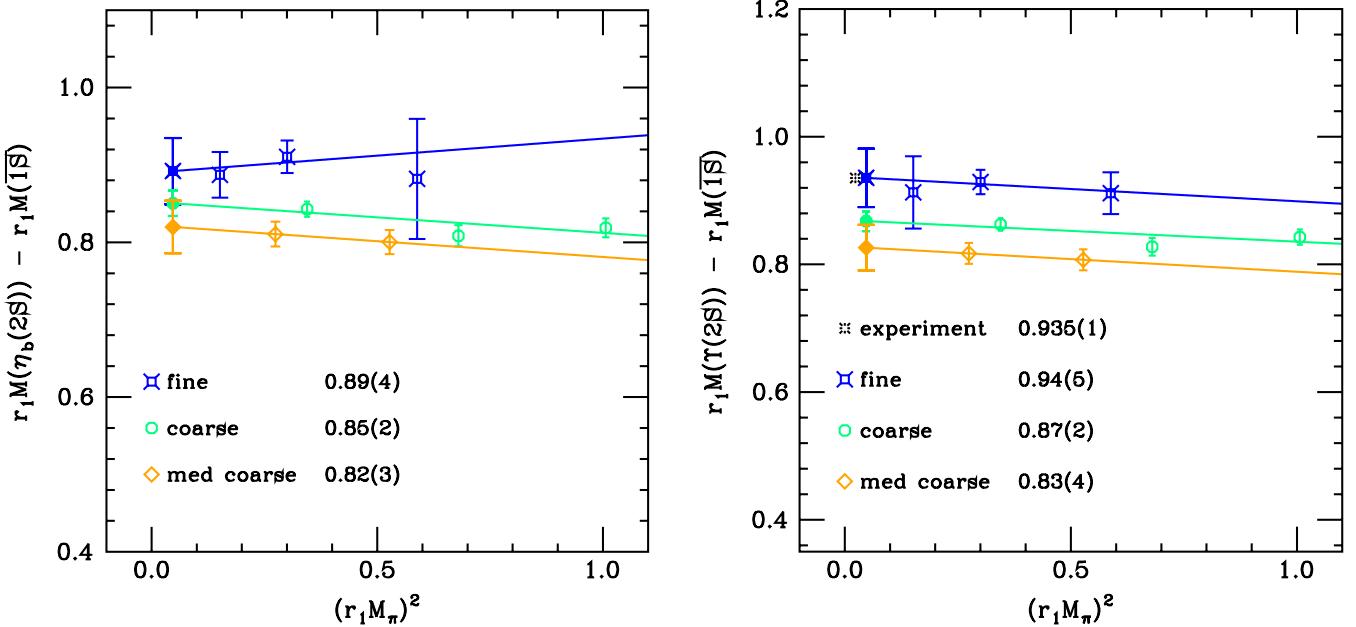
FIG. 6 (color online). Splitting between the  $\overline{2S}$  and  $\overline{1S}$  levels of (a) charmonium and (b) bottomonium. The dotted line in (a) indicates the open-charm threshold. The experimental point in (b) is *not* the spin-averaged splitting, but the  $Y(2S)-\overline{1S}$  mass difference, since the  $\eta'_b$  has not been observed.

FIG. 7 (color online). Splittings in charmonium between the individual  $2S$  states and the  $1\bar{S}$  level.

above the experimental value at the smaller lattice spacings. The individual  $2S$  levels show the same trends we observe in the spin-averaged level. In Fig. 7 we plot separately the  $\eta_c(2S)$ - $1\bar{S}$  and  $\psi(2S)$ - $1\bar{S}$ . We see that both  $\eta_c(2S)$  and  $\psi(2S)$  are responsible for the behavior seen in Fig. 6(a), the latter especially so. The results for bottomonium (Fig. 6(b)) are more satisfactory.

We suggest two possible reasons for the behavior of the charmonium  $2S$ - $1\bar{S}$  splitting results. First, the  $2S$  are the

only excited states in this study. Excited states are more difficult than ground states to determine accurately. With only two operators, our fits are less reliable, even though our fit model has at least three states. Second, the fit procedure does not take into account adequately the possible contribution of multiple open-charm levels. For example, we have not used a two-body operator in the matrix correlator. With unphysically large quark masses, the open-charm levels are unphysically high. As the sea-quark mass

FIG. 8 (color online). Splittings in bottomonium between the individual  $2S$  states and the  $1\bar{S}$  level.

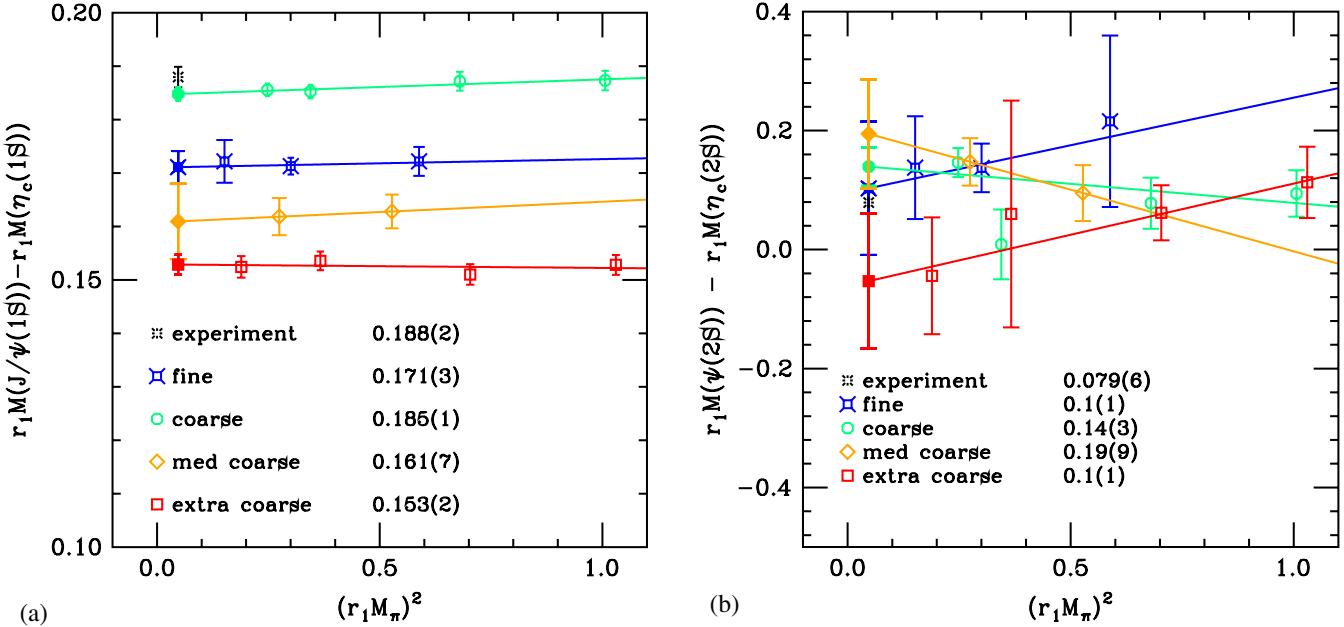


FIG. 9 (color online). Charmonium hyperfine splittings for (a) 1S and (b) 2S.

is decreased, they come down. Moreover, the box size of our lattices at the lightest sea-quark mass is larger, which decreases the discrete level spacing of the would-be open-charm continuum. The dotted line in Fig. 6(a) shows the location of the physical open-charm threshold. It is dangerously close to the physical 2S levels, especially the  $\psi(2S)$ . Thus it is conceivable that nearby multiple open-charm levels are being confused with the 2S and artificially raise its fitted mass. This explanation is consistent with the observed gradual rise of this level in the fine ensembles with decreasing light quark mass but not with the trends seen in the coarse and medium-coarse ensembles.

For bottomonium in Fig. 6(b), the open bottom threshold is safely distant (off scale in this plot), so we do not expect a similar confusion in this channel. Figure 8 shows the individual 2S bottomonium levels separately. There is no comparison for the first excited pseudoscalar state  $\eta_b(2S) - \bar{1}S$ , because the state has not yet been observed [24], although the extrapolated values appear to approach a consistent continuum limit. The first excited vector state splitting  $Y(2S) - \bar{1}S$  is given in Fig. 8(b). The chirally extrapolated values monotonically approach the experimental value and for the fine ensembles our splitting agrees with the experiment.

### B. Hyperfine splittings

Now let us turn to the hyperfine structure. Our results for the hyperfine splitting in charmonium and bottomonium are presented in Figs. 9 and 10. For the 1S levels and for 2S bottomonium, there is little dependence on the sea-quark mass. To assess the approach to the continuum limit one must bear in mind that the errors in Figs. 9 and 10 are

statistical only, and the systematic error from  $\kappa$  tuning must also be taken into account. We thus take the values at the physical pion mass, apply the  $\kappa$ -tuning error and plot these data vs  $a^2$ , as shown in Fig. 11. Both data sets are consistently linear in  $a^2$ , so we carry out such an extrapolation. The extrapolated values in units of  $r_1$  are 0.187(12) for charmonium, with  $\chi^2/\text{dof} = 1.9/2$ , and 0.087(20) for bottomonium, with  $\chi^2/\text{dof} = 0.55/1$ . One can see, from comparing Fig. 11 with Fig. 9 and 10, that the  $\kappa$ -tuning uncertainties inherited from the heavy-strange kinetic mass are larger than the statistical uncertainties of the quarkonium rest-mass splittings.

In physical units these extrapolated results are  $M_{J/\psi(1S)} - M_{\eta_c(1S)} = 116.0 \pm 7.4^{+2.6}_{-0.0}$  MeV and  $M_{Y(1S)} - M_{\eta_b(1S)} = 54.0 \pm 12.4^{+1.2}_{-0.0}$  MeV, where the second error comes from converting from  $r_1$  units to MeV. For charmonium the average of experimental measurements is  $116.4 \pm 1.2$  MeV [24], so our result is perfectly consistent. For bottomonium, the experimental measurements are  $71.4^{+2.3}_{-3.1} \pm 2.7$  MeV [53],  $66.1^{+4.9}_{-4.8} \pm 2.0$  MeV [54], and  $68.5 \pm 6.6 \pm 2.0$  MeV [55]; symmetrizing the error bars and taking a weighted average, we find  $M_{Y(1S)} - M_{\eta_b(1S)} = 69.4 \pm 2.8$  MeV. Our hyperfine splitting thus falls  $1.2\sigma$  short. Note that with lattice NRQCD, the HPQCD Collaboration finds  $M_{Y(1S)} - M_{\eta_b(1S)} = 61 \pm 4 \pm 13$  MeV [19], which agrees with the recent experimental measurements, yet also with our result.

The errors on the final 1S hyperfine splittings quoted here encompass statistics (as amplified by extrapolations),  $\kappa$  tuning, and  $r_1$ . In addition, the coupling  $c_B$  has been adjusted only at the tree level, introducing an error of  $O(\alpha_s a)$  that our continuum extrapolation would not elimi-

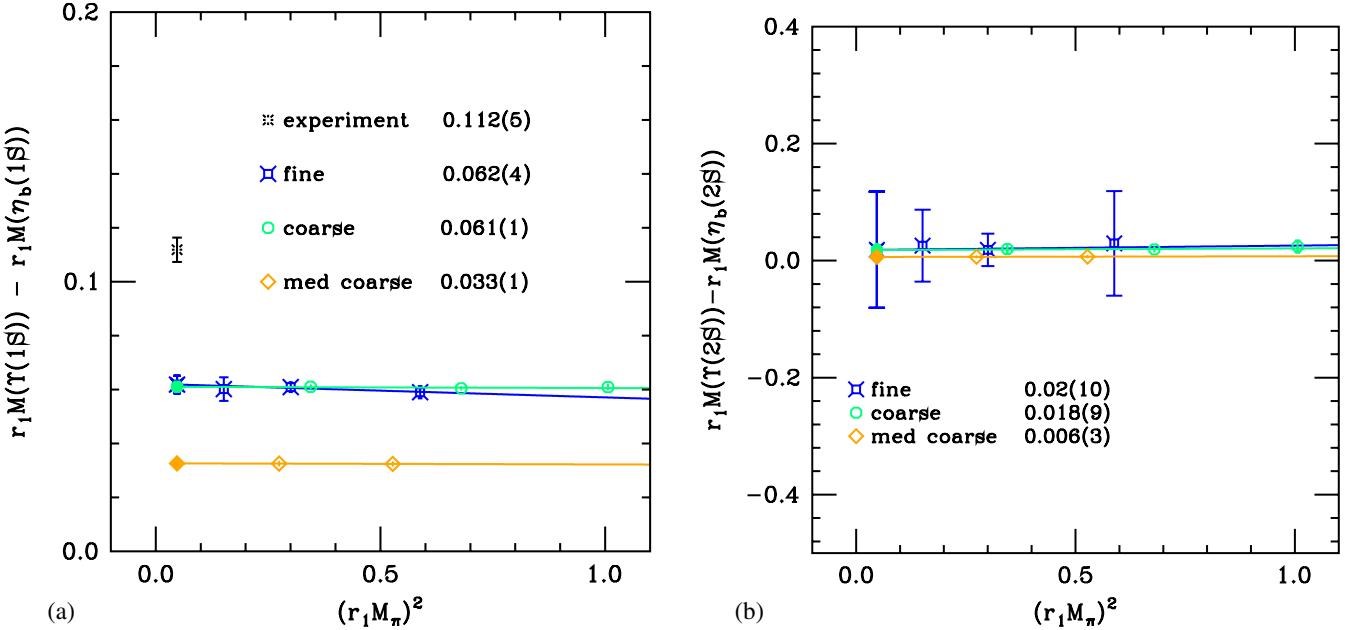
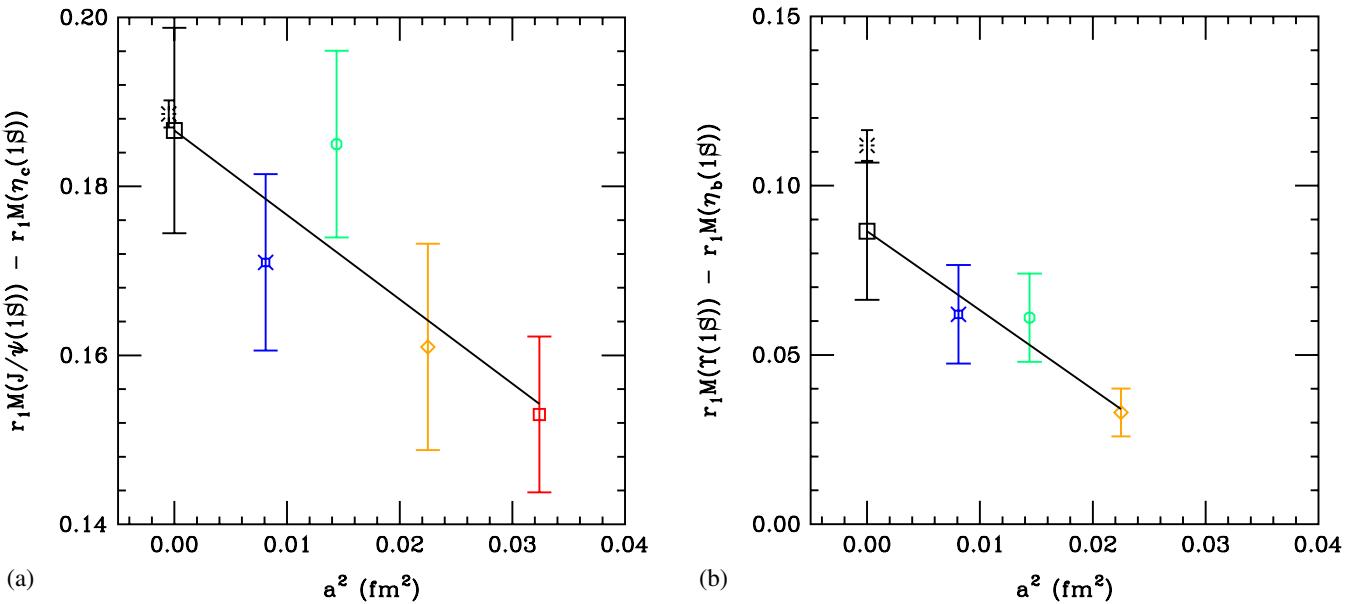


FIG. 10 (color online). Bottomonium hyperfine splittings for (a) 1S and (b) 2S.

rate. A preliminary result for the one-loop correction to  $c_B$  is available [56], suggesting that a very small correction is needed beyond the tadpole improvement of Eq. (2.26), when  $u_0$  is set from the Landau link.

The 2S hyperfine splittings for both charmonium and bottomonium are shown in Figs. 9(b) and 10(b). Unfortunately, these results are not very useful. Although

the charmonium splitting agrees, within large errors, with experiment, one should bear in mind the issue of threshold effects surrounding our determination of the  $\psi(2S)$  mass, discussed above. The bottomonium splitting does not suffer from this problem, but the statistical and fitting errors are still too large to make a prediction of the as yet unobserved  $\eta_b(2S)$  mass.

FIG. 11 (color online). Continuum extrapolations for the 1S hyperfine splittings for (a) charmonium and (b) bottomonium. The symbols and colors of the data points are the same as throughout the paper. Here the error bars on the data points include our estimates for the  $\kappa$ -tuning systematic error. The plotted experimental  $\eta_b$  mass comes from the average of recent measurements [53–55], as discussed in the text.

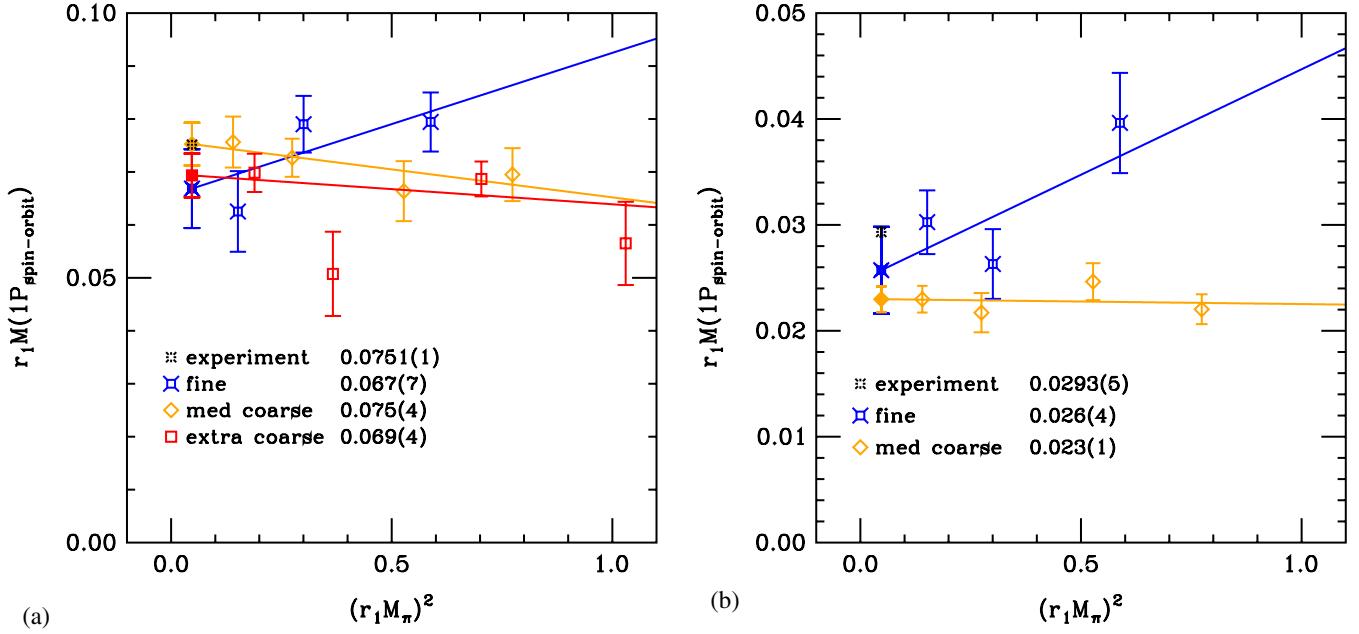


FIG. 12 (color online). Spin-orbit splittings in  $1P$  levels, with  $M(1P_{\text{spin-orbit}})$  defined in Eq. (2.4), for (a) charmonium and (b) bottomonium.

### C. $P$ -state splittings

We now turn to splittings between the  $1^3P_J$  levels, which stem from two contributions [25]. As discussed above, one comes from exchanging a Coulomb gluon between a spin-orbit term,  $\bar{h}^{(\pm)} i\boldsymbol{\sigma} \cdot (\mathbf{D} \times \mathbf{E}) h^{(\pm)}$  in Eq. (2.15), and the static potential,  $\bar{h}^{(\mp)} A_4 h^{(\mp)}$ . The other comes from exchanging a transverse gluon between the chromomagnetic terms,  $\bar{h}^{(+)} i\boldsymbol{\sigma} \cdot \mathbf{B} h^{(+)}$  and  $\bar{h}^{(-)} i\boldsymbol{\sigma} \cdot \mathbf{B} h^{(-)}$ . These two con-

tributions can be separated by forming the combinations in Eqs. (2.4) and (2.5) [26].

The spin-orbit splittings  $M(1P_{\text{spin-orbit}})$  are shown in Fig. 12 for charmonium and bottomonium. They exhibit a small lattice-spacing dependence and agree well with experiment, indicating that the chromoelectric interactions and, hence,  $c_E$  are adjusted accurately enough. The tensor splittings  $M(1P_{\text{tensor}})$  are shown in Fig. 13 for charmonium and bottomonium. These chromomagnetic effects seem to

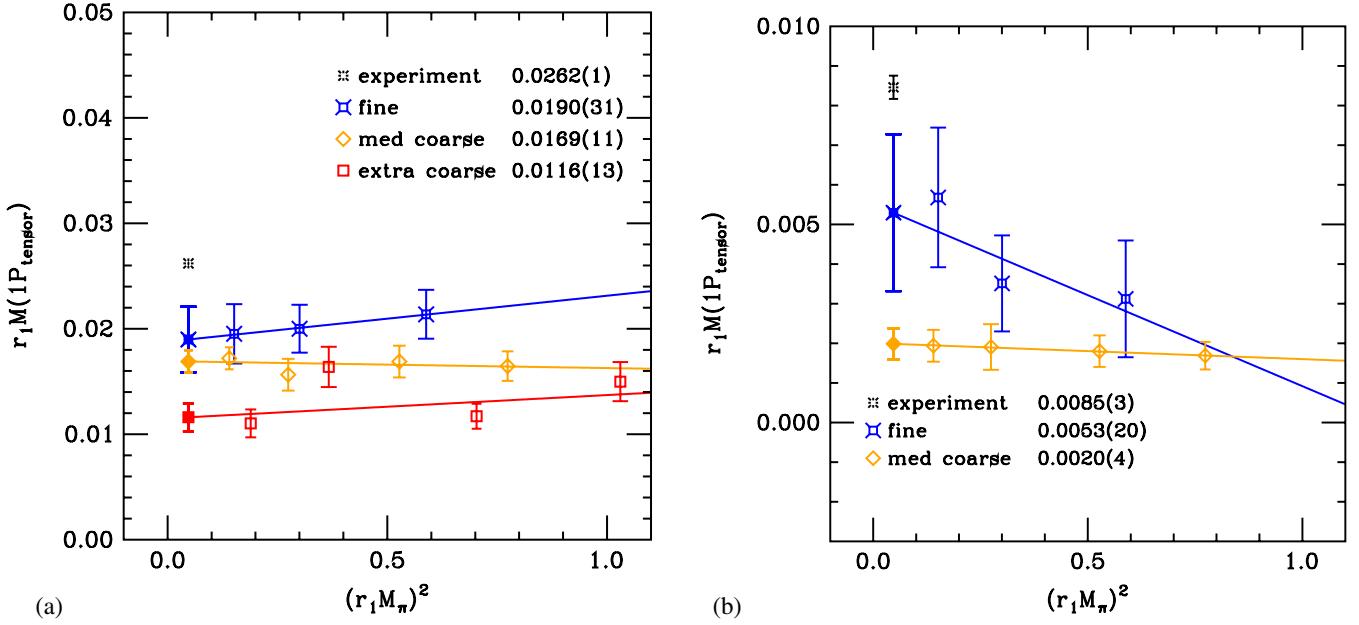


FIG. 13 (color online). Tensor splittings in  $1P$  levels, with  $M(1P_{\text{tensor}})$  defined in Eq. (2.5), for (a) charmonium and (b) bottomonium.

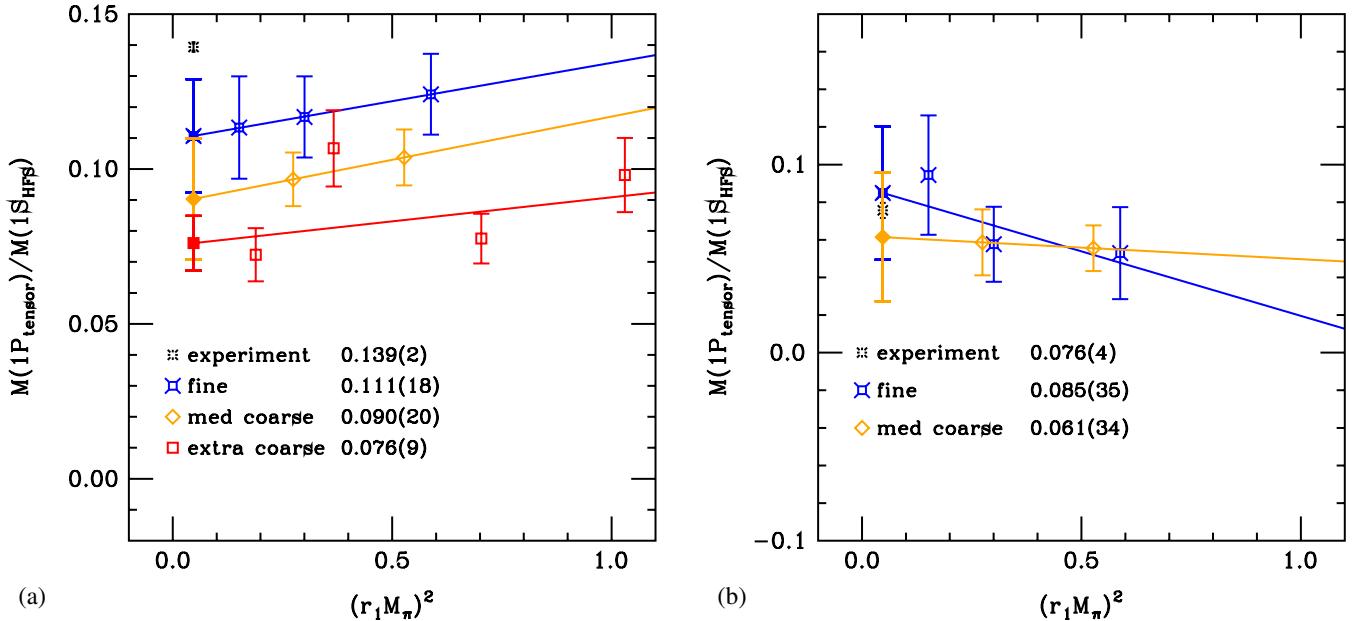


FIG. 14 (color online). Ratio of the 1P tensor and 1S hyperfine splittings, for (a) charmonium and (b) bottomonium.

approach the experimental value as  $a$  decreases. Since the tensor and the spin-spin potential components both measure the effects of the chromomagnetic interaction, we plot the ratio of the 1P tensor splitting to the 1S hyperfine splitting in Fig. 14. The coefficient  $m_B^{-1}$  should drop out from the ratio and if there are no effects from higher-dimension operators, the ratio should be a constant which agrees with the continuum limit. If a higher-dimension operator has a significant contribution, then the ratio need not agree any better than the splittings themselves. The charmonium case, Fig. 14(a), seems to suggest that the

higher-dimension operator matters, the bottomonium case, Fig. 14(b), seems to suggest it does not. This outcome is plausible, because the  $v^2$  suppression of the higher-dimension operator is 10% in bottomonium, but only 30% in charmonium [cf. Eqs. (2.11) and (2.12)].

#### D. Quarkonium vs heavy-strange mesons

Unlike other approaches to heavy quarks, lattice QCD is supposed to treat heavy-light mesons and quarkonium on the same footing. If we form the splitting

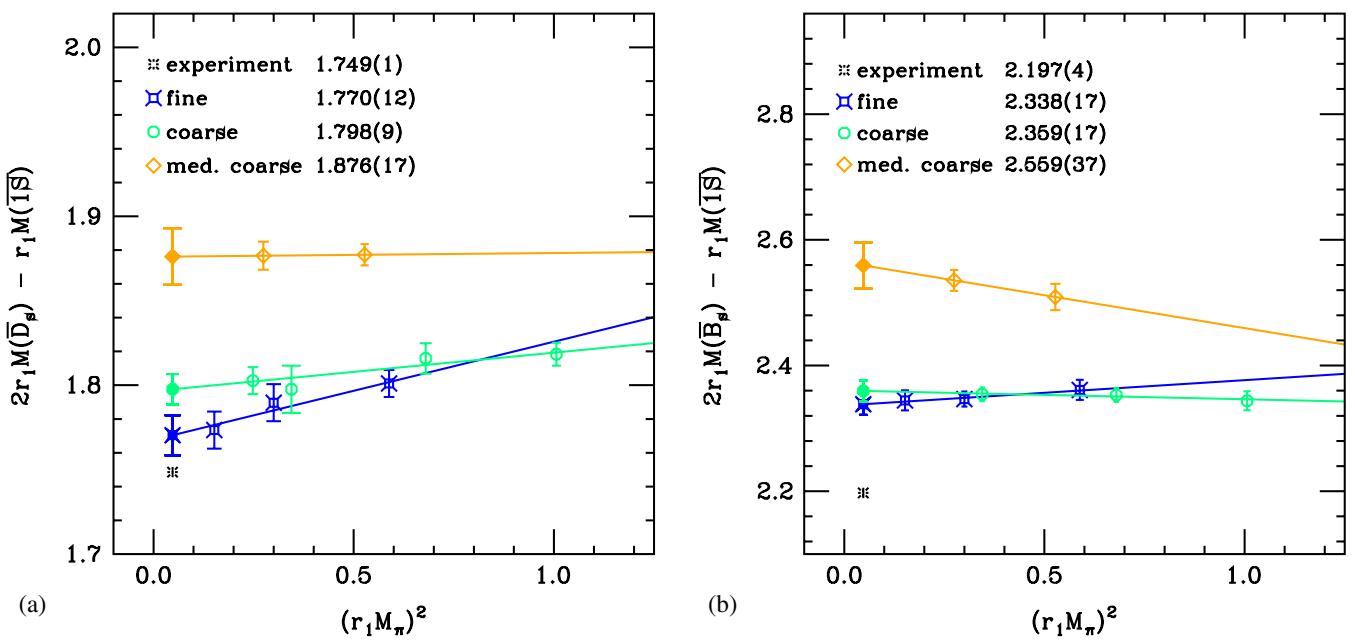
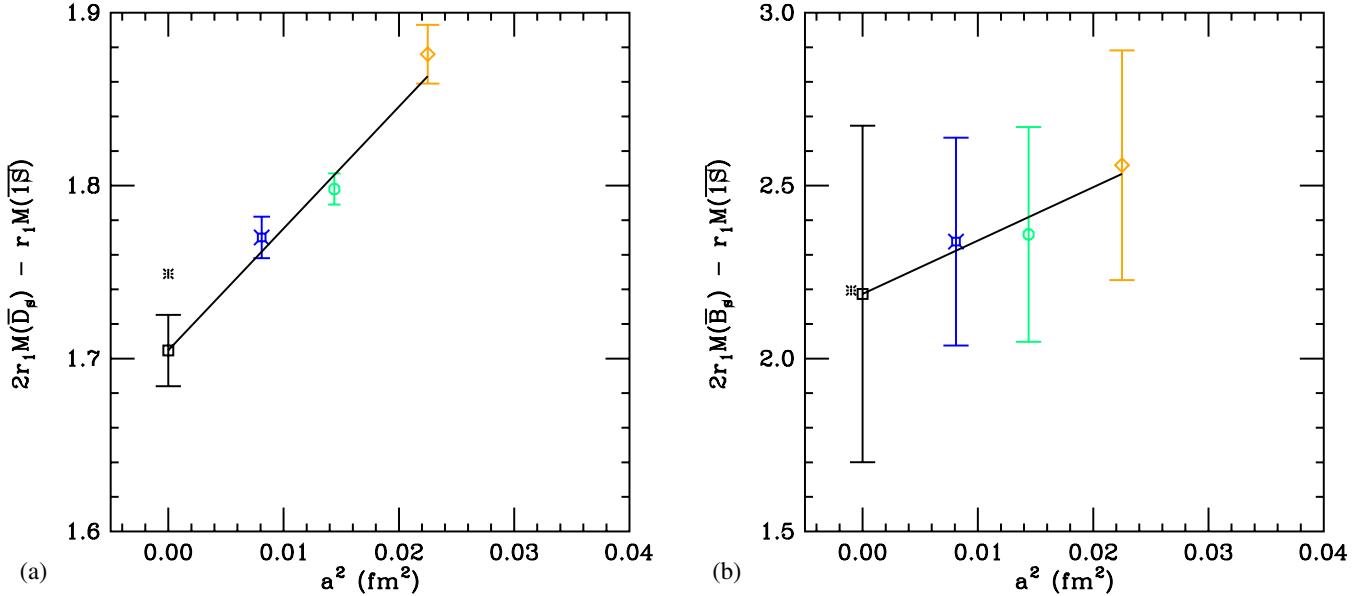


FIG. 15 (color online). Quarkonium–heavy-light splittings (a)  $2M(\bar{D}_s) - M(\bar{1S})$  and (b)  $2M(\bar{B}_s) - M(\bar{1S})$ .

FIG. 16 (color online). Continuum extrapolations of (a)  $2M(\bar{D}_s) - M(\bar{1S})$  and (b)  $2M(\bar{B}_s) - M(\bar{1S})$ .

$$2M(\bar{D}_s) - M(\bar{1S}) \quad (4.1)$$

the rest mass drops out, leaving a pure QCD quantity. Here  $M(\bar{D}_s)$  denotes the spin average of  $D_s$  and  $D_s^*$  masses. This mass difference is interesting from the point of view of the discretization effects, which should contribute less to the  $\bar{D}_s$  and  $\bar{B}_s$  than to the charmonium and bottomonium  $\bar{1S}$  states. We show this splitting (also for the bottom-quark sector) combining our quarkonium rest masses with the Fermilab-MILC heavy-strange rest masses [9] in Fig. 15. The correlation in the error is treated correctly with the bootstrap method and, as elsewhere in this paper, the bootstrap errors are symmetrized. Clearly, discretization effects are important at nonzero  $a$ .

In Fig. 16, we incorporate the  $\kappa$ -tuning errors and show the  $a$  dependence of the above splittings. Carrying out an extrapolation linear in  $a^2$ , which is empirically suitable, we find  $r_1[2M(\bar{D}_s) - M(\bar{1S})] = 1.705 \pm 0.021$  and  $r_1[2M(\bar{B}_s) - M(\bar{1S})] = 2.19 \pm 0.49$ ; these correspond to  $2M(\bar{D}_s) - M(\bar{1S}) = 1058 \pm 13^{+24}_{-0}$  MeV and  $2M(\bar{B}_s) - M(\bar{1S}) = 1359 \pm 304^{+31}_{-0}$  MeV, with the uncertainty in  $r_1$  yielding the second error bar. The bottomonium extrapolation agrees with the experimental value, but the combined statistical and  $\kappa$ -tuning errors are quite large. The charmonium extrapolation is  $1\sigma$  shy of the experimental value. Given the empirical nature of our continuum extrapolation, this is completely satisfactory.

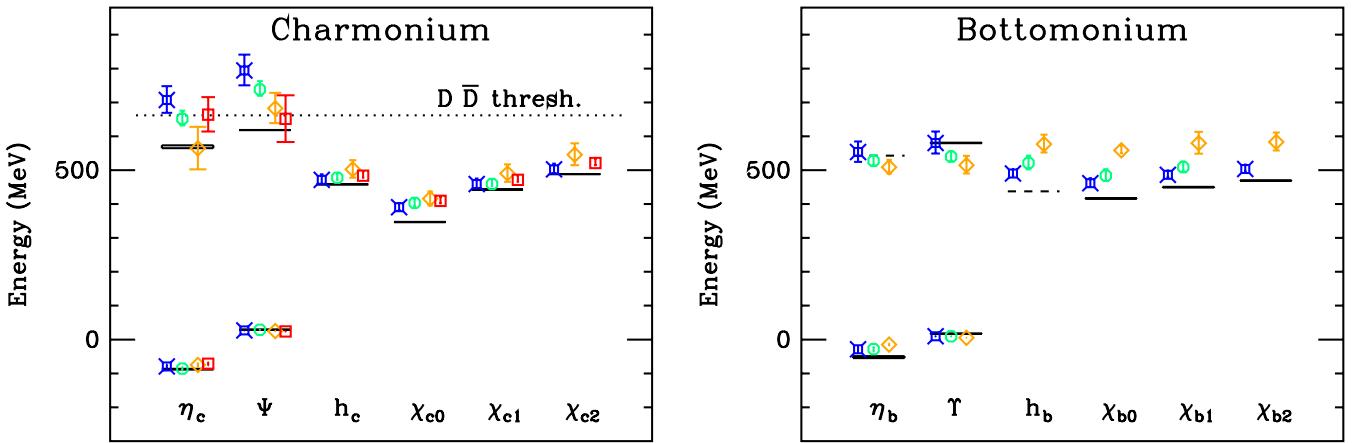
FIG. 17 (color online). Quarkonium spectrum as splittings from the  $\bar{1S}$  level for  $\bar{c}c$  (left) and  $\bar{b}b$  (right). The fine-ensemble results are in blue fancy squares, the coarse in green circles, the medium-coarse in orange diamonds, and the extra-coarse in red squares. Solid lines show the experimental values, and dashed lines estimates from potential models. The dotted line in the left panel indicates the physical open-charm threshold. The error on the data points combines statistical,  $\kappa$ -tuning, and  $r_1$  uncertainties.

TABLE IX. Continuum extrapolations of splittings in charmonium and bottomonium in MeV. The first error comes from statistics and accumulated extrapolation systematics; the second comes from the uncertainty in scale setting with  $r_1 = 0.318^{+0.000}_{-0.007}$  fm.

Splitting	Charmonium		Bottomonium	
	This work	Experiment	This work	Experiment
$1P\bar{1}S$	$473 \pm 12^{+10}_{-0}$	$457.5 \pm 0.3$	$446 \pm 18^{+10}_{-0}$	$456.9 \pm 0.8$
$^1P_1\bar{1}S$	$469 \pm 11^{+10}_{-0}$	$457.9 \pm 0.4$	$440 \pm 17^{+10}_{-0}$	...
$\bar{2}S\bar{1}S$	$792 \pm 42^{+17}_{-0}$	$606 \pm 1$	$599 \pm 36^{+13}_{-0}$	$(580.3 \pm 0.8)^a$
$1^3S_1\bar{1}S_0$	$116.0 \pm 7.4^{+2.6}_{-0}$	$116.4 \pm 1.2$	$54.0 \pm 12.4^{+1.2}_{-0}$	$69.4 \pm 2.8$
$1P$ tensor	$15.0 \pm 2.3^{+0.3}_{-0}$	$16.25 \pm 0.07$	$4.5 \pm 2.2^{+0.1}_{-0}$	$5.25 \pm 0.13$
$1P$ spin-orbit	$43.3 \pm 6.6^{+1.0}_{-0}$	$46.61 \pm 0.09$	$16.9 \pm 7.0^{+0.4}_{-0}$	$18.2 \pm 0.2$
$1S\bar{s}Q\bar{Q}Q$	$1058 \pm 13^{+24}_{-0}$	$1084.8 \pm 0.8$	$1359 \pm 304^{+31}_{-0}$	$1363.3 \pm 2.2$

<sup>a</sup>Y(2S)- $\bar{1}S$  instead of  $\bar{2}S\bar{1}S$ .

### E. Summary of spectrum results

To summarize our results, Fig. 17 shows the charmonium and the bottomonium spectra as splittings from the  $\bar{1}S$  level and compares them to the experimental results. We have plotted the chirally extrapolated values at each lattice spacing and included statistical,  $\kappa$ -tuning, and  $r_1$  uncertainties. Solid lines show the experimental values, where they are known, and dashed lines show estimates from potential models [51] in other cases.

For the splittings discussed above, Table IX shows the continuum limit, taken via linear extrapolations in  $a^2$ . One should bear in mind that the NRQCD-based theory of cutoff effects, explained in Sec. II C, anticipates a less trivial lattice-spacing dependence. The linear-in- $a^2$  extrapolations are consistent with the data, which are not yet sufficient to resolve more complicated functional forms. In Table IX the second (asymmetric) error bar comes from the conversion to MeV with  $r_1 = 0.318^{+0}_{-0.007}$  fm =  $1.611^{+0}_{-0.035}$  GeV $^{-1}$  [33–35].

The charmonium and bottomonium spectra by and large show good agreement with experiment. The charmonium hyperfine splitting agrees very well; the bottomonium splitting agrees at  $1.2\sigma$ . The tensor and spin-orbit splittings also agree well, for both systems. The  $^1P_1\bar{1}S$  and  $\bar{1}P\bar{1}S$  spin-averaged splittings agree at  $1.1\text{--}1.3\sigma$  for  $\bar{c}c$ ; the  $\bar{1}P\bar{1}S$  at  $0.6\sigma$  for  $\bar{b}b$ . As discussed above, the charmonium  $2S$  states are too high, because our operator basis and statistics proved to be insufficient to disentangle the bound states from open-charm threshold effects. For bottomonium the  $\bar{2}S\bar{1}S$  splitting does not suffer from threshold effects and agrees well. When the  $r_1$  uncertainty is included, the splitting of quarkonium relative to the heavy-strange spectrum,  $2M(\bar{D}_s) - M(\bar{1}S)$  and  $2M(\bar{B}_s) - M(\bar{1}S)$ , also agrees well with experiment.

## V. CONCLUSIONS

Quarkonium properties offer an excellent test of lattice QCD, because they are relatively well-understood hadrons, via potential models and effective field theories. This paper attempts a thorough study of the charmonium and botto-

monium mass splittings, using lattice gauge fields with 2 + 1 flavors of sea quarks. By using the Fermilab method for heavy quarks, we are able to study both systems, as well as heavy-light hadrons, with the same basic theoretical tool. By using the MILC ensembles, we are able to study a wide range of lattice spacing, and a wide range of up and down sea-quark masses, down to  $0.10m_s$ .

Our aim here has been to develop methods and to compare discretization effects against expectations that are gleaned from an effective theory analysis. An important technical finding for ground  $P$  states is that nonrelativistic operators are superior to relativistic operators in overlap and, hence, statistics.

Our calculations reproduce most features of the mass splittings, to the extent expected. This optimistic conclusion is marred somewhat, because we find that the errors from  $\kappa$  tuning are significant for spin-dependent splittings. Agreement with experiment is found only when these uncertainties, which stem from the heavy-strange kinetic mass, are taken into account. In some other cases, such as leptonic decay constants for heavy-light mesons [11], uncertainties in  $\kappa$  also influence significantly the final error budget.

In the continuation of this project, we hope to improve on the results presented here in several ways. First, the MILC ensembles now contain approximately 4 times as many configurations, and they extend to smaller lattice spacings,  $a \approx 0.06$  fm and  $a \approx 0.045$  fm. The finer lattice will bring charm into the region where Symanzik-motivated continuum extrapolations are justified and should bring bottomonium discretization effects under 1%. To this end it may also prove worthwhile to incorporate the  $p^4$  corrections of the improved Fermilab action [47]. Higher statistics and twisted-boundary conditions [57] should improve the tuning of  $\kappa$  and, thus, reduce errors from this source as well.

## ACKNOWLEDGMENTS

Computations for this work were carried out on facilities of the USQCD Collaboration, which are funded by the Office of Science of the U.S. Department of Energy. This

work was supported in part by the U.S. Department of Energy under Grants No. DE-FC02-06ER41446 (T.B., C.D., L.L.), No. DE-FG02-91ER40661 (S.G.), No. DE-FG02-91ER40677 (A.X.K.), No. DE-FG02-91ER40628 (E.D.F.); by the National Science Foundation under Grants No. PHY-0555243, No. PHY-0757333, No. PHY-

0703296 (T.B., C.D., L.L.), and No. PHY-0555235 (E.D.F.); and with support from American Physical Society (E.D.F.). Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

- 
- [1] C. Quigg and J. L. Rosner, Phys. Rep. **56**, 167 (1979).
  - [2] W. Kwong, J. L. Rosner, and C. Quigg, Annu. Rev. Nucl. Part. Sci. **37**, 325 (1987).
  - [3] G. P. Lepage and B. A. Thacker, Nucl. Phys. B, Proc. Suppl. **4**, 199 (1988); B. A. Thacker and G. P. Lepage, Phys. Rev. D **43**, 196 (1991).
  - [4] G. P. Lepage, L. Magnea, C. Nakhleh, U. Magnea, and K. Hornbostel, Phys. Rev. D **46**, 4052 (1992).
  - [5] A. X. El-Khadra, A. S. Kronfeld, and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997).
  - [6] C. Aubin *et al.* (MILC Collaboration), Phys. Rev. D **70**, 094505 (2004); C. W. Bernard *et al.* (MILC Collaboration), Phys. Rev. D **64**, 054506 (2001).
  - [7] M. Di Pierro *et al.*, Nucl. Phys. B, Proc. Suppl. **119**, 586 (2003); **129**, 340 (2004); S. Gottlieb *et al.*, Proc. Sci., LAT2005 (2006) 203 [arXiv:hep-lat/0510072]; LAT2006 (2006) 175 [arXiv:0910.0048].
  - [8] A. X. El-Khadra, S. A. Gottlieb, A. S. Kronfeld, P. B. Mackenzie, and J. N. Simone, Nucl. Phys. B, Proc. Suppl. **83**, 283 (2000).
  - [9] C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations) (unpublished).
  - [10] C. Aubin *et al.* (Fermilab Lattice and MILC Collaborations), Phys. Rev. Lett. **94**, 011601 (2005); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), Phys. Rev. D **79**, 014506 (2009); J. A. Bailey *et al.* (Fermilab Lattice and MILC Collaborations), Phys. Rev. D **79**, 054507 (2009); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), Phys. Rev. D **80**, 034026 (2009).
  - [11] C. Aubin *et al.* (Fermilab Lattice and MILC Collaborations), Phys. Rev. Lett. **95**, 122002 (2005); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2008 (2008) 278 [arXiv:0904.1895]; A. Bazavov *et al.* (Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2009 (2009) 249.
  - [12] R. T. Evans *et al.* (Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2006 (2006) 081; LAT2007 (2007) 354 [arXiv:0710.2880]; LAT2008 (2008) 052.
  - [13] G. S. Bali, Phys. Rep. **343**, 1 (2001).
  - [14] J. J. Dudek and R. G. Edwards, Phys. Rev. Lett. **97**, 172001 (2006).
  - [15] J. J. Dudek, R. G. Edwards, and D. G. Richards, Phys. Rev. D **73**, 074507 (2006); J. J. Dudek, R. Edwards, and C. E. Thomas (Hadron Spectrum Collaboration), Phys. Rev. D **79**, 094504 (2009).
  - [16] J. J. Dudek, R. G. Edwards, N. Mathur, and D. G. Richards, Phys. Rev. D **77**, 034501 (2008).
  - [17] S. Meinel, Phys. Rev. D **79**, 094501 (2009).
  - [18] C. T. H. Davies *et al.* (HPQCD, UKQCD, MILC and Fermilab Lattice Collaborations), Phys. Rev. Lett. **92**, 022001 (2004).
  - [19] A. Gray *et al.* (HPQCD Collaboration), Phys. Rev. D **72**, 094507 (2005).
  - [20] E. Follana *et al.* (HPQCD Collaboration and UKQCD Collaboration), Phys. Rev. D **75**, 054502 (2007).
  - [21] C. McNeile, C. T. H. Davies, E. Follana, K. Hornbostel, G. P. Lepage, and J. Shigemitsu (HPQCD Collaboration), arXiv:0910.2921.
  - [22] I. F. Allison, C. T. H. Davies, A. Gray, A. S. Kronfeld, P. B. Mackenzie, and J. N. Simone (HPQCD and Fermilab Lattice Collaborations), Phys. Rev. Lett. **94**, 172001 (2005).
  - [23] E. B. Gregory *et al.* (HPQCD Collaboration), Phys. Rev. Lett. **104**, 022001 (2010).
  - [24] C. Amsler *et al.* (Particle Data Group), Phys. Lett. B **667**, 1 (2008) and update at <http://pdg.lbl.gov/>.
  - [25] E. Eichten and F. Feinberg, Phys. Rev. D **23**, 2724 (1981).
  - [26] M. E. Peskin, Stanford Linear Accelerator Center Report No. SLAC-PUB-3273, 1983 (unpublished).
  - [27] S. A. Gottlieb, W. Liu, D. Toussaint, R. L. Renken, and R. L. Sugar, Phys. Rev. D **35**, 2531 (1987).
  - [28] M. Lüscher and P. Weisz, Phys. Lett. **158B**, 250 (1985); P. Weisz, Nucl. Phys. **B212**, 1 (1983); P. Weisz and R. Wohlert, Nucl. Phys. **B236**, 397 (1984); **B247**, 544(E) (1984); G. Curci, P. Menotti, and G. Paffuti, Phys. Lett. **130B**, 205 (1983); **135B**, 516(E) (1984); M. Lüscher and P. Weisz, Commun. Math. Phys. **97**, 59 (1985); **98**, 433(E) (1985).
  - [29] K. Orginos and D. Toussaint (MILC Collaboration), Phys. Rev. D **59**, 014501 (1998); J. F. Lagaë and D. K. Sinclair, Phys. Rev. D **59**, 014511 (1998); D. Toussaint and K. Orginos (MILC Collaboration), Nucl. Phys. B, Proc. Suppl. **73**, 909 (1999); G. P. Lepage, Phys. Rev. D **59**, 074502 (1999); K. Orginos, R. Sugar, and D. Toussaint, Nucl. Phys. B, Proc. Suppl. **83**, 878 (2000).
  - [30] Z. Hao, G. M. von Hippel, R. R. Horgan, Q. J. Mason, and H. D. Trottier, Phys. Rev. D **76**, 034507 (2007).
  - [31] R. Sommer, Nucl. Phys. **B411**, 839 (1994).
  - [32] C. W. Bernard *et al.*, Phys. Rev. D **62**, 034503 (2000).
  - [33] A. Bazavov *et al.*, arXiv:0903.3598 [Rev. Mod. Phys. (to be published)].
  - [34] A. Bazavov *et al.* (MILC Collaboration), Proc. Sci., CD09 (2009) 007 [arXiv:0910.3618]; LAT2009 (2009) 079

- [arXiv:0910.3618].
- [35] C. T. H. Davies, E. Follana, I. D. Kendall, G. P. Lepage, and C. McNeile (HPQCD Collaboration), arXiv:0910.1229 [Phys. Rev. D (to be published)].
- [36] K. G. Wilson, in *New Phenomena in Subnuclear Physics*, edited by A. Zichichi (Plenum, New York, 1977).
- [37] B. Sheikholeslami and R. Wohlert, Nucl. Phys. **B259**, 572 (1985).
- [38] B. P. G. Mertens, A. S. Kronfeld, and A. X. El-Khadra, Phys. Rev. D **58**, 034505 (1998).
- [39] A. S. Kronfeld, Phys. Rev. D **62**, 014505 (2000).
- [40] J. Harada, S. Hashimoto, K.-I. Ishikawa, A. S. Kronfeld, T. Onogi, and N. Yamada, Phys. Rev. D **65**, 094513 (2002); **71**, 019903 (2005).
- [41] J. Harada, S. Hashimoto, A. S. Kronfeld, and T. Onogi, Phys. Rev. D **65**, 094514 (2002).
- [42] W. E. Caswell and G. P. Lepage, Phys. Lett. **167B**, 437 (1986).
- [43] G. T. Bodwin, E. Braaten, and G. P. Lepage, Phys. Rev. D **46**, R1914 (1992).
- [44] H. W. Lin and N. Christ, Phys. Rev. D **76**, 074506 (2007).
- [45] G. P. Lepage and P. B. Mackenzie, Phys. Rev. D **48**, 2250 (1993).
- [46] N. H. Christ, M. Li, and H. W. Lin, Phys. Rev. D **76**, 074505 (2007).
- [47] M. B. Oktay and A. S. Kronfeld, Phys. Rev. D **78**, 014504 (2008).
- [48] A. S. Kronfeld, Nucl. Phys. B, Proc. Suppl. **53**, 401 (1997).
- [49] C. T. H. Davies, K. Hornbostel, A. Langnau, G. P. Lepage, A. Lindsey, J. Shigemitsu, and J. H. Sloan, Phys. Rev. D **50**, 6963 (1994).
- [50] J. L. Richardson, Phys. Lett. **82B**, 272 (1979).
- [51] W. Buchmüller and S.-H. H. Tye, Phys. Rev. D **24**, 132 (1981).
- [52] A. X. El-Khadra, Nucl. Phys. B, Proc. Suppl. **30**, 449 (1993).
- [53] B. Aubert *et al.* (BABAR Collaboration), Phys. Rev. Lett. **101**, 071801 (2008).
- [54] B. Aubert *et al.* (BABAR Collaboration), Phys. Rev. Lett. **103**, 161801 (2009).
- [55] G. Bonvicini *et al.* (CLEO Collaboration), arXiv:0909.5474.
- [56] M. Nobes and H. Trottier, Proc. Sci., LAT2005 (2006) 209 [arXiv:hep-lat/0509128].
- [57] P. F. Bedaque, Phys. Lett. B **593**, 82 (2004); C. T. Sachrajda and G. Villadoro, Phys. Lett. B **609**, 73 (2005).

## Vis: Online Analysis Tool for Lattice QCD

---

**Massimo Di Pierro\***

*School of Computing - DePaul University - Chicago*  
*E-mail:* [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu)

**Yaoqian Zhong**

*School of Computing - DePaul University - Chicago*  
*E-mail:* [ati\\_zhong@hotmail.com](mailto:ati_zhong@hotmail.com)

**Brian Schinazi**

*School of Computing - DePaul University - Chicago*  
*E-mail:* [schinazi@gmail.com](mailto:schinazi@gmail.com)

Vis is a system that implements Software as a Service for Lattice QCD computations. At its core, it is a repository of gauge configurations accessed via either a web interface or programmatically via a web service. It gives users the ability to upload data and queue computing jobs for background execution. Jobs can be analysis algorithms and/or visualization algorithms (topological charge, polyakov lines, energy density, etc.). It exposes web services which are accessible from a command line script that allows, for example, to upload all gauge configurations in the current folder, request the server to make a plot of the average plaquette, and generate a movie of the topological charge. It interfaces with VisIt (also running server-side) for 3D visualizations and uses matplotlib for 2D plots. Data and results are automatically posted online with a role based access control mechanism. All major tasks can be executed directly from the web interface.

*Lattice 2010*

---

\*Speaker.

## 1. Introduction

In this paper we present a tool for storing and organizing Lattice gauge configurations, for scheduling PBS jobs, including visualization jobs, and for viewing the results of those jobs.

The typical Lattice QCD workflow can be summarized in three steps:

- Lattice gauge configurations are generated, one after the other, in a Markov Chain Monte Carlo. Here we will refer to each chain as a stream.
- An algorithm runs on each gauge configuration in a stream and produces a numerical output (typically a correlation function) which is stored in a file.
- The numerical outputs generated in the previous steps are aggregated and averaged in order to compute a quantity of physical interest (for example masses, lifetimes, or wave functions).

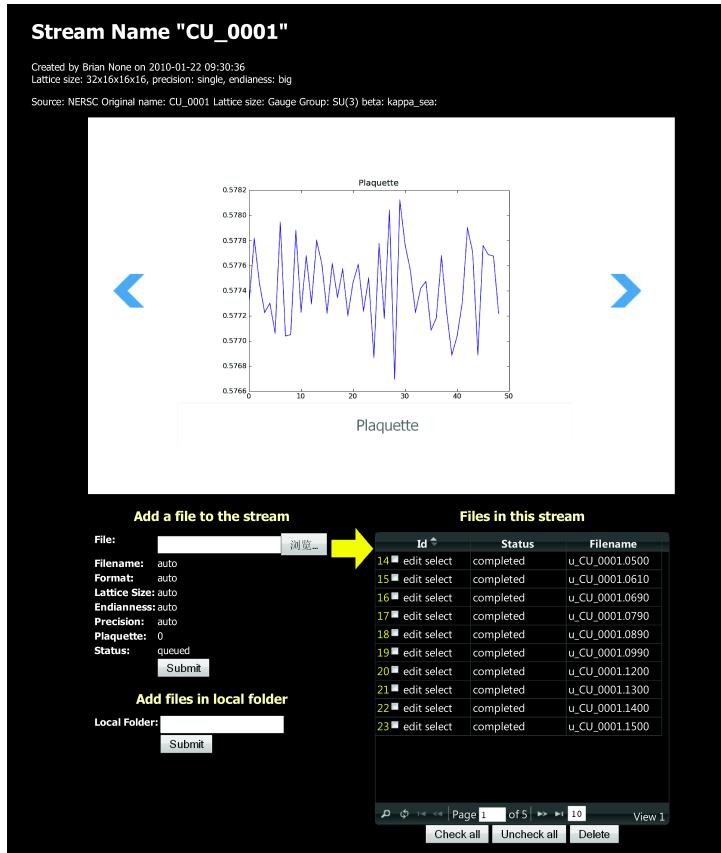
This process is today performed by the larger community in an industrial fashion. Physicists have come together to standardize file formats for storing gauge configurations, for their remote storage and distribution, and for job management and submission. Yet today there are still multiple formats (NERSC [7] 2x3, NERSC 3x3, SciDAC [3], IDLG [6], MILC [2], FermiQCD [2]), with each format having the possibility to have big or small endianness or to contain single or double precision values, there are multiple systems for distribution (scp, gridftp, ILDG), and there are multiple hardware architectures to submit jobs to, which can have different configurations.

This multitude of options creates an entry barrier for young scientists who want to be involved in Lattice QCD research and also constitutes a major expense (in terms of time) for the more senior physicists already in this field. This is particularly true when visualization algorithms are involved. These involve additional file formats, third party dependencies, and domain-specific knowledge.

Vis is prototype software that provides an abstraction layer on top of existing systems that perform the tasks described above. It provides a simple to use web interface to store data that is independent of the file format. It supports importing all of the most common file formats for gauge configurations. Streams can be searched and individual files can be uploaded and downloaded via the web interface. Files can also be uploaded and downloaded using batch scripts that provide file integrity, pause and resume functionality, and security. All users are authenticated and all uploads are signed with the user credentials. Streams can be private to the owner or made publicly available.

Once files are uploaded into a stream, they are automatically converted into a common file format and some standard initial computations are performed on the data: endianness and precision are detected and the average (aggregate and moving) of the plaquette is computed. Metadata about stream is then stored, based on the credentials of the user and the computed parameters, and is available for use for search. These steps require no actions from the user's side.

Once a stream has been uploaded and cataloged, it is made available based on the permissions above. Users can then select a stream and run additional algorithms on it, such as an algorithm that computes the topological charge density. This is one of the major applications of Vis, for which it incorporates some domain specific logic to handle VTK files (for visualization purposes) and interoperates with VisIt [5] for 3D volume plots, iso-surfaces and rendering, and with matplotlib [4] for 2D plots.



**Figure 1:** As an example we downloaded a stream of gauge configurations from NERSC [7] into Vis (CU\_0001). Vis detects the endianness, precision, computes and plots the average plaquette, as represented in the figure (the plot is generated using matplotlib [4]).

With Vis, users can interact with data and schedule jobs without domain specific knowledge and with minimal previous training. This frees the physicist from some of the most pedantic and repetitive tasks. It also allows tracking of data, tracking of progress, and avoids duplication of effort.

## 2. Implementation

Vis is implemented in the Python programming language using the web2py [1] web framework. It uses a Database Abstraction Layer to interface with the database. It includes a collection of algorithms written in C++ using FermiQCD, but it is not limited to FermiQCD. Other binary programs can be registered with the system.

In its simplest installation it comes with its own web server and a file based transaction-safe relational database based on SQL. In a more complex installation, it can use other web servers (for example Apache) and other databases (MySQL, PostgreSQL, Oracle, MSSQL, FireBird, DB2, Informix, and Ingres). All data but the gauge configurations themselves are stored in the database.



**Figure 2:** The figure shows the screen that allows the user to select an algorithm and schedule a job. The right hand side lists submitted jobs and completed ones.

The gauge configurations are stored in binary files which are organized into a structure of nested sub-directories, to avoid having too many files under the same folder. This folder structure can and should be hosted on a different filesystem than the primary database. These files are then referenced by individual configfile records, which are referenced by the table containing the record of the files' stream (fig. 1).

Since configfiles are uploaded by users, even though the system requires authentication, the system must account for exposure to directory traversal attacks. One such attack is performed via filenames that contain special characters not allowed by the file system. Vis implements many security features to prevent this vulnerability, as well as others classified by the Open Web Application Security Project (OWASP), including SQL Injection and Cross Site Scripting.

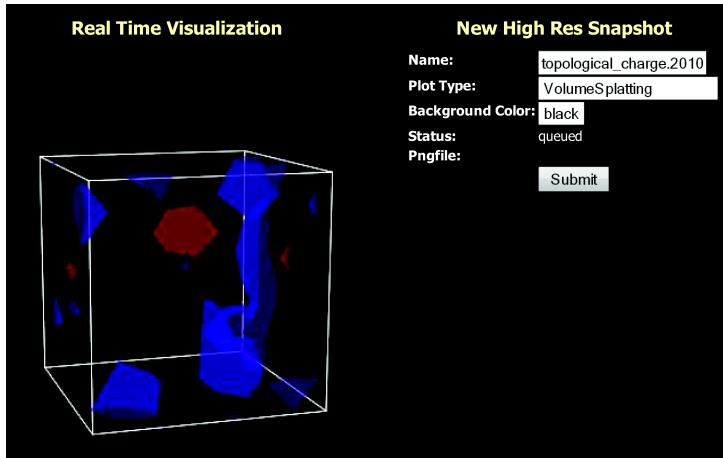
Although streams are created by the web interface, and the files can be uploaded in the same manner, this is not a practical approach and Vis provides an alternative mechanism. Upon creation, each stream is associated with a security token that is a random universally unique identifier (UUID). The owner of the stream can use a provided shell script to easily discover all gauge configuration file from a given local folder and upload them to the server into the stream identified by the UUID. Files have their hashes computed, and are then uploaded one at the time, while showing a progress bar. In case of network error or other upload failure, the script will resume and will use the hash values of the files to determine whether each file has already been successfully uploaded into the stream or whether it has changed locally and needs to be re-uploaded. The script communicates with the server using XML-RPC. An option can be set to have the script also submits jobs to be subsequently run against the data being uploaded.

Progress can be monitored both locally and via the web interface.

Algorithms are submitted via the Portable Batch System (PBS). Vis implements two queues, one that lists algorithms to be submitted per stream and a queue that maps the PBS queue. Vis does not immediately submit all jobs to PBS but monitors the PBS jobs to keep the workload limited and constant (fig. 2).

At the time of writing, only a few algorithms are supported and adding new algorithms requires editing of the Vis source code. Planned improvements include the ability for the user to register any third party program with Vis by providing a startup and configuration script.

Visualization algorithms are somewhat special because they consist of three steps. In the first step, data is analyzed and projected into one or more 3D scalar fields, which are then saved in VTK files. In the second step, the user interacts with the VTK file (this requires a crude but fast visual representation of the data). In the third step the data is visualized at high resolution, for purposes such as printing or high-quality display.



**Figure 3:** When a job is completed, the user can interact with the output (the VTK files) using the 3D JavaScript Widget shown in the figure. The user can submit a request for high resolution rendering of the data (for example using the volumetric splatting algorithm on black background).

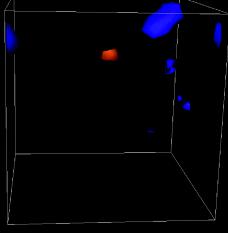
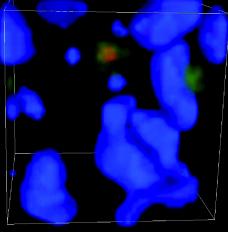
Vis includes workflow and tools that distinguish these steps. Step one consists of a regular Lattice QCD algorithm and it is treated as any other algorithm. Step two is the most complex one to handle via a web interface. For this purpose, we created a 3D JavaScript widget that computes iso-surfaces and allows real time rotation (fig. 3) with the need to install a visualization program on the client-side. Once the user has identified an optimal point of view for visualization of the data, step three can be performed by selecting a high resolution volume or iso-surfaces plot and clicking a button.

Following submission, a VisIt job runs in the background and produces the specified image. A multiple choice menu allows users to pick settings for VisIt (such as colors, axes formatting, etc). There is no need to interact with the visualization engine directly (fig.4).

It is also possible to schedule visualizations of every gauge configuration in a stream and have Vis assemble the resulting images into a video using the ffmpeg open source multimedia tool.

### 3. Conclusions

Although Vis is still a work in progress, it constitutes one more step toward the ideal goal of the authors to provide a complete QCD Software as a Service platform. Today only the step of generating gauge configurations is performed efficiently in a semi-industrial way. This is primarily because this step is the most computationally expensive and there are few reliably-tested programs available to generate gauge configurations. Yet many other tasks performed by lattice QCD physicists can be automated. This presents many benefits: it reduces the margin of error, provides better optimization of resources, allows better tracking of data and work progress, and, most importantly, it frees the time of scientists for more intellectual activities such as developing new models, new algorithms, and tackling new problems.

<b>Id</b>	<b>Status</b>	<b>Pngfile</b>
293	edit select completed	
291	edit select completed	

Page 1 of 1 | View 1 - 3 of 3 | Check all | Uncheck all | Delete

**Figure 4:** This screen shows a list of visualizations scheduled for the current VTK file. If a visualization is completed, a thumbnail of the image is displayed. The image can be downloaded by clicking on the thumbnail.

At this point Vis is a stand-alone web application but it can be integrated with existing ILDG tools to provide a unified interface to Lattice QCD computations.

In the near future we are planning a better integration with FermiQCD to provide a wider choice of algorithms; a better customization to allow support for algorithms written with different Lattice QCD libraries; a more sophisticated workflow management system to handle complex conditional dependencies; and integration with mc4qcd (a web based plotting and analysis tool also written by the authors).

The hardware currently dedicated to VIS is very limited both in terms of speed and space. Dedication of better hardware to this system will enable us to provide QCD as a service to interested users.

The current version of Vis can be downloaded from:

<https://launchpad.net/qcdvis>

### Acknowledgements

This project was funded by the Department of Energy, grant DEFC02-06ER41441.

## References

- [1] Massimo Di Pierro. web2py website, 2010.
- [2] Massimo Di Pierro and Jonthan M. Flynn. Lattice QFT with FermiQCD. *PoS*, LAT2005:104, 2006.
- [3] Robert G. Edwards. Nuclear physics using lattice QCD in the SciDAC era, 2009.
- [4] John Hunter, Darren Dale, and Michael Droettboom. matplotlib website, 2010.
- [5] Lawrence Livermore National Security, LLC. Visit website, 2010.
- [6] C. Maynard. International Lattice Data Grid: Turn on, plug in, and download. In *Symposium on Lattice Field Theory*, 2009.
- [7] U.S. DOE. Nersc website, 2010.

Werner Benger,  
Andreas Gerndt,  
Simon Su,  
Wolfram Schoor,  
Michael Koppitz,  
Wolfgang Kapferer,  
Hans-Peter Bischof,  
and Massimo Di Pierro

## Proceedings of the 6th High-End Visualization Workshop

Open issues in visualization  
with special concentration on applications in  
astrophysics, numerical relativity, computational fluid dynamics  
and high-performance computing

December 8<sup>th</sup>- 12<sup>th</sup>, 2010  
Obergurgl, Tyrol, Austria

<http://vizworkshop.cct.lsu.edu/viz2010/>

---

## Article 2

# Visualization Workflow for Lattice QCD

Brian Schinazi, Yaoqian Zhong, Massimo Di Pierro

School of Computing, College of Computing and Digital Media, DePaul University  
243 S Wabash Avenue, Chicago, IL, USA

Vis is a web based application that implements Software as a Service for Lattice QCD computations. Lattice QCD is a numerical approach to the mathematical model that describes quarks and gluons, the constituents of protons, neutrons and many other composite particles. Lattice QCD computations are implemented as Markov Chain Monte Carlo. Vis allows to store this data, explore it, schedule computing jobs using a local or remote PBS cluster, and schedule visualization jobs using VisIt as back-end. All the major operations of Vis can be performed via the web-based interface as well as scripted. Vis provides an access control mechanism and strong security features. It is a single platform that may allow physicists to collaborate better by sharing their data online.

### 2.1 Introduction

Lattice QCD [Massimo Di Pierro, 2006] is a numerical approach to the study of quarks, the elementary constituents of protons, neutrons and other forms of matter. In 1968, the study of physics reached a turning point when the structure and interactions of all known particles were described by a single mathematical expression, known as the Standard Model Lagrangian [Novaes, 1999]. Since that time, predictions based on the Standard Model have been extremely accurate, and have been able to account for the results of every high energy physics experiment.

## 2.1. INTRODUCTION

---

Still, physicists continue to explore nature at smaller and smaller scales and to look for a breakdown of the model, manifested as a discrepancy between predictions and experiments. This would be a major discovery.

The part of the Standard Model that describes quarks specifically is called Quantum Chromodynamics [Altarelli, 2002]. This constitutes perhaps the most fascinating part of the Standard Model, as quarks are the only elementary particles subject to the strong nuclear force – a highly non-linear interaction that allow quarks to bind together in complex structures. Practically all of the composite particles we see in experiments are made up of quarks.

The goal of Lattice QCD is twofold: to compute from first principles the properties (such as masses and lifetimes) of these composite particles, and to extract fundamental parameters of QCD (such as particles' masses) from a comparison of theory with experiment.

Typical computations consist of taking a small portion of space ( $10^{-15} m$  of side) and its evolution over a short period of time ( $10^{-23} s$ ), and then performing a Markov Chain Monte Carlo (MCMC) simulation of all of its possible evolutions (1000 histories). We call the data saved at each MCMC step a gauge configuration. Next, correlation functions are measured over all simulated evolutions of the system. It can be proven that such an algorithm is equivalent to simulating a quantum-mechanical system. Finally, observable quantities are extracted from the correlation functions.

Until recently, visualization techniques have not been used in the study of QCD. The main reason is that the objects being computed have no obvious correspondence with physical 3D objects. The content of the portion of space which is simulated contains purely random data, since each data set is just a step of a MCMC. The physics is encoded in the probability distribution used to generate the MCMC, and not in the data itself.

We believe that there are some useful applications of visualization techniques to Lattice QCD: they can be used for didactic purposes, they can be used to better understand the behavior of the MCMC algorithms, and they can be used to detect certain types of error in the computation.

One type of error we are interested in is a systematic one: the possible long auto-correlation of topological charge distribution. The portion of space-time that is simulated contains a field that can be thought of as the chromo-electro-magnetic field of gluons, the particles that mediate interactions between quarks (analogous to the electro-magnetic field being represented by photons, but in this case having three types of charges). One can also associate a local topological charge density to the field. It should be noted that these fields all live in 4D and therefore must be projected on to 3D in order to be visualized.

The elementary steps of Lattice QCD algorithms are local and, for typical

## 2.1. INTRODUCTION

---

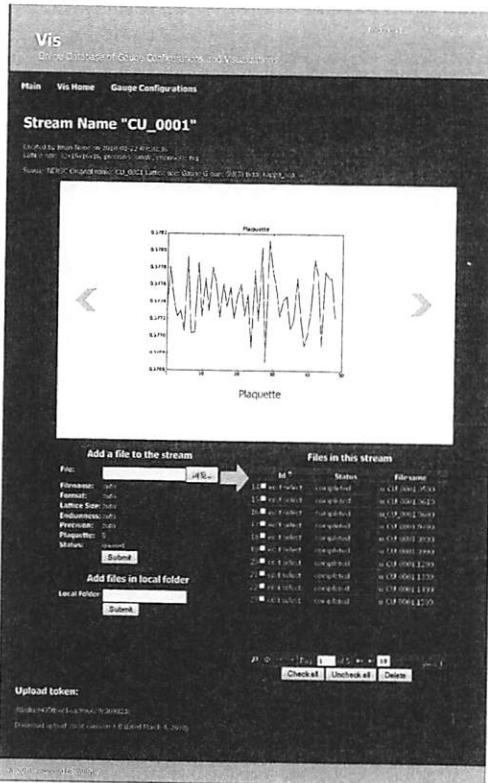


Figure 2.1: Screenshot of the stream view, including the MC history for the average plaquette and a list of files in the stream.



Figure 2.2: Screenshot of the processing view, showing information about the processing of files in a stream, including the status of the computations and the algorithm that was used.

production computations, they do not significantly change the total topological charge. The question, therefore, is whether or not they change the local topological charge density. If they do not, then the computation is biased because the MCMC must get stuck in a topological sector and is not sampling properly. Visualization techniques can be applied in this case, because we have, for example, been able to use them to show that the answer is yes – typical production computations are in fact not stuck in a topological sector.

Our goal is to automate the workflow of physicists working with this data and allow them to:

- store and share gauge configuration for multiple MCMC streams.
- schedule computing jobs for each stream, in particular the computation of the topological charge density.
- visualize the topological charge density (and other derived fields) using iso-surfaces and/or volume plots.
- interact with the data using a web interface (rotate the topological charge and change visualization parameters).

## 2.2 Implementation

Vis, at its core, is a web application for storing collections of datasets and scheduling computations on the files in the sets. A set here is a MCMC stream, and the files in the set are the gauge configurations. Computations can be numerical algorithms and/or visualizations. The computations are submitted via an available Portable Batch System installation and can be parallel jobs.

Users can create an account in the system, login, and perform operations such as creating a new stream, uploading files into the stream, scheduling computations, searching and downloading streams submitted by other users, and viewing the results of computations performed on streams already in the system.

Some computations are scheduled automatically when data is uploaded, because new files need to be explored in order to detect their structure, they must be converted to a standard format, and then analyzed to extract some basic physical parameters that are important for cataloging the file and detecting possible errors.

Individual files can be very large (100M-1GB each) and therefore it may not be practical to upload them via the web interface, which does not support pausing and resuming. To avoid this problem, the system provides an alternate upload mechanism. When a new stream is created, a security token is issued to the user. The user can utilize a provided program to automatically upload every file from a local folder, authenticating via the downloaded token.

## 2.2. IMPLEMENTATION

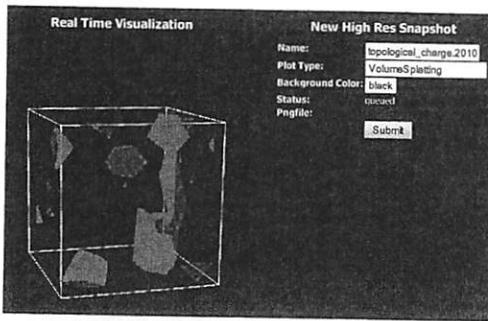


Figure 2.3: Screenshot showing widget that allows limited manipulation of visualized datasets.

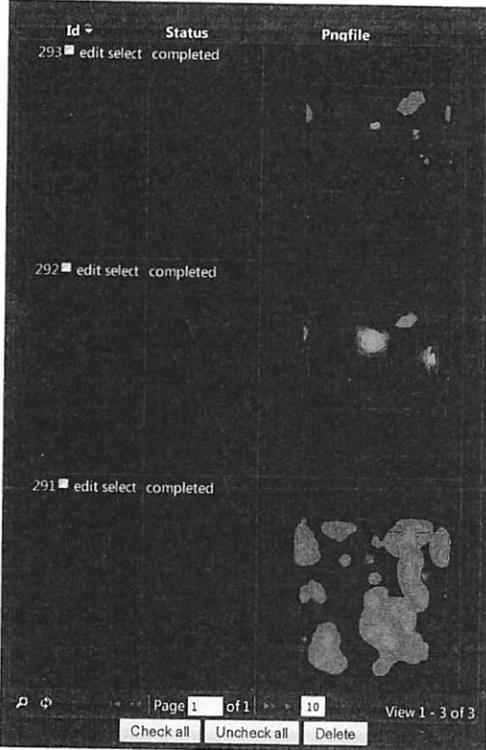


Figure 2.4: Using the browser-based interface, the user can set parameters for the visualization and then submit the job for processing.

### 2.3. SUMMARY

The algorithm that computes the topological charge density generates output in VTK format. These files can, to a limited extent, be manipulated via the browser using a JavaScript widget that displays isosurfaces at 60% of the dataset's minimum and maximum values.

The user can choose a visualization angle, specify additional parameters, and then schedule a full visualization job to generate a high-res image.

The web interface was built using web2py [Di Pierro, 2010]. We utilize matplotlib [Hunter et al., 2010] for 2D plotting and VisIt for 3D visualization of volume plots and iso-surfaces, although we are exploring the possibility of moving to Vish [Benger, 2010] for the latter.

## 2.3 Summary

At this point Vis is primarily in the prototype stage, because it is hosted on a small PC and lacks the computing resources and bandwidth to transfer and store very large files. However, the program is fully functional and has been used to process some streams of gauge configurations that are made freely available by various groups via the NERSC archive [U.S. DOE, 2010].

Visualization algorithms can help physicists gain new insights into the physics of QCD, and Lattice QCD computations in particular. Our hope is that Vis can lower the barrier of entry, and enable physicists to look more deeply into their data. Vis can be downloaded from: <https://launchpad.net/qcdvis>

## Acknowledgments

Project funded by the Department of Energy grant DEFC02-06ER41441.

## Bibliography

- [Altarelli, 2002] Altarelli, G. (2002). A QCD primer. URL: <http://arxiv.org/abs/hep-ph/0204179v1>.
- [Benger, 2010] Benger, W. (2010). Vish website. URL: <http://vish.origo.ethz.ch/>.
- [Di Pierro, 2010] Di Pierro, M. (2010). web2py website. URL: <http://web2py.com>.
- [Hunter et al., 2010] Hunter, J., Dale, D., & Droettboom, M. (2010). matplotlib website. URL: <http://matplotlib.sourceforge.net/>.
- [Massimo Di Pierro, 2006] Massimo Di Pierro (2006). AN ALGORITHMIC APPROACH TO QUANTUM FIELD THEORY. *International Journal of Modern Physics A*, 21, 405–448.
- [Novaes, 1999] Novaes, S. F. (1999). Standard model: An Introduction. *Proceedings of the X J. A. Swieca Summer School*. URL: <http://arxiv.org/abs/hep-ph/0001283v1>.
- [U.S. DOE, 2010] U.S. DOE (2010). Nersc website. URL: <http://qcd.nersc.gov/>.

## mc4qcd: Online Analysis Tool for Lattice QCD

PoS (ACAT2010) 054

**Massimo Di Pierro\***

*School of Computing - DePaul University - Chicago*  
*E-mail:* [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu)

**Yaoqian Zhong**

*School of Computing - DePaul University - Chicago*  
*E-mail:* [ati\\_zhong@hotmail.com](mailto:ati_zhong@hotmail.com)

**Brian Schinazi**

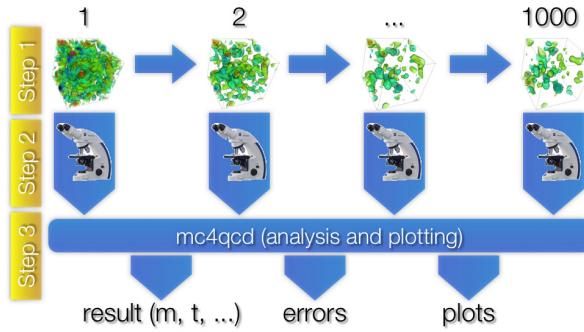
*School of Computing - DePaul University - Chicago*  
*E-mail:* [schinazi@gmail.com](mailto:schinazi@gmail.com)

mc4qcd is a web based collaboration tool for analysis of Lattice QCD data. Lattice QCD computations consists of a large scale Markov Chain Monte Carlo. Multiple measurements are performed at each MC step. Our system acquires the data by uploading log files, parses them for results of measurements, filters the data, mines for required information by aggregating results, represents the results as plots and histograms, and it further allows refining and interaction by fitting the results. The system computes moving averages and autocorrelations, builds bootstrap samples and bootstrap errors, and allows modeling the data using Bayesian correlated constrained linear and non-linear fits. It can be scripted to allow real time visualization of results form an ongoing computation. The system is modular and it can be adapted to automating the analysis workflow of different types of MC computations.

*13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research,  
ACAT2010  
February 22-27, 2010  
Jaipur, India*

---

\*Speaker.



**Figure 1:** The typical three steps of a lattice computation.

## 1. Introduction

Lattice QCD is a numerical approach to Quantum Chromodynamics. Its primary goals are to describe the non-perturbative behavior of quarks and to compute properties of hadronic matter. Typical Lattice QCD computations are comprised of three steps (Figure 1). Step 1 consists of a Markov Chain Monte Carlo that generates *gauge configurations*, i.e. datasets that represent possible evolutions of the gluon field. Step 2 consists of performing measurements on each of these gauge configurations, typically measurements of correlation functions between different operators at different locations in space and time. Step 3 consists of averaging the results of step 2 and performing a statistical analysis of the result. The outputs of step 3 may include values such as the masses and lifetimes of mesons and baryons.

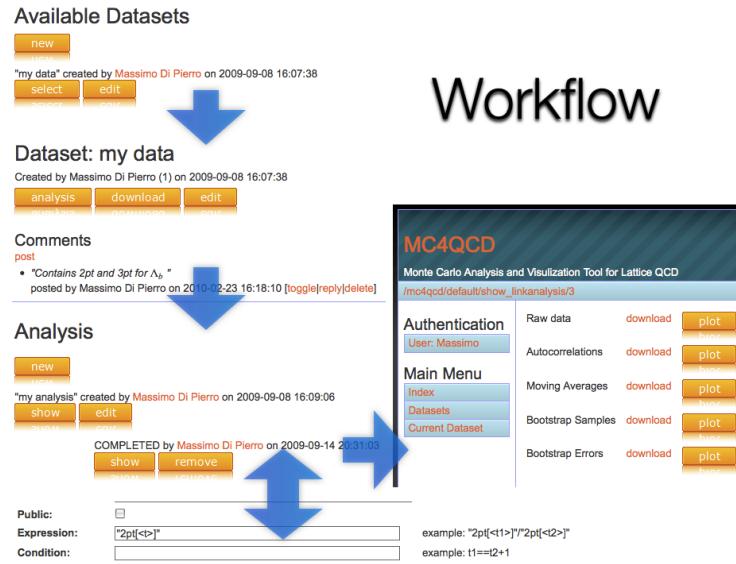
In this procedure, Step 1 is the most computationally expensive, but it is also the most automated. Step 2 is also computationally intensive, of magnitude dependent on the type of question that one is trying to answer. Typically, multiple measurements are performed on each gauge configuration such that one can answer multiple questions by aggregating them in different ways in Step 3. This final step is relatively inexpensive from a computational point of view, but it is where most physicists spend their time: analyzing data and generating plots. The mc4qcd software aims to automate this process.

The data communicated between Step 1 and Step 2 is highly structured, using one of a small set of file formats that scientists have agreed on. Conversely, there is no standard format for encoding the data communicated between Step 2 and Step 3. Such standardization would be very difficult because the data is unstructured and is rarely communicated between different research groups. Additionally, the data is normally stored in flat text files with no metadata, save occasional comments printed in the output that are meant for human, not machine, consumption.

The mc4qcd application is specifically designed to ameliorate this situation.

The software performs the following basic operations:

- Import: Data can be uploaded either via a web interface or by means of web services.
- Extraction and transformation: The application parses the output text files from Step 2 computations and uses regular expressions to identify important variables. The regular expres-



**Figure 2:** A collection of screenshots of the mc4qcd program.

sions provide a template for transforming the original files into a structured dataset (in the form a collection of tables).

- **Filtering:** In addition to regular expressions, the user can apply mathematical constraints to filter data.
- **Mining:** mc4qcd aggregates data by computing expressions of interest and performing standard statistical analysis.
- **Visualization:** It produces those plots that physicists normally look at to better read the information, including moving averages, autocorrelations, and bootstrap error.
- **Exploration:** It allows users to explore their data by zooming plots and interactively applying addition filters.
- **Fitting:** mc4qcd performs non-linear, constrained, correlated, and Bayesian fits, or combinations thereof.
- **Output:** Data, results, and plots can be exported.
- **Administration:** The software provides several administrative features, including web-based collaboration and commenting, traceability, ownership and permissions, public and private datasets, logging, storage of results, and annotations using LaTeX.

None of the above operations are new or exceedingly original. Physicists already perform them using other programs and plotting tools. The limitation of the current solutions is that they are not sufficiently general, and they require *ad hoc* programming.

The added value of mc4qcd lies in its generality, automation capabilities, ease of use, collaborative features, and its enforcement of a workflow (Figure 2).

## 2. Architecture

mc4qcd is written in Python. It consists of three core modules, which can be accessed either using its web interface or through shell scripting:

- **ibootstrap**: This module is responsible for reading a log file from a lattice computation, extracting the data contained therein, and performing aggregations and statistical analysis. The required inputs to this module are: the name of the log file, the regular expressions required to extract values, the expression to be computed with those values, and optional constraints. Once configured, it loops over these tasks and generates multiple output files in CSV format.
- **iplot**: This module reads the output of ibootstrap and generates corresponding plots.
- **ifit**: This module implements various fitting algorithms that can read the output of ibootstrap and cooperate with iplot to include fitting results in plots.

The web application is built around these three modules using the web2py framework. The framework provides Application Program Interfaces to perform the following tasks: concurrency handling, generation of dynamic web pages, web services, streaming of large files both in and out, security (authentication, authorization, XSS and Injection prevention), connecting to a database for storage (web2py supports 10 different database engines), maintaining state through user sessions and cookies, caching results of recurrent operations (for speed), and internationalization capabilities.

In the Physics community ROOT is the “de facto” library for fitting and plotting, yet matplotlib is the standard plotting library for Python. For our application portability was a priority and we found matplotlib to be adequate to our needs.

## 3. Example

A typical example of medium complexity consists of the computation of a meson’s mass. This value can be computed (after Wick rotation to Euclidean time) by fitting a two point correlation

$$C_2(t_x - t_y) \equiv FT_{\mathbf{xy}} \langle 0 | j(\mathbf{x}, t_x) j^\dagger(\mathbf{y}, t_y) | 0 \rangle \quad (3.1)$$

with a sum of exponentials

$$C_2(t) = \sum_i A_i e^{-m_i t} \quad (3.2)$$

Here  $FT_{\mathbf{xy}}$  is a zero momentum spatial Fourier transform (in  $\mathbf{x}$  and  $\mathbf{y}$ ), and  $j(\mathbf{x}, t_x)$  is a creation operator with the same quantum numbers as the meson we are interested in. The expectation value is computed numerically on the lattice in Step 2, as described in the introduction.

The  $m_i$  parameters are the masses of those states with the same quantum numbers as  $j$ . They can be extracted form the fit. The smaller of mass  $m_0$  is the ground state for the meson created by the operator  $j$ .

Assume we have multiple measures for  $C_2(t)$ , along with other measurements, one for each gauge configuration, and that their values are stored in a flat text file as in the following:

```
loading gauge configuration 0
C2[00] = 0.00000
C2[01] = 0.00000
C2[02] = 0.00000
C2[03] = 0.00000
C2[04] = 0.00000
C2[05] = 0.00000
C2[06] = 0.00000
C2[07] = 0.00000
C2[08] = 0.00000
C2[09] = 0.00000
C2[10] = 0.00000
C2[11] = 0.00000
C2[12] = 0.00000
C2[13] = 0.00000
C2[14] = 0.00000
C2[15] = -1.029314
C3[00][00] = 5.203888
C3[00][01] = 4.372048
...
C3[15][15] = 4.372048
...
comments ...
```

In this case C2 [i] is just an arbitrary label – any name could be used. The index in square brackets is the value for  $t$ . For each  $t$  there are as many measurements as there are gauge configurations. The file may also contain additional text and comments. The different pages in the figure correspond to different gauge configurations and their output is in the same file.

We upload this file into mc4qcd and then ask it to compute the following expression:

1 | "C2[<t>]"

which means: *look for all variables labeled C2 [<t>] where t is an index implicitly defined.* Everything inside double quotes identify variables via their corresponding pattern. Everything outside double quotes must be a valid Python expression, which can use all functions from the Python math module (such as log, exp, sin, etc.), as well as user defined functions.

The program, for each  $t$ , extracts all values from the file, computes simple averages, moving averages, autocorrelations, bootstrap errors and sample distributions (some sample plots are shown in Figure 3).

Next, to fit this with an exponential we simply type:

1 | a\*exp(-m\*t )@a=1 ,a\_bu=0.2 ,m=0.1

Here mc4qcd understands that  $t$  is the parameter defined implicitly when parsing the data, while  $a$  and  $m$  are arbitrary unknown parameters that will be determined in the fit. The initial values used for fitting (priors) are specified on the right-hand side of the @ symbol.  $a_{\text{bu}}$  is the Bayesian uncertainty of the prior  $a$ . In this case, we performed a single mass fit, because this term dominates for large  $t$ . Multimass fits are also supported by mc4qcd.

The results of the fit are shown, superimposed to the data, in Figure 3. The fitting results and the Hessian describing the correlation between the fitting parameters ( $a$  and  $m$ ) are displayed under the plot.

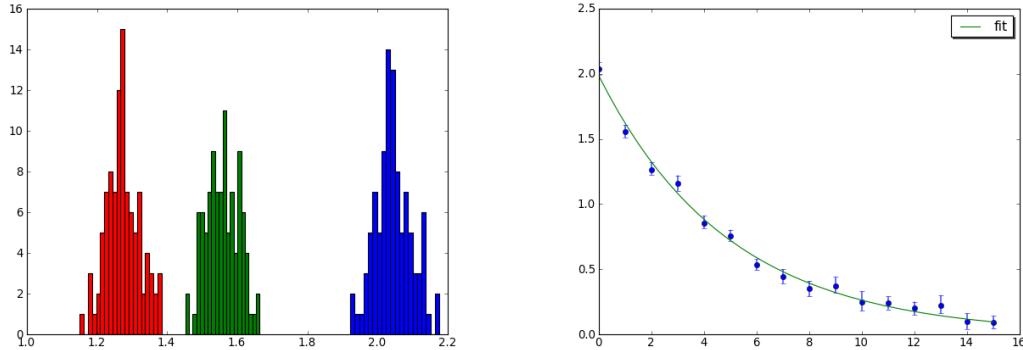
mc4qcd also allows more complex expression. Here are some examples:

```
1 log("C2[<t2>]"/"C2[<t2>]") for t1==t2+1  
2 0.5*(C2[<t1>]"+"C2[<t2>]) for t1==32-t2 and t1>16  
3 "C3[<t1>]<[t2>]"/"C2[<t1>]"/"C2[<t2>]")
```

The latter example extracts a matrix element from a 3 point correlation function  $C_3$ .

The fitting module also allows for complex expressions such as correlated Bayesian fits:

```
| { 1:a ,2:b }[ t1 ]*exp(-m*t1 )@a=1,a bu=0.2,b=2,m=0.1
```



**Figure 3:** The figure on the left shows the distribution of bootstrap samples for  $C_2(t)$  at three different values of  $t$ . The figure on the right shows the two point correlation function at different  $t$  with its bootstrap errors, fitted by an exponential function.

(an exponential fit in  $t_1$  with different coefficients depending on  $t_2$ ).

For the data using in this example, the results of the analysis are generated in real time. In the most general case the timing depends on the complexity of the expression and the number of MC samples. For actual production data running the analysis rarely exceeds one minute.

#### 4. Conclusions

mc4qcd is a new tool designed specifically for Lattice QCD analysis. It is very general, with certain limitations. For example, even though it can read unstructured data, it still often requires that the data be tagged by strings that can be recognized by the regular expression parser. Planned future enhancements include extending mc4qcd to give it the ability to learn more complex templates and handle more use cases. Another limitation is the restricted ability to customize plots. Because all of the work is carried out automatically, it generates good quality images for screen presentation, but does not allow as much customization as is sometimes required for publication in scientific journals.

mc4qcd can be a valuable tool for increasing collaboration between scientists working on Lattice QCD. It is distributed as a web2py package. It can be demoed and downloaded from <http://latticeqcd.org>

**Acknowledgements** This work has been supported by the US Department of Energy under grant DEFC02-06ER41441.

#### References

- [1] M. Di Pierro, *An Algorithmic approach to Lattice Field Theory*, Int. J. of Mod. Phys. A, Vol. 21, Num. 3, (2006)
- [2] Sandro Tosi, *Matplotlib*, Packt Publishing (2009)
- [3] M. Di Pierro, *web2py Manual*, 2nd Ed., Wiley (2009)

# ***B and D meson decay constants from 2+1 flavor improved staggered simulations***

**E. T. Neil<sup>\*a,†</sup>, Jon A. Bailey<sup>a,b</sup>, A. Bazavov<sup>c</sup>, C. Bernard<sup>d</sup>, C. M. Bouchard<sup>e,a,g</sup>,  
 C. DeTar<sup>h</sup>, M. Di Pierro<sup>i</sup>, A. X. El-Khadra<sup>e</sup>, R. T. Evans<sup>e</sup>, E. Freeland<sup>e,f</sup>, E. Gamiz<sup>a,j</sup>,  
 Steven Gottlieb<sup>k</sup>, U. M. Heller<sup>l</sup>, J. E. Hetrick<sup>m</sup>, R. Jain<sup>e</sup>, A. S. Kronfeld<sup>a</sup>, J. Laiho<sup>n</sup>,  
 L. Levkova<sup>h</sup>, P. B. Mackenzie<sup>a</sup>, M. B. Oktay<sup>h</sup>, J. N. Simone<sup>a</sup>, R. Sugar<sup>o</sup>,  
 D. Toussaint<sup>p</sup>, and R. S. Van de Water<sup>c</sup>**

<sup>a</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois, USA*

<sup>b</sup>*Department of Physics and Astronomy, Seoul National University, Seoul, ROK*

<sup>c</sup>*Physics Department, Brookhaven National Laboratory, Upton, NY, USA*

<sup>d</sup>*Department of Physics, Washington University, St. Louis, Missouri, USA*

<sup>e</sup>*Physics Department, University of Illinois, Urbana, Illinois, USA*

<sup>f</sup>*Department of Physics, Benedictine University, Lisle, Illinois, USA*

<sup>g</sup>*Department of Physics, The Ohio State University, Columbus, OH, USA*

<sup>h</sup>*Physics Department, University of Utah, Salt Lake City, Utah, USA*

<sup>i</sup>*School of Computing, DePaul University, Chicago, Illinois, USA*

<sup>j</sup>*Department of Physics, University of Granada, Granada, Spain*

<sup>k</sup>*Department of Physics, Indiana University, Bloomington, Indiana, USA*

<sup>l</sup>*American Physical Society, Ridge, New York, USA*

<sup>m</sup>*Physics Department, University of the Pacific, Stockton, California, USA*

<sup>n</sup>*SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow, UK*

<sup>o</sup>*Department of Physics, University of California, Santa Barbara, California, USA*

<sup>p</sup>*Department of Physics, University of Arizona, Tucson, Arizona, USA*

<sup>†</sup>*E-mail: [eneil@fnal.gov](mailto:eneil@fnal.gov)*

## **Fermilab Lattice and MILC Collaborations**

We give an update on simulation results for the decay constants  $f_B, f_{B_s}, f_D$  and  $f_{D_s}$ . These decay constants are important for precision tests of the standard model, in particular entering as inputs to the global CKM unitarity triangle fit. The results presented here make use of the MILC (2+1)-flavor asqtad ensembles, with heavy quarks incorporated using the clover action with the Fermilab method. Partially quenched, staggered chiral perturbation theory is used to extract the decay constants at the physical point. In addition, we give error projections for a new analysis in progress, based on an extended data set.

*XXIX International Symposium on Lattice Field Theory  
 July 10-16, 2011  
 Squaw Valley, Lake Tahoe, California*

---

<sup>\*</sup>Speaker.

## 1. Introduction

Within the standard model, the decay of mesons containing heavy quarks (in particular,  $B$  and  $D$  mesons) into purely leptonic final states provides an important testing ground for a number of theoretical ideas. Such decays involve both weak and strong interactions simultaneously, so that a complete understanding of the standard model is necessary to describe them. In particular, the decay width of a charged meson is proportional to both the meson decay constant (determined by strong interactions) and the CKM mixing angle,

$$\Gamma(H \rightarrow \ell \bar{\nu}_\ell) \propto f_H^2 |V_{Qq}|^2. \quad (1.1)$$

Because this fully leptonic decay has no hadrons in the final state, the meson decay constant  $f_P$  can be readily and accurately determined by lattice simulations [1, 2, 3, 4]. Such a determination is in fact necessary in order to extract the CKM angles from experimental measurements of these decays, and precise computations of the decay constants could potentially reveal the presence of new physics through tension in the CKM unitarity triangle [5, 6, 7, 8]. In addition, certain leptonic decay channels (e.g.  $B_s \rightarrow \mu^+ \mu^-$ ) are both loop and CKM suppressed in the standard model, and so they may be particularly sensitive to flavor-changing interactions induced by new physics [9].

## 2. Simulation Details

We make use of the MILC asqtad-improved staggered gauge configurations, with  $2 + 1$  dynamical quarks in the sea [10]. For the light valence quarks, we make use of the same staggered action, while charm and bottom valence quarks are incorporated using the clover action with the Fermilab interpretation [11]. The particular set of ensembles used to obtain the results presented here, along with the number of configurations and other relevant information, are detailed in Table 1. Data on the coarsest lattice spacing  $a \approx 0.15$  fm are shown in the analysis but are used only for the purpose of estimating the discretization errors; these points are excluded from the final fit used for chiral and continuum extrapolation.

## 3. Analysis and Fitting

The heavy meson decay constant is determined through the overlap of the meson wavefunction  $|H\rangle$  with the axial vector current:

$$\langle 0 | \mathcal{A}^\mu | H(p) \rangle (M_H)^{-1/2} = i(p^\mu / M_H) (f_H \sqrt{M_H}) \equiv i(p^\mu / M_H) \phi_H. \quad (3.1)$$

The quantity  $\phi_H \equiv f_H \sqrt{M_H}$  is thus proportional to the ground-state amplitude of the two-point function between the axial vector current and a heavy-light pseudoscalar operator  $\mathcal{O}$ . We therefore extract  $\phi_H$  by fitting this two-point function simultaneously with the two-point pseudoscalar correlator,

$$\Phi_2^s(t) = \frac{1}{4} \sum_{a=1}^4 \langle A_a^{4\dagger}(t, \mathbf{x}) \mathcal{O}_a^{(s)}(0) \rangle, \quad (3.2)$$

$$C_2^{s,s'}(t) = \frac{1}{4} \sum_{a=1}^4 \langle \mathcal{O}_a^{(s)\dagger}(t, \mathbf{x}) \mathcal{O}_a^{(s')}(0) \rangle, \quad (3.3)$$

$\approx a$ [fm]	$am_h$	$am_l$	$\beta$	$r_1/a$	$N_{\text{conf}}$
0.09	0.031	0.0031	7.08	3.75	435
		0.0062	7.09	3.79	557
		0.0124	7.11	3.86	518
0.12	0.050	0.005	6.76	2.74	678
		0.007	6.76	2.74	833
		0.010	6.76	2.74	592
0.15	<i>0.0484</i>	0.020	6.79	2.82	460
		0.030	6.81	2.88	549
		0.0194	6.586	2.26	631
<i>0.15</i>	<i>0.0484</i>	0.0290	6.600	2.29	576

**Table 1:** Table of gauge configurations used for the full analysis to be presented. The ensembles with  $a \approx 0.15$  fm (italics) are excluded from the final chiral/continuum extrapolation, as described in the text.

where  $a$  is a staggered taste index. Precise definitions of the interpolating operators  $A^4$  and  $\mathcal{O}$  are given in Ref. [12].

For a correlation function with source type  $s$  and sink type  $s'$ , we fit to the “factorized” functional form

$$C_{ss'}(t) = \sum_{n=0}^{N_X} \left[ A_{s,n} A_{s',n} \left( e^{-E_n t} + e^{-E_n(N_t-t)} \right) - (-1)^t A'_{s,n} A'_{s',n} \left( e^{-E'_n t} + e^{-E'_n(N_t-t)} \right) \right] \quad (3.4)$$

where  $N_X$  denotes the number of excited states included in the fit. The pseudoscalar source and sink type  $s,s'$  can be either point-like or smeared, so that a total of six distinct correlators are available for analysis. In the results shown here, joint fits are carried out to various combinations of correlators, with Bayesian priors imposed as constraints on the fit parameters.

From the two-point fits, we extract a bare value for the ground-state amplitude between an axial-vector current and pseudoscalar operator, which must then be renormalized in order to obtain the decay constant:

$$\phi_H = \sqrt{2} Z_{A_{Qq}^4} A_{A_{Qq}^4,0}. \quad (3.5)$$

To compute the heavy-light axial current renormalization constant  $Z_{A_{Qq}^4}$ , we divide it into flavor-diagonal contributions  $Z_{V_{qq}^4}, Z_{V_{QQ}^4}$ , which are determined non-perturbatively, and a flavor off-diagonal piece  $\rho_{A_{Qq}^4}$  which is computed in lattice perturbation theory [13]. Our renormalized result for  $\phi_H = f_H \sqrt{M_H}$  is thus given by

$$\phi_H = \sqrt{2} Z_{A_{Qq}^4} A_{A_{Qq}^4,0} = \sqrt{2} (\rho_{A_{Qq}^4} \sqrt{Z_{V_{qq}^4} Z_{V_{QQ}^4}}) A_{A_{Qq}^4,0}. \quad (3.6)$$

We use rooted staggered chiral perturbation theory (rS $\chi$ PT) [14] to extrapolate our results simultaneously to the continuum limit and to the physical light-quark masses. (Heavy quark masses are tuned non-perturbatively to give physical heavy-light meson masses, so no extrapolation is necessary for them. Details of the tuning procedure are given in [15].) The chiral fit functions incorporate terms describing a number of different effects, including discretization errors, finite-volume corrections, and hyperfine splittings.

**Table 2:** Error budget for the decay constants, as obtained in Section 4. In addition, projected improvements to the decay constant error budget for the updated analysis in progress (discussed in Section 5) are shown italicized and in brackets.

Source	$f_{D^+}$ ( MeV)	$f_{D_s}$ ( MeV)	$f_{B^+}$ ( MeV)	$f_{B_s}$ ( MeV)
Statistics	2.3 [1.1]	2.3 [1.1]	3.6 [1.8]	3.4 [1.7]
Heavy-quark disc.	8.2 [3.6]	8.3 [3.6]	3.7 [1.9]	3.8 [2.0]
Light-quark disc.	2.9 [0.7]	1.5 [0.3]	2.5 [0.6]	2.1 [0.5]
Chiral extrapolation	3.2 [1.6]	2.2 [1.1]	2.9 [1.5]	2.8 [1.4]
Heavy-quark tuning	2.8 [2.0]	2.8 [2.0]	3.9 [2.4]	3.9 [2.4]
$Z_{V_{QQ}^4}$ and $Z_{V_{qq}^4}$	2.8 [1.4]	3.4 [1.7]	2.6 [1.5]	3.1 [1.9]
$u_0$ adjustment	1.8 [0]	2.0 [0]	2.5 [0]	2.8 [0]
Other sources	3.8 [3.8]	3.0 [3.0]	3.5 [3.5]	4.8 [4.8]
Total [ <i>projected</i> ] error	11.3 [6.1]	10.8 [5.6]	8.9 [5.5]	9.5 [6.4]

## 4. Results

Applying the procedure outlined above, we obtain the values for  $\phi$  shown in Figures 1 and 2, renormalized and in units of the standard scale  $r_1$ . The chiral best-fit curve to the points is also shown, both explicitly at each lattice spacing and extrapolated to the continuum limit.

Evaluating the continuum best-fit curves at the physical points, we obtain the following values for the decay constants and their ratios:

$$f_{B^+} = 196.9(8.9) \text{ MeV}, \quad (4.1)$$

$$f_{B_s} = 242.0(9.5) \text{ MeV}, \quad (4.2)$$

$$f_{B_s}/f_{B^+} = 1.229(0.026), \quad (4.3)$$

$$f_{D^+} = 218.9(11.3) \text{ MeV}, \quad (4.4)$$

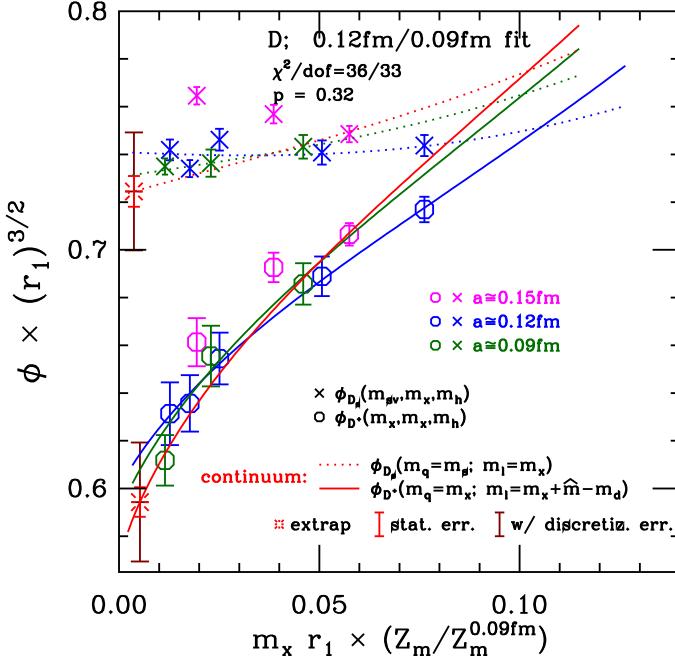
$$f_{D_s} = 260.1(10.8) \text{ MeV}, \quad (4.5)$$

$$f_{D_s}/f_{D^+} = 1.188(0.025). \quad (4.6)$$

The error bars quoted here include both statistical and systematic sources of error, which are accounted for in a detailed error budget. A summary of the full error budget for the individual decay constants is given in Table 2. We discuss the error budget further in Section 5 below, but a thorough discussion is beyond the scope of this paper. Instead, we refer the reader to Ref. [12], which contains a complete discussion of the systematic error analysis, including the full error budget for the decay-constant ratios.

## 5. Outlook

A new analysis following the approach outlined above is currently in progress, based on an expanded set of gauge configurations as shown in Table 3. In addition to extending the available simulations to finer lattice spacing and smaller quark mass, the “new” data set includes large increases in statistics for several ensembles.

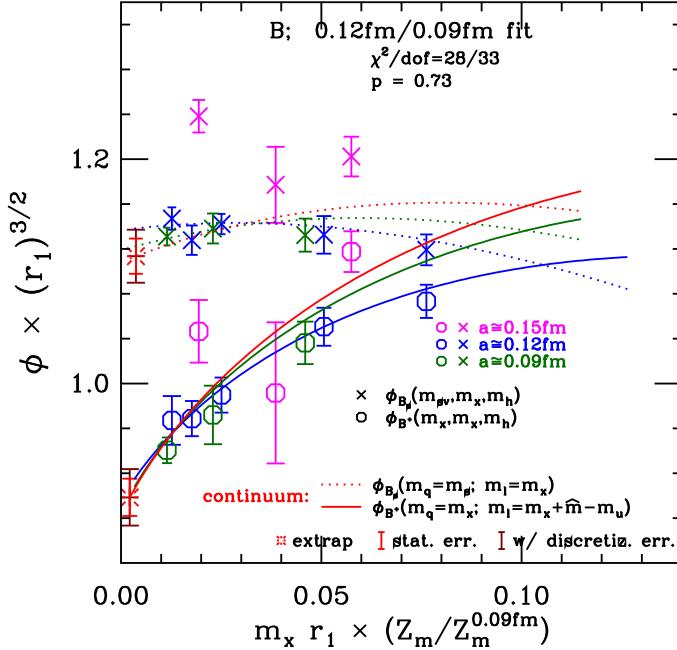


**Figure 1:** Extracted values of  $\phi$  and chiral best-fit curves for the  $D$  system. Only points where valence and sea light-quark masses are equal are shown here. Data from the  $a \approx 0.15$  fm ensemble are shown but not included in the fit. The red curve and symbols show the continuum extrapolation and the continuum/physical point extrapolation, respectively, for both  $\phi_{D_s}$  and  $\phi_D$ . For the fully extrapolated points, the inner error bars (bright red) represent statistical errors only, while the outer errors (dark red) include discretization errors.

Projected improvements in the error budget when the new data set is included are shown alongside the previous error estimates in Table 2. Statistical errors are projected to improve as  $\sqrt{N_{\text{cfg}}}$ , with  $N_{\text{cfg}}$  the number of gauge configurations available for a given ensemble. For the various discretization errors, the projected improvements are a result of reducing the smallest lattice spacing available from  $a = 0.09$  fm to  $a = 0.045$  fm. Light-quark discretization errors are estimated to scale as  $\mathcal{O}(\alpha_s a^2)$ ; the heavy-quark discretization errors are estimated using the known functional dependence, which has several terms. The decrease in the chiral extrapolation error is projected based on the lightest available value of the quark mass in  $r_1$  units,  $m_x r_1$ . The heavy-quark tuning error is based on a combination of statistical and discretization errors, and is treated as such. The “ $u_0$  adjustment” error is the result of using different tadpole improvement factors for the valence and sea quarks. This is rectified in the new data analysis, eliminating the associated error. Finally, the error estimates for the heavy-quark renormalization factors  $Z_{V_{QQ}^4}, Z_{V_{qq}^4}$  are based on preliminary non-perturbative results for those quantities.

## Acknowledgments

Computations for this work were carried out with resources provided by the USQCD Collaboration, the Argonne Leadership Computing Facility, the National Energy Research Scientific Com-



**Figure 2:**  $\phi$  values and chiral best-fit curves as in Figure 1, but for the  $B$  system.

puting Center, and the Los Alamos National Laboratory, which are funded by the Office of Science of the U.S. Department of Energy; and with resources provided by the National Institute for Computational Science, the Pittsburgh Supercomputer Center, the San Diego Supercomputer Center, and the Texas Advanced Computing Center, which are funded through the National Science Foundation's Teragrid/XSEDE Program. This work was supported in part by the U.S. Department of Energy under Grants No. DE-FC02-06ER41446 (C.D., L.L., M.B.O.), No. DE-FG02-91ER40661 (S.G.), No. DE-FG02-91ER40677 (C.M.B., R.T.E., E.D.F., E.G., R.J., A.X.K.), No. DE-FG02-91ER40628 (C.B.), No. DE-FG02-04ER-41298 (D.T.); by the National Science Foundation under Grants No. PHY-0555243, No. PHY-0757333, No. PHY-0703296 (C.D., L.L., M.B.O.), No. PHY-0757035 (R.S.), and No. PHY-0704171 (J.E.H.); by the URA Visiting Scholars' program (C.M.B., R.T.E., E.G., M.B.O.); and by the Fermilab Fellowship in Theoretical Physics (C.M.B.). This manuscript has been co-authored by employees of Brookhaven Science Associates, LLC, under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

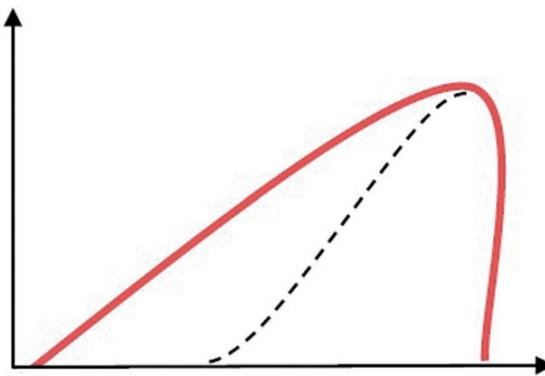
## References

- [1] C. McNeile, C. T. H. Davies, E. Follana, K. Hornbostel and G. P. Lepage, arXiv:1110.4510 [hep-lat].
- [2] C. T. H. Davies *et al.*, Phys. Rev. D **82**, 114504 (2010) [arXiv:1008.4018 [hep-lat]].
- [3] P. Dimopoulos *et al.* [ETM Collaboration], arXiv:1107.1441 [hep-lat].

$\approx a$ [fm]	$am_h$	$am_l$	$\beta$	$r_1/a$	$N_{\text{conf}}$ (old run)	$N_{\text{conf}}$ (new run)
0.045	0.014	0.0028	7.81	7.21	—	800
		0.0018	7.46	5.31	—	825
		0.0025	7.465	5.33	—	800
		0.0036	7.47	5.35	—	631
		0.0072	7.48	5.40	—	591
	0.031	0.00155	7.075	3.74	—	790
		0.0031	7.08	3.75	435	577
		0.00465	7.085	3.77	—	983
		0.0062	7.09	3.79	557	1377
		0.0124	7.11	3.86	518	1476
0.12	0.050	0.005	6.76	2.74	678	1419
		0.007	6.76	2.74	833	1274
		0.010	6.76	2.74	592	1664
		0.020	6.79	2.82	460	1637
		0.030	6.81	2.88	549	—
0.15	0.0484	0.0097	6.572	2.22	631	—
		0.0194	6.586	2.26	631	—
		0.0290	6.600	2.29	576	—

**Table 3:** Table of gauge configurations for the updated analysis in progress. Only configurations labelled “old run” were used to obtain the results presented in Section 4. The analysis in progress will make use of all configurations from both runs combined.

- [4] F. Farchioni *et al.*, PoSLATTICE **2010**, 128 (2010) [arXiv:1012.0200 [hep-lat]].
- [5] A. Lenz *et al.* [CKMfitter Collaboration], Phys. Rev. D **83**, 036004 (2011) [arXiv:1008.1593 [hep-ph]].
- [6] E. Lunghi and A. Soni, Phys. Lett. B **697**, 323 (2011) [arXiv:1010.6069 [hep-ph]].
- [7] J. Laiho, E. Lunghi and R. Van De Water, PoSFPCP **2010**, 040 (2010) [arXiv:1102.3917 [hep-ph]].
- [8] A. J. Bevan *et al.* [UTfit Collaboration], PoS **ICHEP2010**, 270 (2010) [arXiv:1010.5089 [hep-ph]].
- [9] T. Aaltonen *et al.* [CDF Collaboration], Phys. Rev. Lett. **107**, 191801 (2011) [arXiv:1107.2304 [hep-ex]].
- [10] A. Bazavov *et al.*, Rev. Mod. Phys. **82**, 1349 (2010) [arXiv:0903.3598 [hep-lat]].
- [11] A. X. El-Khadra, A. S. Kronfeld and P. B. Mackenzie, Phys. Rev. D **55**, 3933 (1997) [hep-lat/9604004].
- [12] A. Bazavov *et al.* [Fermilab Lattice and MILC Collaborations], “B- and D-meson decay constants from three-flavor lattice QCD,” [arXiv:1112.3051 [hep-lat]].
- [13] A. X. El-Khadra *et al.*, PoSLAT **2007**, 242 (2007) [arXiv:0710.1437 [hep-lat]].
- [14] C. Aubin and C. Bernard, Phys. Rev. D **73**, 014515 (2006) [hep-lat/0510088].
- [15] C. Bernard *et al.* [Fermilab Lattice and MILC Collaborations], Phys. Rev. D **83**, 034503 (2011) [arXiv:1003.1937 [hep-lat]].



# Effects of skewness and kurtosis on portfolio rankings

MASSIMO DI PIERRO\*† and JACK MOSEVICH‡

†DePaul University, School of Computer Science, 243 S. Wabash Avenue, Chicago, IL 60604, USA

‡DePaul University, College of Commerce, 1 E. Jackson, Chicago, IL 60604, USA

(Received 23 January 2007; in final form 6 July 2009)

## 1. Introduction

In this paper we discuss the issue of portfolio ranking and selection. We will concentrate on selecting one portfolio among a finite set of portfolios, where each portfolio is characterized by its own distribution of returns  $p(x)$ . This distribution may be inferred from past performance and assumed to be persistent, or it may be derived by some model of future performance.

We distinguish two main cases.

- **With risk-free asset:** An amount  $A$  must be invested and it can be distributed between a risk-free asset (that pays a risk-free rate  $r$ ) and the selected portfolio.
- **Without risk-free asset:** An amount  $A$  must be invested exclusively in the selected portfolio.

The two problems are apparently similar, but they are conceptually different and therefore have different solutions. In this paper we clarify the difference between the two, discuss ranking schemes in both cases, and propose a theoretically sound ranking scheme for the second case in the presence of distributions that exhibit arbitrary skewness and kurtosis.

We conclude that, without a risk-free asset, and in the presence of skewness and kurtosis, for a distribution with finite momenta (distributions without fat tails), a rational investor using the CARA utility function should select that portfolio with a higher value of

$$\mu - \frac{m\sigma^2}{2} + \frac{m^2\sigma^3 S}{6} - \frac{m^3\sigma^4(K-3)}{720}, \quad (1)$$

where  $\mu$  is the average return of the portfolio,  $\sigma$  the standard deviation,  $S$  the skewness,  $K$  the kurtosis, and  $m$  the CARA subjective risk-aversion parameter. Equation (1) extends a previous conclusion of Lévy and Markowitz (1979) by taking into considerations the effects of skewness and kurtosis. Our formula provides a simple practical way to select one out of many mutually exclusive portfolios. In this paper we discuss only the issue of portfolio selection, not portfolio construction. For a review, we refer the reader to the work of Ortobelli *et al.* (2005). In this analysis, we will always identify a portfolio by its distribution of returns  $p$  since we are not interested in the composition of the portfolio.

## 2. Portfolio selection

In this section we discuss the similarities and differences between the cases with and without a risk-free asset. The

\*Corresponding author. Email: mdipierro@cs.depaul.edu

main similarity between the two cases is that they both require subjective choice. In Utility Theory (Neumann and Morgenstern 1947, Nash 1950), this is the choice of a utility function  $U(x)$  and its parameters.  $U(x)$  is a function that takes as input a possible return  $x$  from an investment and outputs a number that represents the investor's degree of satisfaction associated with return  $x$ . In neoclassical economics, an investor is defined 'rational' if:

- the investor has a utility function  $U(x)$ ;
- the investor acts in order to maximize  $U(x)$ ; and
- $U(x)$  is monotonic increasing (a higher return is preferred to a lower return).

A common choice for the utility function is  $U_{\text{CARA}}(x) \equiv -e^{-mx}$ , which is known as the 'Constant Absolute Risk Averse' utility function.  $m$  is the risk-aversion parameter and it is of the order 1. We call a rational investor with a CARA utility function a 'rational risk-averse investor' or, more simply, 'investor'. The investor will rank a portfolio  $p$  by weighting the utility of a return  $x$  with the probability of that return  $p(x)$ , thus he would use the ranking function

$$R_U(p) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} U(x) p(x) dx, \quad (2)$$

or an equivalent function. Two rankings function,  $R_1$  and  $R_2$ , are equivalent if and only if they produce the same ranking, i.e. if there is a monotonic function  $h$  that maps  $R_2(p)$  into  $R_1(p)$  for every  $p$ . We will indicate the equivalence of two ranking functions as  $R_1 \sim R_2$ .

### 2.1. With risk-free asset

In this case, our investor can choose to invest part of the funds  $A$  in a risk-free asset, for example a US Treasury bill. This means that, given any two portfolios  $p_1$  and  $p_2$  characterized by an average return and risk (standard deviation)  $\mu_1, \sigma_1$  and  $\mu_2, \sigma_2$ , if

$$\frac{\mu_1 - r}{\sigma_1} > \frac{\mu_2 - r}{\sigma_2}, \quad (3)$$

then the investor should never choose  $p_2$  over  $p_1$ . This is because the investor can invest a fraction  $\alpha = \sigma_2/\sigma_1$  of the total funds  $A$  in portfolio  $p_1$  and a fraction  $(1 - \alpha)$  in the risk-free asset and obtain a new combined portfolio with the same risk as portfolio  $p_2$ , but a higher return, given by (Sharpe 1964)

$$\mu' = (1 - \alpha)r + \alpha\mu_1 > \mu_2. \quad (4)$$

Hence, portfolio  $p_1$  is always preferable to portfolio  $p_2$ . If one applies this argument to every portfolio in the set, one finds that our investor should invest part of the funds  $A$  in portfolio  $p$  with the largest value of

$$R_{\text{Sharpe}}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{\sigma}. \quad (5)$$

This is the well-known Sharpe ratio or Sharpe ranking function (Markowitz 1952, Sharpe 1964). The value of  $\alpha$  is then determined by maximizing the utility function  $U$ . If the expected returns of the portfolios have a Gaussian distribution, then

$$\alpha = \alpha \text{ that maximizes } \int U(\alpha x + (1 - \alpha)r) p(x) dx = \frac{\sqrt{\mu/r}}{m\sigma}. \quad (6)$$

If the return of the portfolios is not Gaussian-distributed, the first equality in equation (6) remains true, while the second equality is only an approximation since the integral must be performed numerically. Similarly, if one adopts a definition of risk other than the standard deviation of the returns, the argument that led to the  $R_{\text{Sharpe}}$  measure remains valid, but  $\sigma$  must be consistently replaced by the new measure of risk. For example, if one chooses to measure risk as the downside risk only

$$\sigma_n^- \stackrel{\text{def}}{=} \left[ \int_{-\infty}^r p(x)(r - x)^n dx \right]^{1/n}, \quad (7)$$

then  $\sigma$  is replaced by  $\sigma_n^-$ , and the Sharpe ranking function is replaced by the Kappa ranking function (Kaplan and Knowles 2004)

$$R_{\text{Kappa}-n}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{\sigma_n^-}. \quad (8)$$

The Kappa ranking function above is the Sortino (Sortino and Van Der Meer 1991, Sortino and Price 1994, Sortino and Forsey 1996) ranking function when  $n=2$  and it is equivalent to the Omega (Kazemi *et al.* 2003) ranking function when  $n=1$ . A different ranking scheme has been proposed by Stutzer (2000). For a theoretical analysis of 'good' vs. 'bad' risk measures we refer the reader to the work of Artzner *et al.* (2000). It was proven by Ortobelli *et al.* (2005) that all the above rankings are equivalent. In appendix A we provide exact mapping formulas from the above rankings to the Sharpe ratio. If returns are not Gaussian, Sharpe, Sortino, Kappa, and Omega are not equivalent because they follow from different definitions of risk.

### 2.2. Without risk-free asset, the wrong way

In this case, our investor has to choose a portfolio and invest the entire available funds in it, hence the argument presented at the beginning of the previous section does not apply. The reason behind the use of the Sharpe ranking (or the Sortino ranking) falls apart, as pointed out by Sharpe (1964). In this subsection we answer two questions.

- Is the use of the Sharpe ranking (or any of the other rankings) justified?
- If not, what is an appropriate ranking scheme that leads to the correct choice for a rational risk-averse investor?

Let us examine first the Sharpe ranking function and assume that portfolio returns are Gaussian-distributed.

An explicit computation shows that, for every Gaussian portfolio  $p$ ,

$$R_{U_{\text{naive}}}(p) \sim R_{\text{Sharpe}}(p), \quad (9)$$

where

$$U_{\text{naive}}(x) \stackrel{\text{def}}{=} \begin{cases} -1 & (\text{if } x < 0) \\ +1 & (\text{if } x \geq 0) \end{cases}, \quad (10)$$

and the monotonic mapping function between the two rankings is

$$h(y) = \text{erf}(y/\sqrt{2}). \quad (11)$$

Therefore, an investor who ranks portfolios using the Sharpe function in the absence of a risk-free asset is implicitly adopting the utility function in equation (10). The problem here is that equation (10) is not a risk-averse utility function. This function states that a positive return  $x > 0$  (gain) has utility  $+1$  and a negative return  $x < 0$  (loss) has utility  $-1$ . This investor does not believe that a 20% return is better than a 10% return, or that a loss of 100% is worse than a loss of 1%. The investor who uses the Sharpe ranking function in the absence of a risk-free asset is not a rational risk-averse investor.

Under the assumption of Gaussian returns, the Sortino, Omega, Stutzer and Kappa rankings are all equivalent (as proven by Ortobelli *et al.* 2005 and shown in appendix A), therefore the use of any of these rankings, in the case considered here, is not consistent with being risk-averse. Returns of real portfolios are, generally, non-Gaussian-distributed, but, if a ranking function works for a general distribution, it must also work for Gaussian returns. Since this is not true for Sharpe, Sortino, Omega, Stutzer, nor Kappa, these ranking schemes should not be used when the investor is not allowed to invest in a risk-free asset.

### 2.3. Without risk-free asset, the right way

Our investor, a rational risk-averse investor, would use the CARA utility function to make choices and would rank portfolios using equation (2) with  $U$  being  $U_{\text{CARA}}$ . In appendix B we prove that  $R_{U_{\text{CARA}}}$  is equivalent to  $R_*$ , where

$$R_*(p) \stackrel{\text{def}}{=} -\log(-R_{U_{\text{CARA}}}(p))/m, \quad (12)$$

$$= \mu - \frac{m\sigma^2}{2} + \frac{m^2\sigma^3 S}{6} - \frac{m^3\sigma^4(K-3)}{720} + O(m^4\sigma^5), \quad (13)$$

and

- $\mu$  is the average return of portfolio  $p$ ,
- $\sigma$  is the standard deviation of portfolio  $p$ ,
- $S$  is the skewness,
- $K$  is the kurtosis ( $K-3$  is the reduced kurtosis),
- $m$  is a parameter of order 1 that measures the risk-aversion of our investor, and
- $O(m^4\sigma^5)$  is the order of terms that are ignored.

Note that a positive skewness is good while a positive reduced kurtosis is bad because it results in fatter tails for fixed  $\sigma$ . It is also important to note that because of our choice of the CARA utility function, (13) does not converge unless all momenta of the distribution are finite. If this is not the case, for example for fat tail distributions, equation (13) does not produce a finite result and our approximate formula loses its value. For Gaussian-distributed returns equation (12) reduces to

$$R_*(p) = \mu - \frac{m\sigma^2}{2}, \quad (14)$$

and is exact. Note that the ranking (14) was originally proposed by Lévy and Markowitz (1979). Our approximated formula, equation (13), extends that result in the case of non-Gaussian returns.

We conclude that a rational risk-averse investor who has to choose one portfolio among many and has to invest all funds in the selected portfolio, should make their choice based on the ranking function in equation (12). This is a general result and it does not make any assumption concerning the distribution of the returns of the portfolios.

### 3. Practical considerations

Our conclusions have immediate practical applicability when the investor is not allowed to invest in a risk-free asset and has to choose one and only one mutually exclusive portfolio (or investment alternatives) according only to past performance. The following is an outline of the decision algorithm.

- For each portfolio  $k$ , collect the historical returns  $r_{kt}$  (return for portfolio  $k$  at time  $t$ ) and measure

$$\begin{aligned} \mu_k &= (1/N) \sum_t r_{kt}, \\ \sigma_k &= (1/N) \sum_t (r_{kt} - \mu_k)^2, \\ S_k &= (1/N) \sum_t (r_{kt} - \mu_k)^3 / \sigma_k^3, \\ K_k &= (1/N) \sum_t (r_{kt} - \mu_k)^4 / \sigma_k^4 \end{aligned}$$

(where  $N$  is the number of available data points).

- For each portfolio, compute

$$Rank_k = \mu_k - \frac{m\sigma_k^2}{2} + \frac{m^2\sigma_k^3 S_k}{6} - \frac{m^3\sigma_k^4(K_k-3)}{720}. \quad (15)$$

- Sort the portfolios according to  $Rank_k$  and select the one with the highest rank.

If  $S_k=0$  and  $K_k=0$ , our rank is equivalent to the formula proposed by Lévy and Markowitz (1979). Note that the rank depends on the time-scale of our analysis via our choice of returns  $r$ , which can be daily returns, weekly returns, monthly returns, etc. For a fixed skewness and kurtosis, their relative contribution to the rank increases

with the size of the time-scale. In any case, because of the 1/720 factor, the relative contribution of kurtosis is generally very small.

#### 4. Conclusions

In this paper we discuss the issue of portfolio selection for a rational risk-averse investor. We consider two cases: the investor is free to distribute funds between the selected portfolio and a risk-free asset, and the case when the investor has to invest all funds in the selected portfolio. In the first case we find that Sharpe, Omega, Sortino, and Kappa provide valid ranking schemes, although they follow from different definitions of risk,  $\sigma$ ,  $\sigma_1^-$ ,  $\sigma_2^-$  and  $\sigma_n^-$ , respectively. We also find that, in the Gaussian case, the Sharpe, Omega, Sortino, Kappa, and Stutzer rankings are all equivalent. In the non-Gaussian case, these rankings are not equivalent (Artzner *et al.* 2000, Ortobelli *et al.* 2005).

In the second case we find that, contrary to what is sometimes claimed, the use of any of the above ranking schemes does not correspond to being a rational risk-averse investor. In fact, we prove that a rational risk-averse investor (compatible with the choice of the CARA utility function) would choose a portfolio according to the following ranking function:

$$R_*(p) \stackrel{\text{def}}{=} -\log(-R_{U_{\text{CARA}}}(p))/m, \quad (16)$$

$$= \mu - \frac{m\sigma^2}{2} + \frac{m^2\sigma^3 S}{6} - \frac{m^3\sigma^4(K-3)}{720} + O(m^4\sigma^5), \quad (17)$$

which correctly takes into account the skewness of the portfolio,  $S$ , and its kurtosis,  $K$ , for distributions with finite momenta. Here,  $m$  is a coefficient of order 1 that measures the risk-aversion of the investor. Our approximated formula, equation (13), extends a result originally due to Lévy and Markowitz (1979) in the case of non-Gaussian returns. Our results do not apply to fat tail distributions, since some of their momenta are not finite and the integral in equation (13) does not converge. To take this case into account it is necessary to move beyond the CARA utility function. We will consider this case in a subsequent paper.

#### References

- Artzner, P., Delbaen, F., Heath, J.M. and Eber, D., Coherent measures of risk. *Math. Finance*, 2000, **9**, 203–228.  
 Horvath, P.A. and Scott, R.C., On the direction of preference for moments of higher order than variance. *J. Finance*, 1980, **35**, 915–919.  
 Kaplan, P.D. and Knowles, J.A., Kappa: A generalized downside risk-adjusted performance measure. *J. Perform. Measur.*, 2004, **8**(3), 42–54.
- Kazemi, H., Schneeweis, T. and Gupta, R., Omega as performance measure. *CISDM 2003 Proceedings*, 2003.  
 Lévy, H. and Markowitz, H., Approximating expected utility by a function of mean and variance. *Am. Econ. Rev.*, 1979, **69**, 308–317.  
 Markowitz, H.M., Portfolio selection. *J. Finance*, 1952, **7**(1), 77–91.  
 Nash, J.F., The bargaining problem. *Econometrica*, 1950, **18**, 155.  
 Neumann, J. von and Morgenstern, O., *Theory of Games and Economic Behavior*, 2nd ed., 1947 (Princeton University Press: Princeton, NJ).  
 Ortobelli, S., Rachev, S.T., Stoyanov, S., Fabozzi, F. and Biglova, A., The proper use of risk measures in portfolio theory. *Int. J. Theor. Appl. Finance*, 2005, **8**(8), 1107–1133.  
 Sharpe, W.F., Capital asset prices: A theory of market equilibrium under conditions of risk. *J. Finance*, 1964, **19**(3), 425–442.  
 Sortino, F.A., From alpha to omega. In *Managing Downside Risk in Financial Markets*, edited by F.A. Sortino and S.E. Satchell, 2001 (Reed Educational and Professional: London).  
 Sortino, F.A. and Forsey, H.J., On the use and misuse of downside risk. *J. Portfol. Mgmt*, 1996, **22**(2), 35–42.  
 Sortino, F.A. and Price, L.N., Performance measurement in a downside risk framework. *J. Invest.*, 1994, **3**(3), 59–64.  
 Sortino, F.A. and Van Der Meer, R., Downside risk. *J. Portfol. Mgmt*, 1991, **17**(4), 27–32.  
 Stutzer, M., A portfolio performance index. *Financial Anal. J.*, 2000, **56**, 52–61.

#### Appendix A:

It was shown by Ortobelli *et al.* (2005) that if  $p(x)$  is a Gaussian distribution with mean  $\mu$  and standard deviation  $\sigma$ , then the Sharpe, Sortino, Omega, and Stutzer ranking schemes are equivalent. In this appendix we provide explicit mapping formulas from these ranking schemes, and from a general Kappa, into the Sharpe ratio. Two rankings,  $R_1$  and  $R_2$ , are equivalent if and only if, for any two portfolios  $p_1$  and  $p_2$ ,  $R_1(p_1) < R_1(p_2)$  implies  $R_2(p_1) < R_2(p_2)$  and vice versa. This can only occur if there is a monotonic increasing function  $h$  such that, for any portfolio  $p$ ,  $R_2(p) = h(R_1(p))$ .

#### Kappa

#### Proof:

$$R_{\text{Kappa}-n}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{[\int_{-\infty}^r (r-x)^n p(x) dx]^{1/n}} = h(R_{\text{Sharpe}}(p)) \quad (\text{A1})$$

and

$$h(y) = \frac{\pi^{1/2n} 2^{(2-n)/2n} e^{y^2/2n} y}{\left\{ \begin{array}{l} [\Gamma((1+n)/2) {}_1F_1((1+n)/2, 1/2, y^2/2) \\ - \sqrt{2}\Gamma(1+(n/2)) {}_1F_1(1+(n/2), 3/2, y^2/2) \end{array} \right\}^{1/n}}$$

( $\Gamma$  is Euler's Gamma function and  ${}_1F_1$  is a hypergeometric function).  $\square$

***Omega*****Proof:**

$$R_{\text{Omega}}(p) \stackrel{\text{def}}{=} \frac{\int_r^\infty (1 - F_p(x)) dx}{\int_{-\infty}^r F_p(x) dx} = R_{\text{Kappa}-1}(p) \quad (\text{A2})$$

(here  $F_p$  is the cumulative distribution function associated with  $p$ ).  $\square$ 

returns  $p$ . Let  $\mu, \sigma, S$  and  $K$  be the average, standard deviation, skewness and kurtosis of  $p$ ,

$$R_{U_{\text{CARA}}}(p) \stackrel{\text{def}}{=} \int_{-\infty}^\infty -e^{-mx} p(x) dx = \int_{-\infty}^\infty -e^{-mx} \tilde{p}((x - \mu)/\sigma) dx,$$

where  $\tilde{p}(y) = p(\sigma y + \mu)\sigma$ . With the change of variable  $y = (x - \mu)/\sigma$  and a Taylor series expansion in  $y$  of the exponential, we obtain

$$\begin{aligned} R_{U_{\text{CARA}}}(p_i) &= -e^{-m\mu} \sum_{i=0}^{\infty} \frac{(-mo)^i}{i!} \int_{-\infty}^\infty y^i \tilde{p}(y) dy \\ &= -e^{-m\mu} e^{\log(1 + \frac{m^2\sigma^2}{2} - \frac{m^3\sigma^3 S}{3!} + \frac{m^4\sigma^4(K-3)}{6!} + O(m^4\sigma^5))} \\ &= h(R_*(p)), \end{aligned} \quad (\text{B1})$$

***Sortino*****Proof:**

$$R_{\text{Sortino}}(p) \stackrel{\text{def}}{=} \frac{\mu - r}{[\int_{-\infty}^r (r - x)^2 p(x) dx]^{1/2}} = R_{\text{Kappa}-2}(p). \quad (\text{A3})$$

***Stutzer*****Proof:**

$$R_{\text{Stutzer}}(p) \stackrel{\text{def}}{=} \lim_{T \rightarrow \infty} \frac{-\log F_p(rT)}{T} = h(R_{\text{Sharpe}}(p)), \quad (\text{A4})$$

where  $R_{\text{Stutzer}}$  is well-defined only for portfolios with positive Sharpe ratio, and

$$h(y) = y^2/2 \quad (\text{for } y > 0 \text{ only}). \quad (\text{A5})$$

 $\square$ **Appendix B:**

In this section we prove that  $R_{U_{\text{CARA}}}$  is equivalent to  $R_*$ . Consider a portfolio characterized by a distribution of

where  $h(y) = -e^{-my}$  is a monotonic increasing function in  $y$ , and

$$R_*(p) \stackrel{\text{def}}{=} -\log(-R_{U_{\text{CARA}}}(p))/m \quad (\text{B2})$$

$$= \mu - \frac{m\sigma^2}{2} + \frac{m^2\sigma^3 S}{6} - \frac{m^3\sigma^4(K-3)}{720} + O(m^4\sigma^5). \quad (\text{B3})$$

Hence the two ranking functions  $R_*(p)$  and  $R_{U_{\text{CARA}}}(p)$  are equivalent. This is a general result and no assumption concerning the distribution  $p$  has been made. In the special case of  $p$  Gaussian, we are able to perform the integration analytically without the need for a Taylor expansion and we find that the following exact relation holds (Lévy and Markowitz 1979):

$$R_*(p) = \mu - \frac{m\sigma^2}{2}. \quad (\text{B4})$$

Copyright of Quantitative Finance is the property of Routledge and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

# Making QCD Lattice Data Accessible and Organized through Advanced Web Interfaces

POS(Lattice 2011)305

**Massimo Di Pierro\***

*School of Computing - DePaul University - Chicago*

*E-mail:* [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu)

**James Hetrick**

*University of the Pacific*

*E-mail:* [jhetrick@pacific.edu](mailto:jhetrick@pacific.edu)

**Shreyas Cholia**

*Lawrence Berkely Laboratory*

*E-mail:* [scholia@lbl.gov](mailto:scholia@lbl.gov)

**David Skinner**

*Lawrence Berkely Laboratory*

*E-mail:* [deskinner@lbl.gov](mailto:deskinner@lbl.gov)

The Gauge Connection at [qcd.nersc.gov](http://qcd.nersc.gov) is one of the most popular repositories of QCD lattice ensembles. It is used to access 16TB of archived QCD data from the High Performance Storage System (HPSS) at the National Energy Research Scientific Computing Center (NERSC). Here, we present a new web interface for [qcd.nersc.gov](http://qcd.nersc.gov) which allows physicists to browse and search the data, as well as download individual files or entire ensembles in batch. Our system distinguishes itself from others because of its ease of use and web based workflow.

*XXIX International Symposium on Lattice Field Theory*  
*July 10-16 2011*  
*Squaw Valley, Lake Tahoe, California*

---

\*Speaker.

## 1. Introduction

The Gauge Connection Lattice QCD data archive has been operated by the National Energy Research Scientific Computing [6] (NERSC) center since 1998. With over 16TBytes of data, the NERSC archive is one of the largest public repositories of lattice gauge ensembles. Its popularity has largely been due to the ease-of-use of its web interface compared to more complex grid based tools.

We are now updating the Gauge Connection to provide a number of modern features, while maintaining the simplicity and ease-of-use of the original. We are adding the ability to search data using tags. We are also providing capabilities to move data easily between the archive and the user's computer, or between two remote computers using both web and grid tools. The archive is now database driven and its pages are dynamically generated in order to facilitate access to the most recent local and remote data. It provides some visualization of the data. New data can easily be uploaded to the archive where it is automatically discovered and cataloged.

The archive can now deliver data in a variety of file formats (including ILDG [5], single and double precision, and FermiQCD [3]) by translating data on the fly after download. We have designed the new website to provide additional capabilities beyond data access, including wiki capabilities to annotate the data and link external work derived from archive data (derived data, software, tutorials, and publications).

Each gauge ensemble is also associated with a discussion forum allowing registered users to add their own comments. The archive has a more sophisticated access control mechanism and users can have four possible roles: administrators (can manage every aspect), editors (can edit the wiki pages linked to the data), registered users (can download data and comment) and anonymous visitors (can browse the wiki pages and query the data by tags). User management is greatly simplified through the use of a single-sign-on service that allows the user to log in with an existing identity from an external OpenID provider like Google or Yahoo.

## 2. Backend Architecture

The NERSC data is stored on the High Performance Storage System (HPSS), a hierarchical tape storage system designed to manage and access multiple petabytes of data at high speeds [?]. HPSS stores 16 terabytes of Lattice QCD data, mostly generated by the MILC [4] Collaboration. HPSS exposes different data access interfaces including a custom command line client (HSI), a C API, an FTP interface, a parallel FTP interface, as well as a GridFTP interface that can be used via Globus Online. We leverage the HPSS FTP interface to provide web access to the Lattice QCD data.

The system performs nightly introspection of the folder structure (through the HSI and FTP interfaces), reproduces this folder structure in the database, automatically tags data by parsing the file names, groups files with similar name patterns, and publishes the data online. For each folder in HPSS a corresponding web page is generated dynamically. The web structure has the same hierarchy as the folder structure on the HPSS backend.

The new web interface is designed to mimic the iPhone interface and works on both regular browsers and mobile devices.

PoS (Lattice 2011) 305



**Figure 1:** The main web page for `qcd.nerc.gov` allows browsing and searching for gauge ensembles.

**Figure 2:** The left image show statics by tags. The right image shows topological change images from representative elements of various selected ensembles.



**Figure 3:** The image shows the page corresponding to a folder. The name of the folder is on top, followed by a folder description (optional) and by the tags applied to the folder. Files in the folder are grouped by their prefix (pattern). Patterns are shown at the bottom, above users' comments.

Every page on the QCD site is editable using a wiki syntax similar to wikipedia and registered users can comment on the pages. Both the wiki and comments allow Latex syntax for formulas. Gauge ensembles can be searched by direct browsing of the folder structure or by tag (contributing collaboration, lattice size, beta value, dynamical quark masses, etc.). The system generates interactive charts with statistics by tags.

Some ensembles have been processed off-line to generate meta-data such as a topological charge densities and have been linked to images and movies of said topological charge density. The system is data agnostic and can, in principle, store other data besides gauge ensembles. For example, it can store quark propagators and eigenvalues, examples of which are already in the system. The system uses standard third party web analytics tools to track usage and to geo-tag visitors on a map.

Here is how a simple file access workflow works. The user logs in to the Gauge Connection website (fig. 1) and searches for a desired dataset (fig. 2). When the user finds the ensemble that he/she wants (fig. 3), the user click on the appropriate file to download it. The system checks that the user is logged in, and then pulls the data from tape through the HPSS FTP server. Once the data is available (the tape has been mounted in HPSS), it is streamed back directly to the client browser. Alternatively the user can copy the link to the page for the entire ensemble and pass it to a command line script (provided) that will download every file in the ensemble, one by one, in batch, converting on the fly to the desired format.

### 3. Frontend Architecture

The system is based on web2py [2], a framework for rapid development of secure database driven web applications. It is written in Python and supports many standard databases including Sqlite, MySQL, PostgreSQL, Oracle, MSSQL, Informix, DB2, Sybase, Firebird, and Google Bit Table using a Database Abstraction Layer (DAL). The DAL generates SQL dynamically and as needed.

The system uses jQuery Mobile for page layout and a custom JavaScript library for charting. It uses the Google Chart API for Latex rendering and the Janrain web service for Single Sign On.

The visualizations of topological charge density are produced offline using Visit (LLNL) and the FermiQCD Visualization Toolkit.

The system has a modular Model-View-Controller design which separates the data representation from the data presentation and from the application workflow. This makes the code compact and easy to maintain. It includes a web based IDE, a web based database management tool and internationalization capabilities.

The complete model for the system consists of just a few lines of code used to describe the data that is stored:

```

1 # represents a folder in HPSS
2 db.define_table('catalog_folder',
3     Field('root_id', 'reference catalog_folder'),
4     Field('path'),
5     Field('title'),
6     Field('header', 'text'),
7     Field('footer', 'text'),
8     Field('pattern_ignore'),
9     Field('pattern_group'),
10    Field('comments', 'boolean', default=True))
11
12 # a tag to be applied to a folder
13 db.define_table('tag',
14     Field('name'),
15     Field('root_id', 'reference catalog_folder'))
16
17 # a file contained in a folder
18 db.define_table('catalog_file',
19     Field('root_id', 'reference catalog_folder'),
20     Field('filename'),
21     Field('md5'),
22     Field('pattern'),
23     Field('extension'),
24     Field('size', 'decimal(20,0)'),
25     Field('mtime', 'datetime'))
```

Each Field has a name and a type.

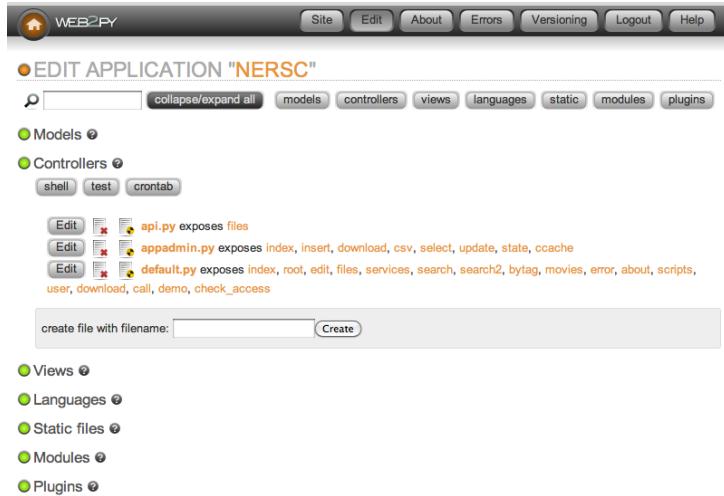
A dispatcher function maps web pages into function calls. These functions are defined in the controller. For example, all web pages of the form

```
http://.../root/<path>
```

Correspond to a <path> in the file system and they are mapped into the following function:

```

1 @cache(request.env.path_info, 60)
2 def root():
3     path = '/'.join(request.args)
4     (d,t,f) = (db.catalog_folder, db.tag, db.catalog_file)
5     query = (d.path==path) if path else (d.root_id==0)
6     page = db(query).select().first() or redirect(URL('error'))
7     tags = db(t.root_id==page.id).select()
8     subfolders = db(d.root_id==page.id).select()
9     patterns = db(f.root_id==page.id).select(
10         f.pattern,f.id.count(),f.size.sum(),
11         f.mtime.year(), groupby=f.pattern,orderby=f.pattern)
12     return locals()
```



**Figure 4:** The system administrative interface.

This code parses the URL in the request for the path, defines some local variables ( $d, t, f$ ) referencing the tables, builds a query to fetch the content of the folder: the path, the tags, the subfolders and the patterns (i.e. groups of files with the same prefix). At statements follow the DAL syntax:

```
1 variable = db(query).select(what,how)
```

The `@cache(...)` decorator instructs the framework to cache each page for 60 seconds. This means the if a page is requested more than once every 60 seconds, it will not hit the database more than one time. This results in better performance.

The system enforces different types of access control. It provides a web based administrative interface only accessible to managers (fig. 4). All downloads require user login.

The system itself is not domain specific and has no knowledge about QCD files and conventions. The domain specific knowledge is in a separate script that runs in background and populates the database from the file/folder structure. This means the system can be used to publish data of a different nature with minimal work. The system retains the original ability to download individual files using the web interface. Since these files can be big, the download requires login in order to prevent Denial of Service attacks.

The system exposes RESTful web services APIs based on the JSON protocol which can be used to access the data programmatically. Some gauge ensembles are comprised of thousands of files. To download them in batch we provide a script `qcdutils_get.py`, written in Python with no dependencies, available from [1].

In this example we use the script to download all files in the `demo` ensemble:

```
1 $ qcdutils_get.py http://qcd.nersc.gov/nersc/api/files/demo
2 http://qcd.nersc.org/nersc/api/files/demo
3 target folder: demo
4 total files to download: 1
5 downloading demo.nersc
```

```
6 demo.nersc 100% |#####
7 completed download: 1/1
```

We now ask the script to convert all data from NERSC to ILDG(32bits):

```
1 $ qcdutils_get.py --convert ildg --float demo/demo.nersc
2 converting: demo/demo.nersc -> demo/demo.nersc.ildg
3   (precision: f, size: 4x8x8x8)
4 100% |#####
```

`qcdutils` keeps an internal log of all the operations completed to avoid duplication of work. It remembers what was downloaded/converted and where things are. We can ask `qcdutils` for a log of the completed tasks:

```
1 $ qcdutils_get.py demo/qcdutils.catalog.db
2 demo.nersc created on 2011-06-17T13:42:30.876812
3   [14e7cf9106bfc16388aeac285ccdad9]
4 demo.nersc.ildg created on 2011-06-17T13:42:34.424604
5   [5a1ae13ddd5cab7ddfe1b17454822ff5]
```

## 4. Conclusions and Outlook

The system allows the user to register any URL and dynamically generates buttons that, when pressed, pass a link to the data at the associated URL. This allows for the creation of third party web services that can feed data directly from the new NERSC web interface allowing for decentralization of services. We can provide tools to help create such services. For example, we can build workflows that interface the data archive with a batch queue on a large computational system.

We envision a future when the different research groups will provide their computing capabilities and their lattice QCD algorithms as web services for the consumption of other members in the collaboration. The NERSC site provides more than just data for these collaborations - it also offers an infrastructure to register those third party services in a transparent way.

### Acknowledgements

This work was funded by Department of Energy grant DEFC02-06ER41441 and by National Science Foundation grant 0970137.

## References

- [1] Massimo Di Pierro. `qcdutils`, 2010.
- [2] Massimo Di Pierro. `web2py` website, 2010.
- [3] Massimo Di Pierro and Jonthan M. Flynn. Lattice QFT with FermiQCD. *PoS*, LAT2005:104, 2006.
- [4] S. Gottlieb. Benchmarking and tuning the MILC code on clusters and supercomputers. *Nuclear Physics B Proceedings Supplements*, 106:1031–1033, 2001.
- [5] C. Maynard. International Lattice Data Grid: Turn on, plug in, and download. In *Symposium on Lattice Field Theory*, 2009.
- [6] U.S. DOE. Nersc website, 2010.

---

## Pattern derivatives

---

Casey S. Schroeder\* and Massimo Di Pierro

School of Computing,  
DePaul University,  
CDM Center, 243 South Wabash Avenue,  
Chicago, IL, USA  
E-mail: cs@csschroeder.com  
E-mail: mdipierro@cs.depaul.edu

\*Corresponding author

**Abstract:** In this paper, we propose a new type of derivative called a pattern derivative. In the simple case of an asset that moves up  $u$  or down  $d$  in value, a pattern is a sequence of  $\{u, d\}$  movements that may occur before expiration. We provide general pricing formulas that rely on a brute force approach as well as efficient valuation algorithms based on recursive formulas for the probability of patterns to occur. We generalise our results to exclusive, inclusive, and multi-pattern options. Finally, we discuss the possible benefits of this type of options.

**Keywords:** patterns; derivatives; pattern derivatives; options; pricing; options pricing.

**Reference** to this paper should be made as follows: Schroeder, C.S. and Di Pierro, M. (2011) 'Pattern derivatives', *Int. J. Financial Markets and Derivatives*, Vol. 2, No. 4, pp.249–257.

**Biographical notes:** Casey S. Schroeder is Researcher and Consultant with interests in the fields of rational decision making and leveraging technology for better decision making. He obtained his Master in Philosophy from The University of California at San Diego and Master of Science in Computational Finance from DePaul University in Chicago.

Massimo Di Pierro is an Associate Professor at the School of Computing and Digital Media of DePaul University. He is Director of the Master of Science in Computational Finance and expert in large scale numerical applications of computing in physics and finance. He is the author of many open source libraries including web2py.com and fermiqcd.net. He obtained his PhD in High Energy Particle Physics from the University of Southampton in UK.

---

### 1 Introduction

In this paper, we propose a new type of path-dependent derivative, which we call *pattern derivatives*. Pattern derivatives are a family of derivatives characterised by the fact that payouts are dependent upon patterns of sequential, discrete events occurring or failing to occur during the life of the contract.

By a pattern, we mean a string of symbols from an arbitrary but finite alphabet. For example, *bac* is a pattern from the English alphabet and it occurs in the words ‘abacus’ and ‘backward’. Although we will try to be as general as possible, our numerical examples will focus on an alphabet of two symbols  $\{u, d\}$  where  $u$  may represent a positive daily return and  $d$  a negative daily return for an asset (for example a stock, a portfolio, or an index). Patterns such as *dddd*, *uuuu*, *ududud* and *ddudu* will represent patterns of periodic returns on an underlying asset. Most of our results are general and not dependent on this choice of an alphabet.

One particular kind of pattern derivative is a very simple digital option which triggers a one time payout to the holder of the option in the event that the underlying goes down for four consecutive market days (pattern *dddd*) during the life of the contract. Supposing that there are thirty market days until the expiry of the contract, the underlying may have the following description between now and expiration:

*uudduddudududddduuudududuudu*

in this case, the bold **d** indicates the point at which the payout is triggered. The payout can be made at any point after the triggering pattern occurs. We can assume, as for a European style option, that the payout is made at the end of the contract.

Patterns can be used instead of a barrier, in any type of barrier option. Pattern derivatives, however, may offer some protection to option writers where barrier options fail. For instance, many down-side barriers on US options were hit during the flash-crash of May 2010, though there was no persistence under these barriers. One could instead write in the contract that these barriers must be breached at close. But should such a crash happen at the end-of-day, where we assume that the markets would again be quickly buoyed by media and government intervention, such conditions would provide no such protection. A pattern option or hybrid pattern-barrier option specifying extended negative returns may help alleviate these risks to writers, while keeping to the spirit of the agreement.

Pattern derivatives are generally very useful in handling known problematic cases. Should a financial company be aware that a certain pattern of returns across one or more holdings would cause significant trouble, they can purchase a pattern derivative to cover this specific case. Furthermore these derivatives are quite useful in cases where the underlying is naturally discrete. For example, while it may be the case that the Fed can change interest rates by any continuous value they would like, one may have good reason to think that they will continue to make their changes in intervals of a quarter of a point. This case is quite amenable to treatment with pattern derivatives, with letters representing quarters of a point change. Finally, and perhaps most importantly, this methodology clearly applies outside of the specifically financial realm, to that of insurance and real options as well. Consider, for example, the case of being dependant on a supplier unable to deliver for many consecutive days, resulting in a shortage of supplies.

In the following sections, we will provide an explanation of patterns and their properties. We explain why these apparently elementary derivatives have somewhat counterintuitive valuations and we will provide details of the calculations which should be used in their valuation. We then make strides towards a fully general treatment and discuss possible extensions and further generalisations.

## 2 Numerical considerations

To be clear, we need to distinguish between the *inclusive* and *exclusive* use of patterns. Inclusive patterns are patterns that are required to appear during the life of the option (as with *dddd*), and exclusive patterns are patterns that are required not to appear. To start, we will concentrate on inclusive patterns and an alphabet  $\{u, d\}$  where each symbol has equal probability. We will indicate with  $L$  the size of our alphabet; in the case of  $\{u, d\}$ ,  $L = 2$ . We define  $P(x, n)$  as the probability of a pattern  $x$  occurring in a random sequence of length  $n$  (from our alphabet). Here are some numerical results for the  $P(x, n)$  from a brute force approach:

$n$	$u$	$d$	$uu$	$ud$	$du$	$dd$
1	1/2	1/2	0/2	0/2	0/2	0/2
2	3/4	3/4	1/4	1/4	1/4	1/4
3	7/8	7/8	3/8	4/8	4/8	3/8
4	15/16	15/16	8/16	11/16	11/16	8/16
5	31/32	31/32	19/32	26/32	26/32	19/32
6	63/64	63/64	43/64	57/64	57/64	43/64
7	127/128	127/128	94/128	120/128	120/128	94/128
8	255/256	255/256	201/256	247/256	247/256	201/256
9	511/512	511/512	423/512	502/512	502/512	423/512

The first column is the total length of the strings from  $\{u, d\}$ , that we are generating. The fractions represent the fraction of those strings that contain the corresponding pattern in the table head. For example: 19 out of 32 strings of length 5 contain the pattern *ud*. The first thing to observe, and rather obvious, is that the totals for pattern *u* are the same as for pattern *d* (and in this case, are equivalent to the total number of strings of length  $n$ , minus the one string containing all *ds* or all *us*, respectively). It is also obvious that, because of symmetry, the number of occurrences of pattern *uu* and *dd* is the same for any given  $n$  (as with *ud* and *du*). What is not obvious is that the number of occurrences of *uu* is consistently lower than the number of occurrences of *ud*.

Now we consider a market interpretation of the above data, the case of a stock price and its daily returns. Compatibly with Black-Scholes we can assume that daily returns are independent and normally distributed random variables with average zero; therefore the stock has the same probability of going up or down on each day. Under this interpretation of *u* and *d*, the above result tells us that the probability that in the next  $n$  days the stock will go up for two consecutive days is lower than the probability that it will go up one day and down the second.

Considering now all patterns of three days. The probability of those patterns occurring over a period of  $n$  days is:

<i>n</i>	<i>uuu</i>	<i>uud</i>	<i>udu</i>	<i>udd</i>
1	0/2	0/2	0/2	0/2
2	0/4	0/4	0/4	0/4
3	1/8	1/8	1/8	1/8
4	3/16	4/16	4/16	4/16
5	8/32	12/32	11/32	12/32
6	20/64	31/64	27/64	31/64
7	47/128	74/128	63/128	74/128
8	107/256	168/256	142/256	168/256
9	238/512	369/512	312/512	369/512

Similar to *uu* and *ud* in our prior example, the probability of *uuu* is lower than *uud*, and because of symmetry, the probability of *udd* is the same as *uud*. It is notable, however, that the probability of up-down-up (*udu*) is also lower than the probability of up-up-down (*uud*). These differences make a sizable contribution to the proper valuation of pattern derivatives, and as we shall see, are accountable to the fact that *uuu* and *udu* are patterns which overlap with themselves.

### 3 Pricing digital one-pattern-options

Continuing with our basic case, we can consider a European digital option that pays  $A = 1$  dollar at expiration (after  $n$  days) if during the life of the option a pattern occurs (an inclusive use). We assume  $N = 360$  days/year and a yearly 5% interest rate,  $r$ . We can use the following basic formula for pricing our pattern derivatives with one inclusive pattern:

$$C(x, n) = \frac{1}{L^n} \sum_{s \in S_n} A_f(s, x) e^{-rn/N} \quad (1)$$

where  $C(x, n)$  is the price of the derivative,  $x$  is the pattern,  $S_n$  is a set of all possible sequences of  $u$  and  $d$  of length  $n$  and  $f(s, x)$  is a function that returns 1 if  $x$  appears in  $s$  and 0 otherwise. We compute the prices  $C$  for different options of different maturity and different patterns as follows:

<i>n</i>	<i>u</i>	<i>d</i>	<i>uu</i>	<i>ud</i>	<i>uuu</i>	<i>uud</i>
1	0.4999	0.4999	0.0000	0.0000	0.0000	0.0000
2	0.7498	0.7498	0.2499	0.2499	0.0000	0.0000
3	0.8746	0.8746	0.3748	0.4998	0.1249	0.1249
4	0.9370	0.9370	0.4997	0.6871	0.1874	0.2499
5	0.9681	0.9681	0.5933	0.8119	0.2498	0.3747
6	0.9836	0.9836	0.6713	0.8899	0.3122	0.4840
7	0.9912	0.9912	0.7337	0.9366	0.3668	0.5776
8	0.9950	0.9950	0.7843	0.9638	0.4175	0.6555
9	0.9968	0.9968	0.8251	0.9792	0.4643	0.7198

And as we expect, fixing an arbitrary  $n$  (time until expiry), the shorter the pattern, the more expensive the option; since the shorter the pattern, the less that has to ‘go right’. On the other hand, the options involving patterns which overlap with themselves are generally worth less than patterns of the same length and time to expiration, which are *not self-overlapping*.

Similarly, we can price a European call option involving a single exclusive pattern with the formula:

$$C(y, n) = \frac{1}{L^n} \sum_{s \in S_n} A(1 - f(s, y)) e^{-rn/N} \quad (2)$$

where we use the letter  $y$  to indicate an exclusive pattern instead of the letter  $x$  which we used for inclusive patterns.

We can also define a US digital (inclusive) pattern option as one that will be exercised (and therefore pays the \$1) the moment the pattern is expressed. For this purpose, we define a function  $g(s, x)$  that returns 0 if the pattern  $x$  is not in  $s$  or the position of the pattern as an integer number if  $x$  is contained in  $s$ . In this case, the pricing formula for a US digital pattern option is:

$$C(x, n) = \frac{1}{L^n} \sum_{s \in S_n} A \min(g(s, x), 1) e^{-rg(s, x)/N} \quad (3)$$

The difference between European and US options is negligible for ten days but it becomes sizable and the effect we are showing is even greater for US options, because if a pattern is more likely to occur, it is also more likely to occur sooner, and this affects the timing of the payoff.

At this point generalisations are obvious.

While the above formulas are theoretically sound, looping over  $S_n$  becomes computationally expensive for very large  $n$  and already impractical for  $n > 20$ , with an alphabet of just two symbols. Analytical results are therefore needed. Many analytical results about probability theory of pattern occurrence can be found in Blom (1982) and Blom and Thorburn (1982) who build off the work of Feller (1968) and others. According to Blom and Thorburn (1982) and their Theorem 2.1, the probability  $P(x, t)$  defined as the probability that a pattern  $x$  matches for the first time at  $t$  in an arbitrary sequence of length  $n$ , can be given by the following recursive relation:

$$P(x, t) = \frac{1}{L^k} \left( 1 - \sum_{j=k}^{t-k} P(x, j) - \sum_{j=t-k+1}^{t-1} P(x, j) \varepsilon_{k-t+j}(x) L^{k-t+j} \right) \quad (4)$$

Here,  $k$  is the length of pattern  $x$ . The first part of the equation is just the probability that the pattern occurs at  $t$ , whether or not it is the first occurrence. To help determine the probability of it being the first occurrence we must subtract out the second term, which is the probability of it occurring for the first time at a previous, non-overlapping time. If a pattern does not overlap with itself the third term is zero. But if it does, the function  $\varepsilon_j(x)$  indicates the amount of the overlap. It returns one just in case the pattern overlaps with itself at an overlap of size  $j$ . For instance, for  $v = uudduu$ ,  $\varepsilon_1(v) = \varepsilon_2(v) = 1$ , and  $\varepsilon_j(v) = 0$  for every other  $j$ ; and for  $v = uuuu$ ,  $\varepsilon_j(v) = 1$  for all  $j$ ,  $1 \leq j \leq 4$ .

With this fundamental formula, the valuations for our basic pattern-options become straightforward and we can rewrite equation (1) for inclusive European digital options as:

$$C(x, n) = Ae^{-rn/N} \sum_{t=k}^n P(x, t) \quad (5)$$

equation (2), for exclusive European digital, as:

$$C(x, n) = Ae^{-rn/N} \left(1 - \sum_{t=k}^n P(x, t)\right) \quad (6)$$

and equation (3), for inclusive US digital, as:

$$C(x, n) = A \sum_{t=k}^n (P(x, t)e^{-rt/N}) \quad (7)$$

Notice that the brute force approach to the computation of equation (1) has a running time of  $O(L^n)$  and it becomes unfeasible for  $n > 20$ . The exact formula in equation (5) instead has a running time of  $O(n^3)$  and it is very efficient even for large values of  $n$ .

There are three fundamental generalisations to these basic results which must be addressed to achieve sufficient generalisation of the basic case. These are, respectively, the multi-pattern case, the mixed-pattern case, and the market-tradable case.

#### 4 Multi-pattern options

Multi-patterns allow for the possibility that an inclusive-pattern-option (or an exclusive-pattern-option) may involve more than one pattern. That is, the condition of the option is met just in case one or more of the patterns occur (or in the exclusive case, none of the patterns occur). First of all, we observe that patterns may overlap with themselves and with each other, therefore the pricing cannot be a linear function of the pricing of the individual patterns. We will also define  $Q_e$  as a set of exclusive patterns and  $Q_i$  as a set of inclusive ones.

From the definition of digital option, we generalise equation (1) for multi-pattern inclusive digital as:

$$C(Q_i, n) = \frac{1}{L^n} \sum_{s \in S_n} \theta\left(\sum_{x \in Q_i} f(s, x)\right) Ae^{-rn/N} \quad (8)$$

and we generalise equation (2) for multi-pattern exclusive digital as:

$$C(Q_e, n) = \frac{1}{L^n} \sum_{s \in S_n} \left(1 - \theta\left(\sum_{y \in Q_e} f(s, y)\right)\right) Ae^{-rn/N} \quad (9)$$

(Here  $\theta(\dots)$  is 1 if the argument is greater than zero, and zero otherwise.) These valuations from brute force are clearly more computationally intensive than the basic case. Thankfully, Blom and Thorburn (1982) provide an analytical result that allows us to write a general formula for the probability of  $x$  matching at  $t$  and no other prior matches from  $Q$  as:

$$P(x, t) = \frac{1}{L^k} \left( 1 - \sum_{j=k}^{t-k} \sum_{z \in Q} P(z, j) - \sum_{j=t-k+t}^{t-1} \sum_{z \in Q} P(z, j) \varepsilon_{k-t+j}(z, x) L^{k-t+j} \right) \quad (10)$$

Here, the formula is modified only in order to subtract out the probability of the other patterns occurring at a prior time as well. We do this by including summation over all the  $z$  in the set of patterns  $Q$ , and modifying  $\varepsilon_j(z, x)$  to take two parameters, accounting for overlap between a pattern and itself as well as other patterns. It is clear that we can use this formula for the multi-pattern generalisation, in all of the types of options we have as yet considered. It is notable, however, that this only provides a generalisation to the multi-pattern case where the patterns are of constant length  $k$ .

Now, we can rewrite equation (8) as:

$$C(Q_i, n) = Ae^{-rn/N} \sum_{t=k}^n \sum_{x \in Q_i} P(x, t) \quad (11)$$

and equation (9) as:

$$C(Q_e, n) = Ae^{-rn/N} \left( 1 - \sum_{t=k}^n \sum_{y \in Q_e} P(y, t) \right) \quad (12)$$

## 5 Mixed-pattern options

Mixed-pattern options allow us to address the possibility that a single pattern-option may have both a multi-inclusive and a multi-exclusive clause. Consider an option that pays a dividend when one of some  $x$  inclusive patterns is expressed and none of some  $y$  exclusive patterns are expressed:

$$C(Q_i, Q_e, n) = \frac{1}{L^n} \sum_{s \in S_n} \theta \left( \sum_{x \in Q_i} f(s, x) \right) \left( 1 - \theta \left( \sum_{y \in Q_e} f(s, y) \right) \right) Ae^{-rn/N} \quad (13)$$

This is the most general formula for a digital option including both inclusive and exclusive patterns. This is (potentially) twice as complicated as the basic multi-pattern cases, so we again are in need of analytic results. For the mixed case, however, analytical results are more difficult to come by. We must combine the cases in a valid way, which considers not simply the overlap of an inclusive (exclusive) pattern with another inclusive (exclusive) pattern, but also accounts for the overlap of inclusive (exclusive) patterns with exclusive (inclusive) patterns. We will discuss this case in a following work.

## 6 Market tradable pattern options

For any options to be market-tradable, we need to address how the valuation of a pattern option may evolve during the life of an option, considering the fact that a relevant pattern may already be imminent. For instance, if the relevant option involves the pattern *uduud* and the prior four days returns provided *uduu*, we must consider the fact that it is more

likely to strike the pattern now than it was four days ago, for the fact that now only two sequential events need to take place. The inclusion of this case is straightforward, although it introduces an additional level of complexity.

If we identify with  $w$  the pattern that is already expressed in the recent past, and we define  $P(v, t | w)$  as the probability of pattern  $v$  occurring for the first time at  $t$  given the past history  $w$ , we can write a recursive formula for  $P$  using:

$$P(v, t | w) = \varepsilon_{k-t}(w, v) \left( L^{-t} - \sum_{j=1}^{t-1} \sum_{u \in Q} P(u, j | w) \varepsilon_j(u, v) L^{-t+j} \right) \quad (14)$$

for  $0 < t < k$  and the previous formula (10), with suitable alterations, when  $t \geq k$ . Should one wish to trade a pattern option mid-term, this formula must be used with  $w$  being the previous  $k-1$  events or how many events have already taken place, whichever is less. It is notable that this formula is a generalisation of the multi-pattern case and makes no special considerations precluding the use of mixed-patterns.  $P(v, t | w)$  can safely be replaced into equation (12) and equation (11).

## 7 Conclusions and outlook

We have thus far considered a simple coupon payout. It should be clear to the reader that any sort of instrument which can be valued at a future time can be substituted for these coupon payments. As such, the  $A$  in our formulas may represent an asset of any variety or it may itself be an option, as found in certain types of barrier options (knock-in options) and forward starting options (Bouzoubaa and Osseiran, 2010).  $A$  may also be negative and used in conjunction with standard models of valuation to indicate a loss when a trigger is met (knock-out options) (Bouzoubaa and Osseiran, 2010).

There are furthermore natural changes that can be made to the payout structure, where a coupon payment is made or accrued for each occurrence of the pattern, for instance *ddd*. These can come in two varieties. In the first variety, the payment is made given every occurrence; in this case, a *d* immediately following three *d*'s in a row, would trigger another coupon payment, for the fact that it also has two *d*'s immediately preceding it. In the second variety, the payment is made given every fresh occurrence; in this case, the trigger resets itself at *d*, and awaits three fresh *d*'s in a row. In the first case, we can very easily give an equation for the expected value of the option, using the coupon and the mean number of occurrences per series, if we assume payment is made at the end of the term. The mean number of occurrences per series is simply  $(n - k + 1) * L^{-k}$ . Given the considerations above, it is sometimes also a surprising fact that this value is the same for all patterns of length  $k$ , given time to expiry  $n$ , and the size of the alphabet  $L$ , regardless of overlap (though here assuming equal probabilities). However, when there is much overlap, patterns will tend to be struck in bunches; this should be clear from the fact that in the case of *ddd*, it only takes one more *d*, and not three, to initiate another payout. So the distribution of payouts will have a considerably greater variance. In the second case, the expected value is considerably less for patterns with high overlap, so these options should be considerably cheaper than those involving non-overlapping counterparts (in the inclusive use) (Chryssaphinou and Papastavridis, 1988).

And finally, there are very few restrictions on the kinds of underlying and partitions which may be assigned to our alphabet. In our construction of the partition of the underlying distribution, the choice of a two letter alphabet was not essential. All of the formulas here mentioned thus far rely on an alphabet of arbitrary size  $L$ , and as a computational matter, increasing the size of  $L$  does not effect time complexity to any problematic degree. This allows, for instance, one to slice the distribution of returns in the way that uses patterns of extreme returns in an option, for the sake of handling volatility, for instance. Similarly, it appears that it is not necessary that each letter in the alphabet have equal probability of occurring. And it does not appear to be necessary that the successive letters even be independently distributed. Formulas are given by Robin and Daudin (1999) for the case of using a non-uniform distribution over the letters and for the first-order Markovian case. (Though it bears repeating that the influence of overlap does not disappear.) And finally, it is clear that there is nothing special about using the partition of the probability distribution of a single variable, as in the case of returns we consider here. Should we have a joint probability distribution over two or more variables, then we can partition it in the assignment of letters to events. This more general case has clear applications in the area of *perfect storms*.

We have here only addressed the most basic logical categories of pattern derivatives, but we find that they are fundamental and elegant to handle. We will elaborate in future work.

## References

- Blom, G. (1982) 'On the mean number of random digits until a given sequence occurs', *Journal of Applied Probability*, March, Vol. 19, No. 1, pp.136–143.
- Blom, G. and Thorburn, D. (1982) 'How many random digits are required until given sequences are obtained?', *Journal of Applied Probability*, September, Vol. 19, No. 3, pp.518–531.
- Bouzoubaa, M. and Osseiran, A. (2010) *Exotic Options and Hybrids*, Wiley, NY.
- Chryssaphinou, O. and Papastavridis, S. (1988) 'A limit theorem for the number of non-overlapping occurrences of a pattern in a sequence of independent trials', *Journal of Applied Probability*, June, Vol. 25, No. 2, pp.428–431.
- Feller, W. (1968) *An Introduction to Probability and its Applications*, Vol. I, 3, Wiley, NY.
- Robin, S. and Daudin, J.J. (1999) 'Exact distribution of word occurrences in a random sequence of letters', *Journal of Applied Probability*, March, Vol. 36, No. 1, pp.179–193.

# Tuning Fermilab heavy quarks in 2 + 1 flavor lattice QCD with application to hyperfine splittings

C. Bernard,<sup>1</sup> C. DeTar,<sup>2</sup> M. Di Pierro,<sup>3</sup> A. X. El-Khadra,<sup>4</sup> R. T. Evans,<sup>4,5</sup> E. D. Freeland,<sup>1,\*</sup> E. Gámiz,<sup>4,6</sup> Steven Gottlieb,<sup>7,8</sup> U. M. Heller,<sup>9</sup> J. E. Hetrick,<sup>10</sup> A. S. Kronfeld,<sup>6</sup> J. Laiho,<sup>1,11</sup> L. Levkova,<sup>2</sup> P. B. Mackenzie,<sup>6</sup> J. N. Simone,<sup>6</sup> R. Sugar,<sup>12</sup> D. Toussaint,<sup>13</sup> and R. S. Van de Water<sup>14</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>Department of Physics, Washington University, St. Louis, Missouri 63130, USA

<sup>2</sup>Physics Department, University of Utah, Salt Lake City, Utah 84112, USA

<sup>3</sup>School of Computing, DePaul University, Chicago, Illinois 60604, USA

<sup>4</sup>Physics Department, University of Illinois, Urbana, Illinois 61801, USA

<sup>5</sup>Institut für Theoretische Physik, Universität Regensburg, 93040 Regensburg, Germany

<sup>6</sup>Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

<sup>7</sup>Department of Physics, Indiana University, Bloomington, Indiana 47405, USA

<sup>8</sup>National Center for Supercomputing Applications, University of Illinois, Urbana 61801, Illinois, USA

<sup>9</sup>American Physical Society, One Research Road, Ridge, New York 11961, USA

<sup>10</sup>Physics Department, University of the Pacific, Stockton, California 95211, USA

<sup>11</sup>Department of Physics and Astronomy, University of Glasgow, Glasgow, Scotland, UK

<sup>12</sup>Department of Physics, University of California, Santa Barbara, California 93106, USA

<sup>13</sup>Department of Physics, University of Arizona, Tucson, Arizona 85721, USA

<sup>14</sup>Physics Department, Brookhaven National Laboratory, Upton, New York 11973, USA

(Received 12 March 2010; published 23 February 2011)

We report the nonperturbative tuning of parameters— $\kappa_c$ ,  $\kappa_b$ , and  $\kappa_{\text{crit}}$ —that are related to the bare heavy-quark mass in the Fermilab action. This requires the computation of the masses of  $D_s^{(*)}$  and  $B_s^{(*)}$  mesons comprised of a Fermilab heavy quark and a staggered light quark. Additionally, we report the hyperfine splittings for  $D_s^{(*)}$  and  $B_s^{(*)}$  mesons as a cross-check of our simulation and analysis methods. We find a splitting of  $145 \pm 15$  MeV for the  $D_s$  system and  $40 \pm 9$  MeV for the  $B_s$  system. These are in good agreement with the Particle Data Group average values of  $143.9 \pm 0.4$  MeV and  $46.1 \pm 1.5$  MeV, respectively. The calculations are carried out with the MILC 2 + 1 flavor gauge configurations at three lattice spacings  $a \approx 0.15, 0.12$ , and  $0.09$  fm.

DOI: 10.1103/PhysRevD.83.034503

PACS numbers: 12.38.Gc, 12.39.Hg, 14.40.Lb, 14.40.Nd

## I. INTRODUCTION

Lattice QCD calculations play a critical role in the study of standard model physics and the search for new physics. For a set of lattice QCD calculations to be viable, several basic tasks are necessary. The bare gauge coupling must be eliminated in favor of an observable allowing the conversion from lattice to physical units; the bare masses in the lattice action must be tuned to correspond to physical quarks; and experimentally established quantities must be calculated in order to substantiate the method's accuracy and reliability. Once these tasks are complete, a variety of quantities inaccessible to or not yet determined by experiment may be calculated, such as decay constants, form factors, and mass spectra.

The Fermilab Lattice and MILC Collaborations have reported several calculations [1–8] based on ensembles of lattice gauge fields with 2 + 1 flavors of sea quarks, generated by the MILC Collaboration [9,10]. Details of the scale setting can be found in Refs. [11,12], and details of

the light-quark mass tuning in Ref. [11]. In this paper, we report on the necessary tuning of the heavy-quark action for charmed and bottom quarks. In particular, we describe calculations of the heavy-light pseudoscalar and vector meson masses using, for light quarks, the asqtad staggered action [13] and, for heavy quarks, the Fermilab interpretation [14] of the Sheikholeslami-Wohlert (“clover”) action [15] for Wilson fermions [16]. We use the spin average of these meson masses to nonperturbatively tune the hopping parameter  $\kappa$ , which is equivalent to the bare heavy-quark mass. We also describe the determination of  $\kappa_{\text{crit}}$ , the value of  $\kappa$  for which a degenerate Wilson pseudoscalar’s mass vanishes. The value of  $\kappa_{\text{crit}}$  plays a minor role in the calculation of heavy-light matrix elements [3–5], and a more important role when determining a renormalized-quark mass [6]. Finally, as a by-product of these calculations, we report the spin-dependent hyperfine splittings for  $B_s$  and  $D_s$  mesons, which test how well we have improved the chromomagnetic interaction.

Two aspects of the Fermilab method are important here. First, the Fermilab interpretation makes no assumptions about the size of the quark mass. Therefore, we are able to

\*eliz@fnal.gov

treat both charm and bottom quarks within the same framework. Second, since the Sheikholeslami-Wohlert action maintains the spin and flavor symmetries of heavy quarks, heavy-quark effective theory (HQET) can be used to interpret and improve lattice discretization effects [17,18]. HQET techniques can be used to show how the improvement works for observables, such as meson masses, in a way simpler than, though equivalent to, the Symanzik improvement program [19].

This paper is organized as follows. Section II reviews the theoretical framework upon which these calculations are based. Section III contains specific descriptions of the gauge configurations, actions, and operators used for the meson masses. Section IV covers the components of the numerical analysis. Section V details the fitting procedures. Section VI presents the results for the non-perturbative tuning of the heavy-quark hopping parameters  $\kappa_c$  and  $\kappa_b$ , the hyperfine splitting, and the critical hopping parameter  $\kappa_{\text{crit}}$ . Section VII summarizes with a discussion of improvements to these calculations that are currently underway. Details of the meson-mass discretization error estimation are given in Appendix A. Appendices B and C tabulate intermediate numerical results. The partially quenched chiral perturbation theory expression for the hyperfine splitting is derived in Appendix D.

## II. THEORETICAL BACKGROUND

The hopping-parameter form of the heavy-quark action is [14]

$$S = S_0 + S_B + S_E, \quad (2.1)$$

where

$$\begin{aligned} S_0 &= \sum_n \bar{\psi}_n \psi_n - \kappa \sum_{n,\mu} [\bar{\psi}_n (1 - \gamma_\mu) U_{n,\mu} \psi_{n+\hat{\mu}} \\ &\quad + \bar{\psi}_{n+\hat{\mu}} (1 + \gamma_\mu) U_{n,\mu}^\dagger \psi_n], \end{aligned} \quad (2.2)$$

$$S_B = \frac{i}{2} c_B \kappa \sum_{n;i,j,k} \epsilon_{ijk} \bar{\psi}_n \sigma_{ij} B_{n;k} \psi_n, \quad (2.3)$$

$$S_E = i c_E \kappa \sum_{n;i} \bar{\psi}_n \sigma_{0i} E_{n;i} \psi_n, \quad (2.4)$$

where  $\sigma_{\mu\nu} = \frac{i}{2} [\gamma_\mu, \gamma_\nu]$ . The chromomagnetic and chromoelectric fields  $B_{n;i}$  and  $E_{n;i}$  are standard and given in Ref. [14]. The term  $S_0$  includes dimension-five terms to alleviate the fermion doubling problem [16]. The couplings  $c_E$  and  $c_B$  of the dimension-five operators in  $S_B$  and  $S_E$  are chosen to reduce discretization effects [14,15].

The hopping parameter  $\kappa$  is related to the tadpole-improved bare-quark mass by

$$am_0 = \frac{1}{u_0} \left( \frac{1}{2\kappa} - \frac{1}{2\kappa_{\text{crit}}} \right), \quad (2.5)$$

where  $a$  is the lattice spacing,  $u_0$  is the tadpole-improvement factor [20], and  $\kappa_{\text{crit}}$  is the value of  $\kappa$  for which the pseudoscalar meson mass (of two degenerate Wilson quarks) vanishes. Our nonperturbative determination of  $\kappa_{\text{crit}}$  is discussed in Sec. VIC. To motivate our method of tuning  $\kappa$ , we first discuss the meson dispersion relation. We then turn to the HQET description of our Lagrangian to understand how to best use the dispersion relation.

The meson dispersion relation can be written, for  $|\mathbf{p}| \ll m_0, a^{-1}$ , as [14]

$$E(\mathbf{p}) = M_1 + \frac{\mathbf{p}^2}{2M_2} + O(\mathbf{p}^4). \quad (2.6)$$

Here, and throughout this work, we use lowercase  $m$  for quark masses and uppercase  $M$  for meson masses.  $M_1$  and  $M_2$  are known as the rest mass and kinetic mass, respectively. Because the lattice breaks Lorentz invariance,  $M_1 \neq M_2$ , although  $M_1 \rightarrow M_2$  as  $a \rightarrow 0$  for the action in Eq. (2.1). By tuning  $\kappa$ , one could adjust the bare, heavy-quark mass such that either  $M_1$  or  $M_2$  is equal to the physical meson mass. (To set  $M_1 = M_2$  requires the introduction, and tuning, of an additional parameter in the action. This is possible but, as discussed below, not necessary [14].)

To clarify the role of the different masses in Eq. (2.6), it is useful to introduce an effective Lagrangian. This also sets up a language for discussing discretization errors later. Because the action in Eq. (2.1) has the same heavy-quark spin and flavor symmetries as continuum QCD, HQET is an obvious candidate for its description [17,18]. To employ HQET, one separates the short-distance physics at the scale of the inverse heavy-quark mass  $1/m_Q$  from the long-distance physics at the characteristic scale of QCD,  $\Lambda_{\text{QCD}}$ . The fact that we have a lattice does not change the validity or utility of this separation. It simply means that the lattice spacing  $a$  must be included in the description of the short-distance physics. Thus, the short-distance coefficients of HQET applied to Eq. (2.1) differ from those arrived at by applying HQET to continuum QCD; these differences are the heavy-quark discretization errors. Parameters in the lattice action can be chosen to minimize them.

We introduce the heavy-quark effective Lagrangian for our lattice gauge theory by writing [17,18]

$$\mathcal{L}_{\text{LGT}} \doteq \mathcal{L}_{\text{light}} + \mathcal{L}_{\text{HQET}}, \quad (2.7)$$

where  $\mathcal{L}_{\text{light}}$  is the Symanzik local effective Lagrangian for the light degrees of freedom and  $\doteq$  means the Lagrangian on the right-hand side describes the on-shell matrix elements of the Lagrangian on the left-hand side. The HQET Lagrangian has a power-counting scheme, denoted by

$$\mathcal{L}_{\text{HQET}} = \sum_s \mathcal{L}_{\text{HQET}}^{(s)}, \quad (2.8)$$

where  $\mathcal{L}_{\text{HQET}}^{(s)}$  includes all operators of dimension  $4 + s$ , with coefficients of dimension  $-s$  consisting of powers of the short distances,  $1/m_Q$  or  $a$ . The first few terms in  $\mathcal{L}_{\text{HQET}}$  are [17]

$$\mathcal{L}_{\text{HQET}}^{(0)} = -\bar{h}^{(+)}(D_4 + m_1)h^{(+)}, \quad (2.9)$$

$$\mathcal{L}_{\text{HQET}}^{(1)} = \bar{h}^{(+)} \frac{\mathbf{D}^2}{2m_2} h^{(+)} + \bar{h}^{(+)} \frac{i\boldsymbol{\sigma} \cdot \mathbf{B}}{2m_B} h^{(+)}, \quad (2.10)$$

$$\mathcal{L}_{\text{HQET}}^{(2)} = \bar{h}^{(+)} \frac{i\boldsymbol{\sigma} \cdot (\mathbf{D} \times \mathbf{E})}{8m_E^2} h^{(+)} + \bar{h}^{(+)} \frac{\mathbf{D} \cdot \mathbf{E}}{8m_D^2} h^{(+)}, \quad (2.11)$$

where  $h^{(+)}$  is a two-component heavy-quark field,  $\boldsymbol{\sigma}$  are the Pauli matrices, and  $\mathbf{B}$  and  $\mathbf{E}$  are the continuum gauge fields. The masses  $m_1, m_2, m_B, m_E$ , and  $m_D$  are functions of the bare-quark mass  $m_0$  and the gauge coupling. For example, the masses  $m_1$  and  $m_2$  are defined to all orders in perturbation theory by Eq. (2.6), applied now to the pole energy of a one-quark state [21]. The entries in Eqs. (2.9), (2.10), and (2.11) are commonly referred to as follows.  $\mathcal{L}_{\text{HQET}}^{(0)}$  gives the rest mass. The first term of  $\mathcal{L}_{\text{HQET}}^{(1)}$  is the kinetic energy and the second is the chromomagnetic, or hyperfine, interaction. The first term of  $\mathcal{L}_{\text{HQET}}^{(2)}$  is the spin-orbit interaction while the second is known as the Darwin term.

For the pseudoscalar and vector meson rest masses, the HQET formalism can be used to show that [17]

$$M_1^{(*)} = m_1 + \bar{\Lambda}_{\text{lat}} - \frac{\lambda_{1,\text{lat}}}{2m_2} - d_J \frac{\lambda_{2,\text{lat}}}{2m_B} + O(1/m^2), \quad (2.12)$$

where  $J$  is the total meson angular momentum with  $d_0 = 3$  and  $d_1 = -1$  for the pseudoscalar ( $M_1$ ) and vector ( $M_1^*$ ) mesons, respectively. The quantities  $\bar{\Lambda}_{\text{lat}}$ ,  $\lambda_{1,\text{lat}}$ , and  $\lambda_{2,\text{lat}}$  are HQET matrix elements. At nonzero lattice spacing they contain discretization effects from  $\mathcal{L}_{\text{light}}$ , hence the subscript “lat.” The continuum limit of these quantities yields their counterparts in HQET applied to continuum QCD [17], which provides a basis for computing the continuum-QCD quantities  $\bar{\Lambda}$  and  $\lambda_1$  [22].

Mass splittings and matrix elements such as decay constants and form factors are not affected by the value of  $m_1$  [17]. Thus, Eqs. (2.9) and (2.10) show that the kinetic mass  $m_2$  is the first mass in the expansion that does play a role in the dynamics. We therefore would like to associate  $m_2$ , and hence  $M_2$ , with the physical mass, tolerating  $m_1 \neq m_2$  (and  $M_1 \neq M_2$ ) for nonzero lattice spacings. The nonperturbative tuning of  $\kappa$  then entails adjusting  $\kappa$  until the meson kinetic mass—determined by fits of Monte Carlo lattice data to the dispersion relation, Eq. (2.6)—equals that of the physical meson mass. A relation similar to Eq. (2.12) holds for  $M_2$

$$M_2^{(*)} = m_2 + \bar{\Lambda}_{\text{lat}} + O(1/m), \quad (2.13)$$

with the leading discretization errors appearing in the  $1/m$  contribution. Final values for the nonperturbative tuning of  $\kappa$  are given in Sec. VI A.

To calculate the hyperfine splitting of the  $D_s$  or  $B_s$  meson, consider

$$\Delta_1 \equiv M_1^* - M_1. \quad (2.14)$$

From Eq. (2.12),

$$M_1^* - M_1 = 4 \frac{\lambda_{2,\text{lat}}}{2m_B} + \dots, \quad (2.15)$$

which differs from the continuum splitting only by discretization errors in the light quarks and gluons appearing in  $\lambda_{2,\text{lat}}$ , the mismatch of  $m_B$  and its continuum counterpart (or, equivalently, the choice of  $c_B$ ), and similar contributions from higher-dimension operators [17,23]. The splitting of kinetic masses,  $\Delta_2 \equiv M_2^* - M_2$ , does not depend on  $m_B$ ; rather, it depends on other generalized masses which are not tuned in our simulations.<sup>1</sup> Thus,  $\Delta_1$  formally has smaller discretization errors than  $\Delta_2$ .  $\Delta_1$  is also statistically cleaner than  $\Delta_2$ . In Eq. (2.15),  $1/m_B$  is sensitive to the clover coupling  $c_B$  in Eq. (2.3), so  $\Delta_1$  tests how well it has been chosen. The  $B_s$  and  $D_s$  hyperfine splittings are given in Sec. VI B.

### III. SIMULATIONS

In this section, we describe the gauge configurations used and the details of the actions, operators, and correlation functions that describe the heavy-light mesons. In Sec. III A, we discuss the gauge configurations and the parameters that describe each ensemble. We also review how the lattice spacing is determined and the values of the conversion factors  $r_1$  and  $r_1/a$ . In Sec. III B, we discuss parameter choices for the valence quarks and the smearing of the heavy-quark wave function and how correlators are built from heavy and light-quark fields.

#### A. Gauge configurations and related parameters

We use the MILC gauge configurations [9,10] that have  $2 + 1$  flavors of asqtad-improved staggered sea quarks [13] and a Symanzik-improved gluon action [24,25]. Discretization errors from the sea quarks and gluons start at  $O(\alpha_s a^2, a^4)$ . The four-fold degeneracy of staggered sea quarks is removed by taking the fourth root of the determinant. To support the legitimacy of this procedure, Shamir has developed a renormalization-group framework for lattice QCD with staggered fermions, which he uses to argue that nonlocal effects of the rooted staggered theory are absent in the continuum limit [26]. Additional support

<sup>1</sup>Tree-level expressions for these masses, and hence their mismatch, can be found in Ref. [23].

TABLE I. Parameters describing the ensembles used. The dimensions of the lattice are given in terms of the spatial ( $N_L$ ) and temporal ( $N_T$ ) size in lattice units. The gauge coupling is given by  $\beta = 10/g^2$ . The bare masses of the light and strange sea quarks are given by  $am'_l$  and  $am'_s$ , respectively.  $L = aN_L$  is the linear spatial dimension of the lattice in fm. The column labeled  $N_{\text{cf}}$  is the number of configurations used in this work. The plaquette-determined tadpole-improvement factor is  $u_0$  [31]. The physical strange-quark mass is  $am_s$  [31] with errors, statistical and systematic, of less than 1%. The ratio  $r_1/a$  is described in the text; errors are Hessian from the smoothing fit. The final column lists the value of the inverse lattice spacing  $a^{-1}$  using  $r_1 = 0.3108^{(+30)}_{(-80)}$  fm to convert from  $r_1/a$ ; errors are from the error on  $r_1$  and  $r_1/a$ .

	$N_L^3 \times N_T$	$\beta$	$am'_l$	$am'_s$	$L$ (fm)	$N_{\text{cf}}$	$u_0$	$am_s$	$r_1/a$	$a^{-1}$ GeV
Fine $a \approx 0.09$ fm	$40^3 \times 96$	7.08	0.0031	0.031	3.5	435	0.8779	0.0252	3.692(6)	$2.344^{+60}_{-23}$
	$28^3 \times 96$	7.09	0.0062	0.031	2.4	557	0.8782	0.0252	3.701(5)	$2.349^{+61}_{-23}$
	$28^3 \times 96$	7.11	0.0124	0.031	2.4	518	0.8788	0.0252	3.721(5)	$2.362^{+61}_{-23}$
Coarse $a \approx 0.12$ fm	$24^3 \times 64$	6.76	0.005	0.05	2.9	529	0.8678	0.0344	2.645(3)	$1.679^{+43}_{-16}$
	$20^3 \times 64$	6.76	0.007	0.05	2.4	836	0.8678	0.0344	2.635(3)	$1.672^{+43}_{-16}$
	$20^3 \times 64$	6.76	0.010	0.05	2.4	592	0.8677	0.0344	2.619(3)	$1.663^{+43}_{-16}$
	$20^3 \times 64$	6.79	0.020	0.05	2.4	460	0.8688	0.0344	2.651(3)	$1.683^{+43}_{-16}$
Medium coarse $a \approx 0.15$ fm	$20^3 \times 64$	6.81	0.030	0.05	2.4	549	0.8696	0.0344	2.657(4)	$1.687^{+43}_{-16}$
	$16^3 \times 48$	6.572	0.0097	0.0484	2.4	631	0.8604	0.0426	2.140(4)	$1.358^{+35}_{-13}$
	$16^3 \times 48$	6.586	0.0194	0.0484	2.4	631	0.8609	0.0426	2.129(3)	$1.352^{+35}_{-13}$
	$16^3 \times 48$	6.600	0.0290	0.0484	2.4	440	0.8614	0.0426	2.126(3)	$1.350^{+35}_{-13}$

for this procedure comes from chiral perturbation theory arguments [27,28]. Reviews of these papers and of other evidence that this procedure reproduces the correct continuum limit appear in [11,29,30].

Table I lists the parameters of the gauge configurations used in this work. All configurations have been gauge-fixed to the Coulomb gauge. Ensembles of configurations are grouped by their approximate lattice spacing and are referred to as “fine” ( $a \approx 0.09$  fm), “coarse” ( $a \approx 0.12$  fm), and “medium-coarse” ( $a \approx 0.15$  fm). The simulation bare masses of the light and strange sea quarks are denoted by  $am'_l$  and  $am'_s$ , respectively, where  $am'_l$  is the mass of the two lighter sea quarks. The range of  $am'_l$  is light enough that the physical up- and down-quark masses can be reached by a chiral extrapolation, while  $am'_s$  is close to the physical strange-quark mass. For convenience below, we write  $(am'_l, am'_s)$  to identify ensembles, e.g., “the (0.0031, 0.031) fine ensemble.” Also in Table I are the tadpole factors  $u_0$  [20,31], determined from the mean plaquette and used to improve the gauge-configuration actions [9,10]. The value of the physical strange-quark mass is denoted by the unprimed  $m_s$  [31].

To convert between lattice and physical units, the physical value of the lattice spacing must be determined. We define the distance  $r_1$  [12] by

$$r_1^2 F(r_1) = 1, \quad (3.1)$$

where  $F(r)$  is the force between static quarks, calculated on the lattice. For each ensemble, this yields a value of  $r_1$  in lattice units,  $r_1/a$ . The values are then “smoothed” by fitting  $\ln(r_1/a)$ , from all ensembles, to a polynomial in  $\beta$  and  $2am'_l + am'_s$  [31]. The physical value of  $r_1$  is obtained via the lattice calculation of an experimentally measurable

quantity. We consider two current determinations here. One uses a lattice calculation of the Y(2S)-Y(1S) splitting [33] to arrive at  $r_1 = 0.318(7)$  fm [10,34]. A more recent determination using  $r_1 f_\pi$  gives  $r_1 = 0.3108(15)^{(+26)}_{(-79)}$  fm [35]. These two determinations are consistent within errors. Because the determination of  $r_1$  from  $f_\pi$  uses finer lattice spacings, we take that value,

$$r_1 = 0.3108^{(+30)}_{(-80)} \text{ fm} \quad (3.2)$$

with no additional error. While this work was being completed, a new determination of  $r_1$  that uses two mass splittings and one decay constant became available;  $r_1 = 0.3133^{(+23)}_{(-3)}$  [36], which is consistent with the value used in this work. Quantities can now be converted from lattice to physical units by using  $r_1$  and the appropriate value of  $r_1/a$  [31] given in Table I.

## B. Meson correlation functions

Table II lists the values of parameters used in the valence-quark actions. For the light-valence quark, we again use the asqtad action [13] and masses  $am'_q$  close to the physical value of the strange-quark mass, cf. Table I. From Eqs. (2.9), (2.10), and (2.11), one can see that with  $m_2$  tuned to the physical mass, the leading mismatch between lattice and continuum physics is in the hyperfine term in  $\mathcal{L}_{\text{HQET}}^{(1)}$ . In principle, one can tune  $m_B$  to its continuum counterpart yielding a match between lattice and continuum actions for both terms in Eq. (2.10). Here, we use the tree-level expression for  $m_B$ , which leaves the leading mismatch at  $O(\alpha_s a \Lambda)$ . By setting  $c_E = c_B$  we obtain the Sheikholeslami-Wohlert,  $O(a)$  improvement of the

TABLE II. Parameters used in the valence-quark actions. The bare masses of the light and strange sea quarks ( $am'_l$ ,  $am'_s$ ) label the ensemble. The mass of the light (staggered) valence quark is given by  $am'_q$ .  $c_E$  and  $c_B$  are the coefficients of the chromoelectric and chromomagnetic contributions to the Lagrangian. With  $c_E = c_B$ , they are the usual Sheikholeslami-Wohlert coupling.  $u_0$  is the tadpole-improvement factor from measurements of the average plaquette for the fine and medium-coarse ensembles and from the Landau-gauge link on the coarse ensembles. Hopping parameter values  $\kappa$  used for bottomlike and charmlike heavy quarks are given in the final two columns.

Lattice	$(am'_l, am'_s)$	$am'_q$	$c_E = c_B$	$u_0$	Bottom-type $\kappa$	Charm-type $\kappa$
Fine	(0.0031, 0.031)	0.0272, 0.031	1.478	0.8779	0.0923	0.127
	(0.0062, 0.031)	0.0272, 0.031	1.476	0.8782	0.090, 0.0923, 0.093	0.1256, 0.127
	(0.0124, 0.031)	0.0272, 0.031	1.473	0.8788	0.0923	0.127
Coarse	(0.005, 0.050)	0.030, 0.0415	1.72	0.836	0.086	0.122
	(0.007, 0.050)	0.030, 0.0415	1.72	0.836	0.074, 0.086, 0.093	0.119, 0.122, 0.124
	(0.010, 0.050)	0.030, 0.0415	1.72	0.8346	0.074, 0.086, 0.093	0.119, 0.122, 0.124
	(0.020, 0.050)	0.030, 0.0415	1.72	0.8369	0.074, 0.086, 0.093	0.122, 0.124
	(0.030, 0.050)	0.030, 0.0415	1.72	0.8378	0.086	0.122
Medium-coarse	(0.0097, 0.0484)	0.0387, 0.0484	1.570	0.8604	0.070, 0.080	0.115, 0.122 <sup>a</sup> , 0.125
	(0.0194, 0.0484)	0.0387, 0.0484	1.567	0.8609	0.070, 0.076, 0.080	0.115, 0.122, 0.125
	(0.0290, 0.0484)	0.0484	1.565	0.8614	0.070, 0.080	0.115, 0.125

<sup>a</sup>Used only with  $am'_q = 0.484$ .

discretization errors in the action [15]. From the HQET perspective, this leaves  $m_E \neq m_2$  in Eq. (2.11), but the effects of this mistuning are at  $O(a^2\Lambda^2)$  and  $O(\alpha_s a \Lambda^2 / m_Q)$ . Implementing the improvements above and using tree-level tadpole improvement in the perturbative expressions [20,24], we use  $c_E = c_B = u_0^{-3}$ .

The values of  $u_0$  used in the heavy-quark and light-valence actions are given in Table II. For the fine and medium-coarse ensembles, they are the plaquette values used to generate the MILC gauge configurations. For the coarse ensembles, the Landau-gauge link value was used. The use of different  $u_0$  definitions results in a slight mismatch between the light valence- and sea-quark actions. In part because the meson mass is relatively insensitive to the strange sea-quark mass, we do not expect any significant systematic errors from this mismatch. Changes in  $u_0$  result in changes to the bare mass of the heavy quark as well, but this effect is partly absorbed by the nonperturbative tuning of  $\kappa$  and  $\kappa_{\text{crit}}$ . Table II also lists the nominal values of the light valence-quark mass and sets of  $\kappa$  values for bottom and charm mesons. These sets of  $\kappa$  values, and mesons created from them, are referred to as charm-type or bottom-type.

With the parameters of the actions set, we now turn to the construction of the two-point correlators. Contributions from excited states can be significantly reduced by using a spatially smeared source, sink, or both, for the heavy-quark propagator. For the correlators in this work, we use two types of source-sink combinations for the heavy quarks. One is simply a delta function for both the source and sink; we refer to this as the local correlator. The other smears the field  $\psi(t, \mathbf{x})$  with a discretized version [37] of the 1S charmonium wave function,  $S(y)$ , based on the Richardson potential [38]:

$$\phi(t, \mathbf{x}) = \sum_y S(y) \psi(t, \mathbf{x} + y), \quad (3.3)$$

and the smearing wave function is applied after fixing to the Coulomb gauge. Correlators using  $\phi(t, \mathbf{x})$  are referred to as smeared correlators. All light valence quarks have a local source and sink. The meson correlator is

$$C_{i,j}(t, \mathbf{p}) = \sum_x \langle \mathcal{O}_j^\dagger(t, \mathbf{x}) \mathcal{O}_i(0, \mathbf{0}) \rangle e^{i\mathbf{p} \cdot \mathbf{x}}, \quad (3.4)$$

where  $i, j$  denote the source, sink smearing of the heavy-quark field; for this work  $i = j$ .  $\mathcal{O}_i(t, \mathbf{x})$  is a bilinear interpolating operator with a gamma-matrix structure that yields quantum numbers appropriate for either pseudoscalar or vector mesons. To construct this operator, we combine a one-component, staggered light-quark spinor with a four-component, Wilson-type heavy-quark spinor in a manner similar to Ref. [39],

$$\mathcal{O}_\Xi(t, \mathbf{x}) = \bar{\psi}_\alpha(t, \mathbf{x}) \Gamma_{\alpha\beta} \Omega_\beta \Xi(t, \mathbf{x}) \chi(t, \mathbf{x}), \quad (3.5)$$

where  $\Gamma = \gamma_5$  or  $\gamma_\mu$ ;  $\alpha, \beta$  are spin indices; and  $\Omega(x) \equiv \gamma_1^{x_1} \gamma_2^{x_2} \gamma_3^{x_3} \gamma_4^{x_4}$ . The fields  $\bar{\psi}$  and  $\chi$  are the Wilson-type and staggered fields, respectively, and the smeared correlator is constructed in the same way, but with  $\bar{\phi}$  instead of  $\bar{\psi}$ . The transformation properties of  $\mathcal{O}_\Xi(x)$  under shifts by one lattice spacing are such that  $\Xi$  can be viewed as playing the role of the (fermionic) taste index [30,40]. In our correlation functions,  $\mathcal{O}_\Xi(x)$  is summed over  $2^4$  hypercubes, and so  $\Xi$  can be interpreted as a taste degree of freedom in the sense of Refs. [41,42].

#### IV. ANALYSIS OVERVIEW

In this section, we describe the components of our analysis. Section IV A discusses the two-point correlator fits used to determine the meson energies  $aE(\mathbf{p})$ . Section IV B describes how we fit the meson dispersion relation to obtain  $M_2$ . Finally, Sec. IV C explains how  $\kappa$  is tuned and how the hyperfine splitting is determined.

##### A. Two-point correlator fits: $E(\mathbf{p})$

To determine  $E(\mathbf{p})$ , we simultaneously fit the local and smeared heavy-light-meson two-point correlators to the function

$$C_{i,i}(t, \mathbf{p}) = \sum_{\eta=0}^{N-1} \left[ Z_{i,\eta}^2 (e^{-E_\eta(\mathbf{p})t} + e^{-E_\eta(\mathbf{p})(N_T-t)}) + (-1)^{t+1} (Z_{i,\eta}^p)^2 (e^{-E_\eta^p(\mathbf{p})t} + e^{-E_\eta^p(\mathbf{p})(N_T-t)}) \right], \quad (4.1)$$

where  $N_T$  is the temporal extent of the lattice, and terms proportional to  $e^{-E_\eta(\mathbf{p})(N_T-t)}$  are due to periodic boundary conditions. To simplify notation in this subsection, the lattice spacing  $a$  is not written out explicitly. Correlation functions containing staggered light quarks have contributions from both desired- and opposite-parity states with the opposite-parity states having the temporally oscillating prefactor  $(-1)^{t+1}$  [39]. We take each energy level  $\eta$  in Eq. (4.1) to include a pair of states consisting of one desired- and one opposite-parity state; the number of pairs of states in a fit is given by  $N$ . Quantities associated with the tower of opposite-parity states are denoted by the superscript “p.”

Equation (4.1) contains  $2N$  exponentials, and the number of time slices in our data set is finite. Although it is straightforward to separate the two different parities—because of the  $(-1)^{t+1}$ —it is difficult to separate states within each tower. Rather than relying solely on taking  $t$  large enough, we use the technique of constrained curve fitting [39,43,44]. We thus minimize an augmented  $\chi^2$  [43],

$$\chi_{\text{aug}}^2 \equiv \chi^2 + \sum_k \frac{(P_k - \tilde{P}_k)^2}{\sigma_{\tilde{P}_k}^2}, \quad (4.2)$$

which means each fit parameter  $P_k$  is provided a prior Gaussian probability distribution function with central value and width  $(\tilde{P}_k, \sigma_{\tilde{P}_k})$ . The central value for fitted quantities comes from minimizing  $\chi_{\text{aug}}^2$  on the whole ensemble. We take the parameters to be  $E_0^{(p)}$ ,  $\ln(Z_{i,\eta}^{(p)})$ , and (for  $\eta > 0$ )  $\ln(\Delta E_\eta^{(p)})$ , where  $\Delta E_\eta^{(p)} = E_\eta^{(p)} - E_{\eta-1}^{(p)}$ , thereby enforcing a tower of states with increasing energy.

In general, one considers a quantity to be determined by the data only if the statistical error, discussed next, is smaller than the corresponding prior width. In this work,

we are most concerned with the lowest-lying desired parity state, and the data—not the priors—always determine  $E_0$  and  $Z_{i,0}$ . For parameters that are poorly constrained by the data, such as those describing excited states, these priors prevent the fitter from searching fruitlessly along flat directions in parameter space. Because of the freedom in choosing the prior, we test whether the ground-state results are prior independent, and stable. When testing the stability of fit results, we use the Hessian error, defined as

$$\sigma_{P_i} = \sqrt{2 \left( \frac{\partial^2 \chi_{\text{aug}}^2}{\partial P_i \partial P_j} \right)_{ii}}, \quad (4.3)$$

because its straightforward definition allows it to be quickly calculated for a single fit.

When using  $\chi_{\text{aug}}^2$  to measure the goodness of fit, we count the degrees of freedom as the number of data points; the number of fit parameters is not subtracted since there are an equal number of extra terms in  $\chi_{\text{aug}}^2$ . In some cases, this could result in misleadingly low values of  $\chi_{\text{aug}}^2/\text{dof}$ . For example, if the prior width  $\sigma_{\tilde{P}_k}$  is much larger than  $(P_k - \tilde{P}_k)$ , the associated term in  $\chi_{\text{aug}}^2$  will be much smaller than the others. This could be adjusted *a posteriori* by reducing the degrees of freedom, but it would require devising a criterion for “large  $\sigma_{\tilde{P}_k}$ .” We do not make such adjustments in our analyses. Instead, to determine goodness of fit, we monitor the values of  $\chi_{\text{aug}}^2/\text{dof}$  from constrained fits, but rely equally on the stability of fit results.

We estimate statistical uncertainties by generating pseudo-ensembles via the bootstrap method. When fitting a pseudo-ensemble, the central value of each prior is drawn randomly from its Gaussian probability distribution while the prior width is kept the same [39,43]. To prevent large, simultaneous but uncorrelated fluctuations among prior central values, which could destabilize a fit, we restrict the randomized prior central values to  $\pm 1.5\sigma_{\tilde{P}}$ . Final errors quoted for meson energies and functions thereof, such as the spin-averaged mass, are obtained from their bootstrap distributions. We define the upper (lower) 68% distribution point as the value at which 16% of the distribution has a higher (lower) value. We refer to half of the distance between these two points as the average 68% bootstrap error.

##### B. Dispersion relation fits: The kinetic mass

Having determined  $E(\mathbf{p})$ , we use the dispersion relation to determine the kinetic meson mass, which we then use to tune the hopping parameter  $\kappa$ . The low-momentum expansion for  $E(\mathbf{p})$  is

$$E(\mathbf{p}) = M_1 + \frac{\mathbf{p}^2}{2M_2} - \frac{a^3 W_4}{6} \sum_i p_i^4 - \frac{(\mathbf{p}^2)^2}{8M_4^3} + \dots, \quad (4.4)$$

where  $W_4$  and the deviation of  $M_4$  from  $M_2$  capture lattice artifacts. (In the continuum limit  $a^3 W_4 = 0$  and  $M_4 = M_2$ .) The vector  $\mathbf{n}$  is defined by

$$a\mathbf{p} = (2\pi/N_L)\mathbf{n}, \quad (4.5)$$

where  $N_L$  is the spatial extent of the lattice, given in Table I; data are generated for  $|\mathbf{n}| \leq 3$ . Noise in  $E(\mathbf{p})$  increases with increasing momentum, though, and is substantial by the time  $O(\mathbf{p}^4)$  effects become significant. For charm-type mesons, squaring the energy yields a substantial cancellation in the  $O(\mathbf{p}^4)$  contribution because  $aM_1 \approx aM_2 \approx aM_4$ . While this is not true for bottom-type mesons, the mass of these mesons is large enough to cause suppression via the  $1/M$  factors whether  $E(\mathbf{p})$  or  $E^2(\mathbf{p})$  is used. By fitting to  $E^2(\mathbf{p})$  then, the contributions from  $O(\mathbf{p}^4)$  effects are reduced, and we are able to do a linear fit to low-momentum data,  $|\mathbf{n}| \leq 2$ . Setting  $M_1 = E(\mathbf{0})$  from the zero-momentum correlator, we square Eq. (4.4) and fit

$$E^2(\mathbf{p}) - M_1^2 = C\mathbf{p}^2 \quad (4.6)$$

to obtain  $C$ . Finally, we set  $M_2 = M_1/C$ . The largest  $\mathbf{p}$  is chosen so that the  $O(\mathbf{p}^4)$  effects are expected to be negligible, based on tree-level values of the analogous quark quantities  $w_4$  and  $1/m_q^3$ . We confirm the negligibility of these terms by inspecting plots of the data and monitoring  $\chi^2/\text{dof}$ . (We do not use constrained curve fitting here and so we minimize the usual  $\chi^2$ .) This procedure is repeated for each bootstrap-generated pseudo-ensemble, yielding bootstrap distributions for  $aM_1$  and  $aM_2$ .

### C. The hopping parameter $\kappa$ and the hyperfine splitting $\Delta_1$

For tuning  $\kappa$ , it is helpful to remove the leading discretization errors from spin-dependent terms. Let the spin-averaged kinetic meson mass be

$$\bar{M}_2 = \frac{1}{4}(M_2 + 3M_2^*), \quad (4.7)$$

where  $M_2$  and  $M_2^*$  are determined as described in Sec. IV B. This leaves the second, spin-independent term in Eq. (2.11) as the leading source of the discretization error, the contribution is  $O(a^2 \Lambda^2)$ . Our goal then is to determine the value of  $\kappa$  that will result in a value of  $\bar{M}_2$  that agrees with the experimental value taken from the Particle Data Group (PDG).

For each lattice spacing, we use the following procedure to tune  $\kappa$ . Using three or more ensembles, we study the light sea-quark mass dependence of  $a\bar{M}_2$  for at least one combination of  $\kappa$  and  $m_q'$ . This gives us some insight into the behavior of  $a\bar{M}_2$  in the physical sea-quark-mass limit and allows us to assign an uncertainty to  $a\bar{M}_2$  due to nonphysical sea-quark masses. Next, on at least one ensemble, we determine  $a\bar{M}_2$  at two staggered, valence-quark masses near the strange-quark mass. This allows us to determine the dependence of  $a\bar{M}_2$  on the staggered, valence-quark mass and interpolate linearly to the physical

value if no simulated mass is close enough to the tuned strange-quark mass. Having dealt with the staggered-valence and light sea-quark masses, we take  $a\bar{M}_2$  at the physical, strange valence-quark mass at two values of  $\kappa$  and interpolate linearly in  $\kappa$  to the spin-averaged value of the meson masses, given by the PDG [45], converted to lattice units with  $a$  from Table I. Finally, we combine the uncertainties in the tuned value of  $\kappa$  from statistical and discretization errors in the meson mass, staggered-valence mass mistuning, nonphysical sea-quark masses, and errors from the lattice-spacing conversion of the PDG mass.

To determine the hyperfine splitting, we start with the results for  $M_1 = E(\mathbf{0})$ . For each lattice spacing, we use values of  $a\Delta_1$  at, or linearly interpolated to, the tuned charm and bottom  $\kappa$  values. We then consider uncertainties from statistics, the tuning of  $\kappa$  and  $am_s$ , nonphysical sea-quark masses, and discretization. The value of  $a\Delta_1$  on the fine lattice is taken as our central value and results on the coarse and medium-coarse lattices are used in the error analysis. In the final value, we also include an uncertainty due to the conversion to physical units.

## V. FITTING DETAILS FOR $E(\mathbf{p})$ , $M_1$ , $M_2$

In this section, we describe the details of our fitting procedure for the meson energy  $E(\mathbf{p})$  and the meson rest and kinetic masses,  $M_1$  and  $M_2$ . Our objective here is to document thoroughly our fitting procedures, including values for the priors, and tests. Readers who are more interested in a summary can skip to Sec. V C.

Section VA discusses the parameters used in our two-point correlator fits for  $E(\mathbf{p})$  (Sec. VA 1) and the evaluation of goodness of fit via  $\chi^2_{\text{aug}}/\text{dof}$  and tests of stability (Sec. VA 2). In most tests discussed here, Hessian errors were used, because they are fast and straightforward. Our complete data set, exhibited in Table II, contains several ensembles at each of the three lattice spacings. As explained in Sec. VA 1, one ensemble at each lattice spacing is chosen for the purpose of setting priors in Eq. (4.2). For tuning  $\kappa$ , we need data over a range of  $\kappa$  and  $am_q'$  on a fixed ensemble. At the fine lattice spacing, such data were generated on only one ensemble, (0.0062, 0.031), so we set priors and tune  $\kappa$  on that same ensemble. For the coarse and medium-coarse lattice spacings, we have data for a range of  $\kappa$  and  $am_q'$  on several ensembles. We take the coarse (0.010, 0.050), and medium-coarse (0.0194, 0.0484) ensembles to set priors and then the ensembles with the smallest  $am_q'$  (and a range of  $\kappa$  and  $am_q'$ ) to tune  $\kappa$ . We compute the hyperfine splittings from the same ensembles on which  $\kappa$  was tuned. These choices are summarized in Table III. Data from other ensembles listed in Table II are used to estimate uncertainties.

Fits of the dispersion relation to determine  $M_2$  from  $E(\mathbf{p})$  are comparatively simple, and Sec. VB provides details that may be of interest.

TABLE III. Specific ensembles used in steps of the analyses. Setting priors is discussed in Sec. V A 1. Stability and goodness-of-fit tests done for  $E(\mathbf{p})$  results are described in Sec. V A 2.  $\kappa$ -tuning and hyperfine-splitting results are given in Secs. V A and V B, respectively.

Lattice	Setting priors	$E(\mathbf{p})$ tests, tuning $\kappa$ , and the hyperfine splitting $\Delta_1$
Fine	(0.0062, 0.031)	(0.0062, 0.031)
Coarse	(0.010, 0.050)	(0.007, 0.050)
Medium-coarse	(0.0194, 0.0484)	(0.0097, 0.0484)

### A. Two-point fits: $E(\mathbf{p}), M_1$

The number of gauge configurations in each ensemble is given in Table I. To improve statistics, we generate data at four time sources on each of the fine and coarse gauge configurations and at eight time sources for medium-coarse configurations. We also average the correlator points  $C(t)$  and  $C(N_T - t)$ . In order to reduce the effect of correlations between data points from sequential configurations, we bin the data by groups of  $N_{\text{bin}}$  configurations. Because fits for this project were done in concert with other projects,  $N_{\text{bin}} = 4$  was adopted. Comparisons of results using  $N_{\text{bin}} = 2, 4$ , and 6 on the ensembles used here show no significant change in the fit-result error bars or the bootstrap distributions. To account for correlations in the two-point correlator data, the fitter uses the normalized, data-sample covariance matrix as an estimate of the correlation matrix. This matrix is remade for each bootstrap sample.

#### 1. Priors, time ranges, $N$

We consider the setting of priors for the ground-state parameters, excited-state amplitudes, and energy splittings separately. Ground-state ( $\eta = 0$ ) parameters are well determined by the data; thus, the ground-state priors can, and

should, be negligibly constraining. In contrast, energy splittings and excited-state amplitudes are not well determined by the data, and the related priors are chosen such that they put reasonable bounds on the parameters. The next paragraphs describe how the priors are set. Note that the same set of priors is used for all ensembles at a given lattice spacing, for all momenta in the range  $|\mathbf{n}| = 0$  to 2, and for all  $\kappa$  and  $am_q'$  of a given meson type, e.g., charm pseudoscalars. The priors used are tabulated in Tables IV, V, and VI.

We use information from a subset of our data, one ensemble per lattice spacing, to set the priors for the two-point-correlator fits. This is necessary because we do not have enough external knowledge to set them independently. The ensembles used to help set the priors are listed in Table III. Other ensembles are statistically independent of these ensembles and so the prior information can be viewed as external to fits on those ensembles. If possible, though, we do not want to exclude any data from our analysis, including the ensembles used in the setting of priors. For this reason, our procedure for setting priors keeps the amount of information we take from these ensembles to a minimum. Specifically, for a parameter  $P$ , we use averages over ranges of parameters, like the momentum, for the prior central value  $\tilde{P}$  and chose prior widths  $\sigma_{\tilde{P}}$  that are broad enough to cover the expected results for an entire subset of fits; e.g., the same priors are used for fits with  $|\mathbf{n}| = 0$  to 2.

To set ground-state priors, we first fit to large-time data with  $N = 1$  in order to get a general idea of the ground-state parameter values. We then set  $N > 1$  and fit correlators at low and high momenta to ascertain the range of values the ground-state parameters may take. We set prior central values for the ground-state energy of the desired-and opposite-parity states,  $aE_0(\mathbf{p})$  and  $aE_0^P(\mathbf{p})$ , at about the midpoint of the range seen in these fits.

TABLE IV. Priors used for fine-ensemble two-point correlator fits for pseudoscalar and vector mesons. Priors for all higher amplitudes and splittings are the same as those for the first excited state. The fit-parameter numbers 15–20 label the second excited state and so on. A prior of  $\ln(a\Delta E) = -1.45^{+1.0}_{-1.0}$  on the fine ensembles corresponds approximately to  $\Delta E = 550^{+950}_{-350}$  MeV.

Fit parameter	Fit-parameter number	Charm mesons		Bottom mesons	
		Pseudoscalar	Vector	Pseudoscalar	Vector
$E_0$	1	0.90(40)	0.90(40)	1.75(60)	1.75(60)
$E_0^P$	2	1.0(40)	0.95(40)	1.85(60)	1.85(60)
$\ln(Z_{1S,0})$	3	1.0(2.0)	1.0(2.0)	1.0(3.0)	1.0(3.0)
$\ln(Z_{1S,0}^P)$	4	1.0(2.0)	1.0(2.0)	1.0(3.0)	1.0(3.0)
$\ln(Z_{d,0})$	5	-2.0(2.0)	-2.0(2.0)	-2.0(3.0)	-2.0(3.0)
$\ln(Z_{d,0}^P)$	6	-2.0(2.0)	-2.0(2.0)	-2.0(3.0)	-2.0(3.0)
$\ln(\Delta E)$	8	-1.45(1.0)	-1.45(1.0)	-1.45(1.0)	-1.45(1.0)
$\ln(\Delta E^P)$	9	-1.45(1.0)	-1.45(1.0)	-1.45(1.0)	-1.45(1.0)
$\ln(Z_{1S,1})$	10	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{1S,1}^P)$	11	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{d,1})$	12	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{d,1}^P)$	13	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)

TABLE V. Same as Table IV, but for the coarse ensembles. A prior of  $\ln(a\Delta E) = -1.2^{+0.5}_{-0.5}$  on the coarse ensembles corresponds approximately to  $\Delta E = 500^{+300}_{-200}$  MeV.

Fit parameter	Fit-parameter number	Charm mesons		Bottom mesons	
		Pseudoscalar	Vector	Pseudoscalar	Vector
$E_0$	1	1.10(40)	1.2(40)	2.00(40)	2.00(40)
$E_0^p$	2	1.30(40)	1.3(40)	1.10(40)	1.10(40)
$\ln(Z_{1S,0})$	3	1.0(2.0)	1.0(2.0)	1.0(2.0)	1.0(2.0)
$\ln(Z_{1S,0}^p)$	4	1.0(3.0)	0.1(3.0)	-1.0(2.0)	-0.1(2.0)
$\ln(Z_{d,0})$	5	-1.0(2.0)	-1.0(2.0)	-2.0(2.0)	-1.0(2.0)
$\ln(Z_{d,0}^p)$	6	-1.0(3.0)	-2.0(3.0)	-2.0(2.0)	-2.0(2.0)
$\ln(\Delta E)$	8	-1.2(0.5)	-1.2(0.5)	-1.2(0.5)	-1.2(0.5)
$\ln(\Delta E^p)$	9	-1.2(0.5)	-1.2(0.5)	-1.2(0.5)	-1.2(0.5)
$\ln(Z_{1S,1})$	10	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{1S,1}^p)$	11	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{d,1})$	12	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{d,1}^p)$	13	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)

TABLE VI. Same as Table IV, but for the medium-coarse ensembles. A prior of  $\ln(a\Delta E) = -1.0^{+0.5}_{-0.5}$  on the medium-coarse ensembles corresponds approximately to  $\Delta E = 500^{+300}_{-200}$  MeV.

Fit parameter	Fit-parameter number	Charm mesons		Bottom mesons	
		Pseudoscalar	Vector	Pseudoscalar	Vector
$E_0$	1	1.38(50)	1.46(50)	2.35(40)	2.38(50)
$E_0^p$	2	1.50(60)	1.58(60)	2.48(50)	2.50(50)
$\ln(Z_{1S,0})$	3	0.48(1.0)	0.95(1.0)	0.12(1.4)	0.60(1.0)
$\ln(Z_{1S,0}^p)$	4	-0.65(1.0)	0.20(1.0)	-1.0(2.0)	0.1(2.0)
$\ln(Z_{d,0})$	5	-0.90(1.0)	-0.74(1.0)	-1.15(1.0)	-0.8(1.0)
$\ln(Z_{d,0}^p)$	6	-2.4(1.4)	-1.8(2.0)	-2.5(3.0)	-1.8(3.0)
$\ln(\Delta E)$	8	-1.0(0.5)	-1.0(0.5)	-1.0(0.5)	-1.0(0.5)
$\ln(\Delta E^p)$	9	-1.0(0.5)	-1.0(0.5)	-1.0(0.5)	-1.0(0.5)
$\ln(Z_{1S,1})$	10	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{1S,1}^p)$	11	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1
$\ln(Z_{d,1})$	12	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)
$\ln(Z_{d,1}^p)$	13	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)	-1.0(3.0)

To understand our logic for setting the prior widths for  $aE_0(\mathbf{p})$  and  $aE_0^p(\mathbf{p})$ , recall that we use a Gaussian distribution for the prior  $\tilde{P}$  with a width  $\sigma_{\tilde{P}}$ . We set  $\sigma_{a\tilde{E}_0}$  and  $\sigma_{a\tilde{E}_0^p}$  large enough so that results across the entire momentum range used in the analysis should fall well within the  $1 - \sigma_{a\tilde{E}_0}$ , or  $1 - \sigma_{a\tilde{E}_0^p}$ , range of the distribution. After priors for the remaining parameters are set, we perform a complete set of fits and, for at least one ensemble at each lattice spacing, verify that, indeed, the final fit results for  $aE_0$  and  $aE_0^p$  fit well within their respective prior distributions.

Priors for the ground-state amplitudes are loosely based on the preliminary  $N > 1$  fits described above. In most cases, the central value is the nearest whole number to the

average of these results. For the desired-parity state, the widths  $\sigma_{\tilde{P}}$  are chosen such that they easily span the range of values seen in the fits. For the opposite-parity states, which are substantially noisier, the widths span the distance between the prior central value and the observed range in the results by about  $1 - \sigma_{\tilde{P}}$ .

Priors for all excited-state amplitudes were set to have a relatively small central value and a wide width. To set the prior for the energy splitting, we note that experimentally measured meson splittings are a few hundred MeV. We also bear in mind that the sum of a series of exponentials with a very small energy splitting is not a well-posed problem. Therefore, we chose the central value of the splitting to be several hundred MeV, slightly large, with a generous prior width. For example, on the fine lattice the prior for the

TABLE VII. Time range  $t_{\min}$ – $t_{\max}$  and number of (pairs of) states  $N$  used in two-point correlator fits at each lattice spacing. For the time range, the first (second) number in parenthesis is  $t_{\min}$  for the 1S-smeared (local) correlator;  $t_{\max}$  is the same for both correlators.

Lattice spacing	Time range	$N$
Fine	(2, 4)–25	3
Coarse	(2, 8)–15	2
Medium-coarse	(5, 6)–15	2

splitting,  $\ln(a\Delta E) = -1.45(1.0)$  is equivalent to  $\Delta E \approx 550^{+950}_{-350}$  MeV.

In the charm sector, the opposite-parity partner of the  $D_s(0^-)$ , the  $D_{s0}^*(0^+)$ , is close to the  $DK$  threshold. In this case, the energy splitting should not be viewed as a meson-mass splitting, and our choice of prior for the  $D_{s0}^*(0^+)$  energy splitting may be inappropriate. The parity-partner signal is noisy, though, and in tests of the priors widths we see no change in the nonoscillating ground-state energy  $aE(\mathbf{p})$ , which is our main interest. For details, see Sec. V A 2.

To choose the time ranges for the fits,  $(t_{\min}, t_{\max})$ , we first look at the data to determine the time by which the error in the data, e.g., the relative error in the correlator, has increased substantially. This gives us a potential value for  $t_{\max}$ . From effective mass plots we can also see at what time slice the majority of the excited-state contamination has died off, giving us a potential value for  $t_{\min}$ . Constrained curve fitting is designed to reduce excited-state contamination of the lower-state fit parameters. Nevertheless, we do not see a significant reduction in the error from fitting to the smallest possible time slice, which requires including a larger number of states in the fit. For simplicity, we chose final time ranges that are the same for similar sets of data. These can be found in Table VII.

With the time range set, we do fits for increasing values of the number of (pairs of) states  $N$  and look for the ground-state energy to stabilize. We choose the final values of  $N$  to be the minimum value needed to be in the stable region; these are given in Table VII. Figure 1 shows representative plots of  $aE(\mathbf{p})$  versus  $N$  from fits on the (0.0062, 0.031) fine ensemble. It is clear that for the minimum-value  $N$ , the central value of the fit result is always well within the stable region. In some cases, though, the (Hessian) error from the minimum- $N$  fit is smaller than that in the stable region. One could remedy this by choosing to fit with more states. Unfortunately, an increase in the number of states leads to non-Gaussian bootstrap distributions with a significant number of outliers—clearly nonphysical fit results that contain ground states with low energies and very small amplitudes. Using the minimum possible number of states, no outliers have been seen in the distributions.

TABLE VIII. Data used in stability and goodness-of-fit tests.

Lattice	Ensemble	$\kappa$	$am'_q$
Fine	(0.0062, 0.031)	0.127; 0.090 or 0.093	0.0272
Coarse	(0.007, 0.050)	0.122; 0.086	0.0415
Medium-coarse	(0.0097, 0.0484)	0.125; 0.070	0.0484

## 2. Tests of stability and goodness of fit

Having set the priors, time range, and number of states for the fits, we check the stability of the results and goodness of fit in several ways. For result stability, we check the effects of the time range used, the number of (pairs of) states  $N$ , and changes to the prior widths; we also compare the priors to the fit results. We look at a representative subset of fits for each lattice spacing: pseudoscalar and vector meson correlators at two different  $\kappa$  values (one for charm and one for bottom) for a given light-valence mass, on one ensemble per lattice spacing, and with momenta  $\mathbf{n} = (0, 0, 0)$  and  $(1, 1, 1)$  or  $(2, 0, 0)$ . The specific values of  $\kappa$ ,  $am'_q$ , and  $(am'_b, am'_s)$  vary from test to test, and in some cases tests are extended to other values. A description of the data used in the tests discussed here can be found in Table VIII.

For the time-range tests, we vary  $t_{\min}$  over two to four time slices, increasing  $N$  if appropriate, and vary  $t_{\max}$  over five to ten time slices. We verify that there are no changes in the fit results beyond expected fluctuations.<sup>2</sup> For number-of-states tests, we verify that the result is stable as  $N$  is increased. Figure 1 shows example results for the (0.0062, 0.031) fine ensemble. Similar results are seen for the coarse and medium-coarse ensembles and for the ground-state amplitudes  $Z_{1S}$  and  $Z_d$ .

For prior-width tests, we reduce the widths by a factor of 2 for the nonoscillating ground-state quantities and the energy splittings and repeat the fits. All changes observed are within statistical errors and, in most cases, the changes are substantially smaller than 1  $\sigma$ . For charm, we also test for effects of the  $DK$  threshold near the  $D_{s0}^*(0^+)$  state. This splitting is 50 to 100 MeV, which is a several- $\sigma_{\tilde{\Delta}_{aEP}}$  deviation from our prior central value. We ran separate tests on each lattice spacing using a prior width of  $\sigma_{\tilde{\Delta}_{aEP}} = 2.5$  for the oscillating-state energy splitting. In units of MeV, this puts a 50-MeV splitting within  $1\sigma_{\tilde{\Delta}_{aEP}}$  of the prior central value. The ground and first-excited-state energies of the oscillating state are affected by this change but not in a systematic way. This indicates that the oscillating-state

<sup>2</sup>In one case,  $\kappa = 0.086$ , coarse (0.010, 0.005), although the ground-state energy is stable as  $t_{\max}$  is varied, the value of  $\chi^2/\text{dof}$  becomes large as  $t_{\max}$  is increased beyond the final value ( $t_{\max} = 15$ ). This ensemble is not used directly for  $\kappa$  tuning or hyperfine-splitting determinations as explained in the introduction to this section.

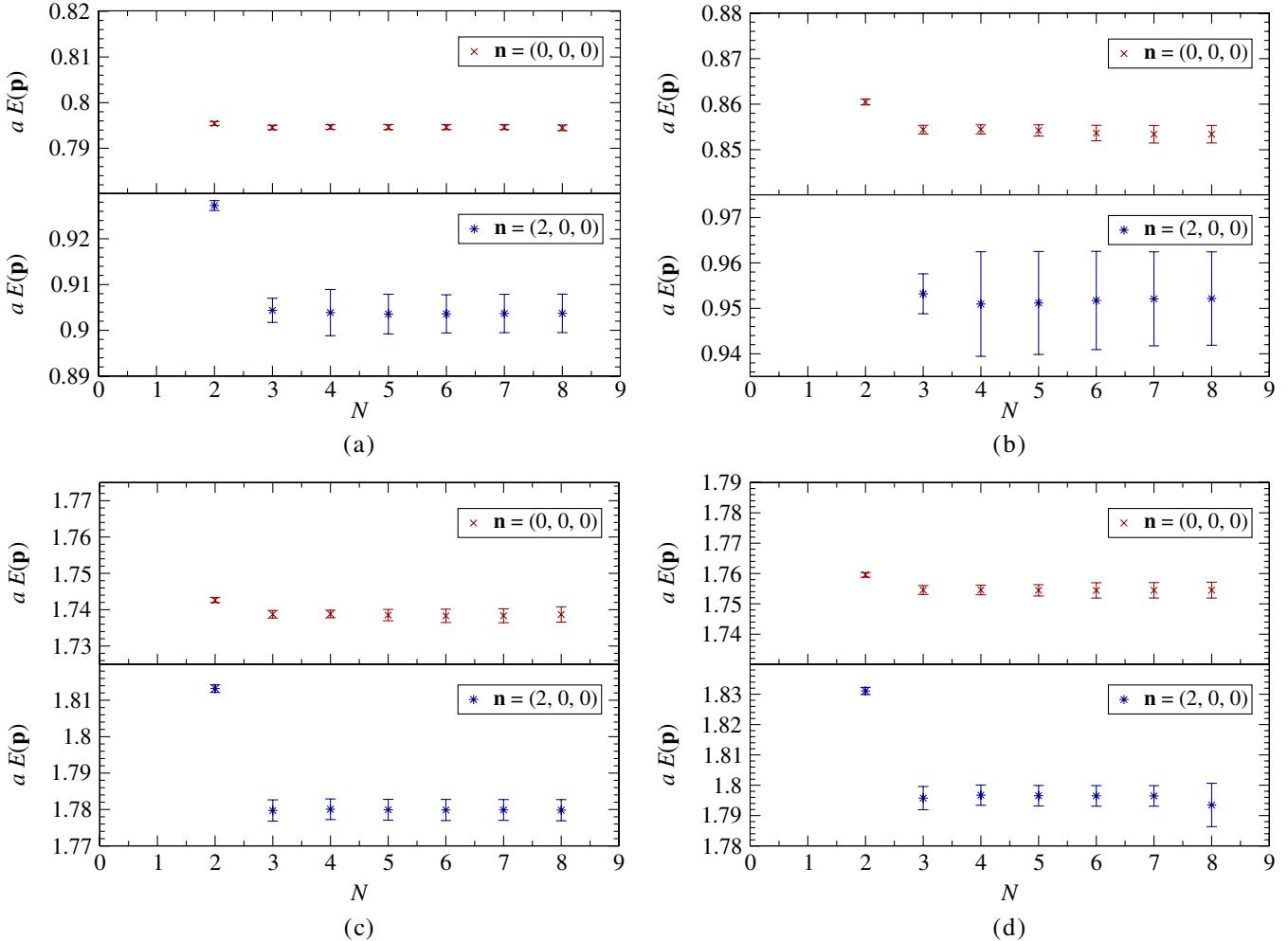


FIG. 1 (color online). Fitted values of  $aE(\mathbf{p})$  vs the number of (pairs of) states  $N$  for  $\kappa = 0.127$ , charm-type (a) pseudoscalar and (b) vector mesons and  $\kappa = 0.090$ , bottom-type (c) pseudoscalar and (d) vector mesons on the (0.0062, 0.031) fine ensemble. Results shown are for mesons with momenta  $\mathbf{n} = (0, 0, 0)$  and  $(2, 0, 0)$ . Errors are Hessian.

signal is not strong in our data. Our main interest, though, is the nonoscillating ground-state energy  $aE(\mathbf{p})$ ; this value is unaffected by the change in  $\sigma_{\tilde{\Delta}_{aE}}$ .

In addition, we compare fit results with their priors. Figure 2 gives examples of these comparisons for fits on the (0.0062, 0.031) fine ensemble for charm- and bottom-type mesons. The  $x$  axis labels the fit-parameter number, defined in Table IV; the ground-state energy and amplitudes of the desired-parity state are at positions 1, 3, and 5. We find that fit results for ground-state quantities are well within the prior widths. For excited states, in some cases the fitter simply returns the prior value, indicating that the quantity is not constrained by the data. In other cases, the results appear to be constrained by the data, indicating that some excited-state signal is in the correlator and the fitter adjusts the amplitudes to absorb it. Although it may appear in Fig. 2 that a number of excited-state quantities are well determined, this is an artifact of a minimum- $N$  fit; unlike the ground-state parameters, the excited-state results are

not stable as  $N$  is increased. For example, Fig. 3 compares the fit results shown in the upper left (pseudoscalar) panel of Fig. 2(a), which uses  $N = 3$ , with a fit which only differs by the use of  $N = 4$ . The comparison demonstrates that the (desired-parity) ground-state quantities are stable to the change in  $N$  while other, excited-state, parameters are not.

For goodness of fit we begin by looking at the augmented  $\chi^2/\text{dof}$  for each fit and verify that it is  $\approx 1$  or smaller, where “ $\approx 1$ ” is based on the 80% range of the  $\chi^2/\text{dof}$  distribution for a given number of degrees of freedom. As a final check, we overlay the result on an effective-mass plot. We define the “effective energy”

$$2aE_{\text{eff}}(\mathbf{p}) = \ln[C(t)/C(t + 2a)] \quad (5.1)$$

using a step of two time units in order to accommodate the oscillating contribution from the opposite-parity state. Figure 4 shows plots comparing  $aE_{\text{eff}}(\mathbf{p})$  to the fit result on the (0.0062, 0.031) fine ensemble. The ground-state-energy result from the multiple-state fit is shown as a

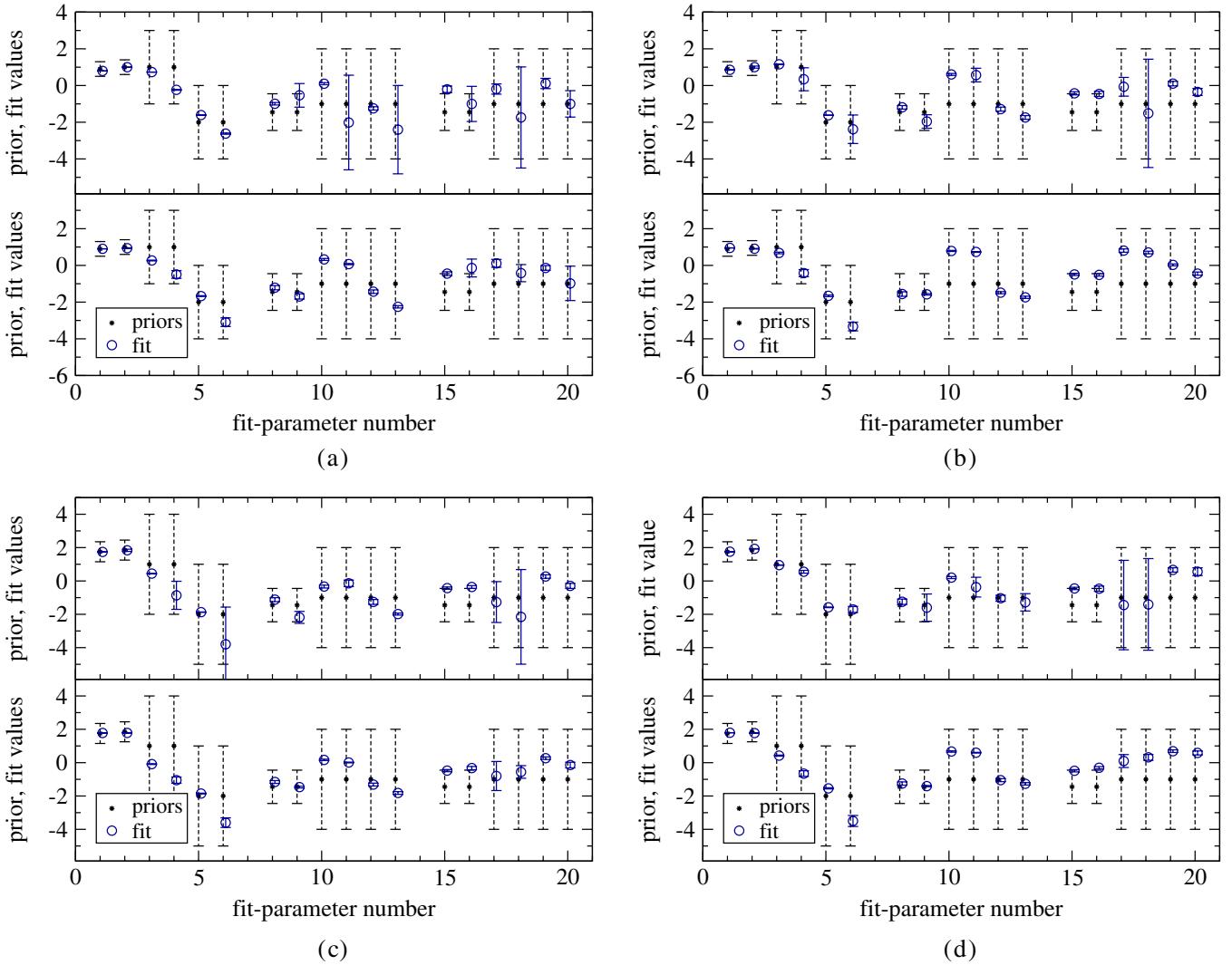


FIG. 2 (color online). Fit results shown as open (blue) circles are overlaid on the priors, black dots with dashed widths, for charm-type (a) pseudoscalar and (b) vector mesons and bottom-type (c) pseudoscalar and (d) vector mesons on the  $(0.0062, 0.031)$  fine ensemble.  $\kappa = 0.127$  and  $0.090$  for charm- and bottom-type mesons, respectively;  $am_q' = 0.0272$ . The upper [lower] plot is from a fit where the meson has momentum of  $\mathbf{n} = (0, 0, 0)$  [ $(2, 0, 0)$ ]. The fit-parameter numbers are defined in Table IV. In each panel, the leftmost cluster corresponds to quantities from the ground state; the middle cluster corresponds to the first excited state; and the right most cluster to the second excited state. Errors on the fit results are Hessian. For clarity, fit results are offset along the  $x$  axis.

straight line segment over the time range fit. The band encompasses the average 68% bootstrap error. In each case, the fit result nicely matches the effective-energy plateau.

### B. The kinetic mass $M_2$

Given results for  $aE(\mathbf{p})$ , we fit data where  $|\mathbf{n}| \leq \sqrt{3}$  to Eq. (4.6) to determine the pseudoscalar and vector kinetic meson masses. Fits use a correlation matrix constructed from the bootstrap distributions. The tables in Appendix B give results for  $aM_2$ ,  $aM_2^*$ , and  $a\bar{M}_2$  on the ensembles used for tuning, listed in Table III. Included in the tables are the  $\chi^2/\text{dof}$  and the probability that  $\chi^2$  would exceed the value from the fit, known as the  $p$  value [45]. Typical dispersion

relation fits are shown for the  $(0.0062, 0.031)$  fine ensemble in Fig. 5.

In addition to statistical errors, we consider uncertainties from unphysical sea-quark masses, mistuning of the valence strange quark, and discretization. The noise in  $\bar{M}_2$  makes it difficult to discern how  $\bar{M}_2$  depends on the sea-quark masses. The  $\bar{M}_1$  data is much cleaner, though, and we can use it to estimate the sea-quark error on  $\bar{M}_2$ , and hence  $\kappa$ . To do this, we first note that, cf. Eq. (2.12),

$$aM_1 = am_1 + a\bar{\Lambda}_{\text{lat}} + O(1/m_Q), \quad (5.2)$$

$$aM_2 = am_2 + a\bar{\Lambda}_{\text{lat}} + O(1/m_Q), \quad (5.3)$$

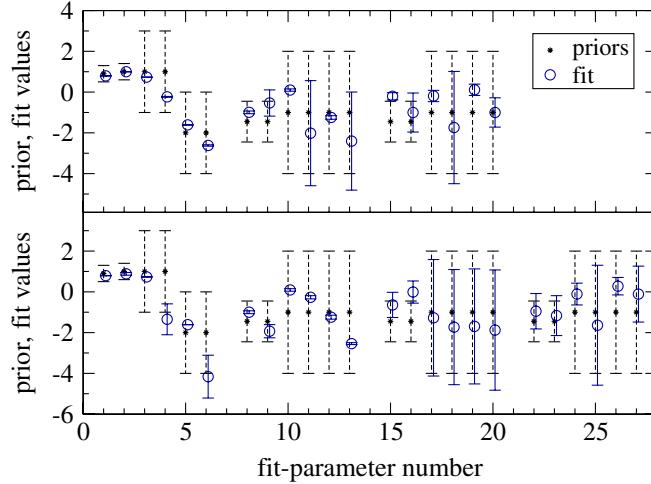


FIG. 3 (color online). Fit results shown as open (blue) circles are overlaid on the priors, black dots with dashed widths, for charm-type mesons on the (0.0062, 0.031) fine ensemble.  $\kappa = 0.127$ ;  $am'_q = 0.0272$ ;  $\mathbf{n} = (0, 0, 0)$ . The upper plot is the same as the upper left (pseudoscalar) panel of Fig. 2(a). The lower plot is from a fit which only differs by the use of  $N = 4$  pairs of states. The fit-parameter numbers are defined in Table IV. In each panel, the leftmost cluster corresponds to quantities from the ground state; the middle cluster corresponds to the first excited state; the next cluster corresponds to the second excited state and so on. The (desired-parity) ground-state quantities are stable to this change while other, excited-state, parameters are not. Errors on the fit results are Hessian. For clarity, fit results are offset along the  $x$  axis.

where  $am_1$  and  $am_2$  capture the leading heavy-quark dependence and  $a\bar{\Lambda}_{\text{lat}}$  depends only on the light degrees of freedom. Taking  $a\bar{\Lambda}_{\text{lat}}$  to be the same for both  $a\bar{M}_1$  and  $a\bar{M}_2$  (see Appendix A and Ref. [46]) we can estimate the size of the effect of nonphysical (light) sea-quark masses on  $a\bar{\Lambda}_{\text{lat}}$ , and hence  $aM_2$ , by studying the behavior of  $aM_1$  as the light sea-quark masses are varied.

In Fig. 6, we plot the spin-averaged meson rest mass  $r_1\bar{M}_1$  versus the ratio of the light to strange sea-quark masses  $m'_l/m'_s$  for the coarse and fine ensembles used here. On the far right of each plot is a bar indicating the size of the  $1 - \sigma$  statistical error on  $r_1\bar{M}_2$ ; for fine this is from the (0.0062, 0.031) ensemble and for coarse the (0.007, 0.050) ensemble. The light sea-quark mass dependence is negligible compared to the statistical error on  $r_1\bar{M}_2$ . We find similar behavior for the medium-coarse ensemble.

We must also consider how the nonphysical value of the strange sea-quark mass affects  $\bar{M}_2$ . The strange sea-quark mass is mistuned by an amount  $0.19am'_s$ ,  $0.31am'_s$  and  $0.12am'_s$  on the fine, coarse, and medium-coarse ensembles, respectively. The continuum chiral perturbation theory expression for the heavy-light spin-averaged mass [47] shows that the leading sea-quark dependence of  $\bar{M}_2$  is proportional to the sum over the sea-quark masses,  $2m'_l + m'_s$ . Hence, varying  $am'_l$  tells us about the effect of varying  $am'_s$ . Figure 6 shows that a change of  $0.3am'_s$  in  $am'_l$  has a negligible effect on  $\bar{M}_2$ , so we conclude that the mistuning of  $am'_s$  has a negligible effect as well.

The tuned value of the strange-quark mass on each ensemble is given in Table I. On the fine lattice, the

valence-quark mass used in the simulation,  $am'_q = 0.0272$ , differs from the physical value  $am_s = 0.0252$  by 0.0020. A comparison of our results for  $a\bar{M}_2$  in Table XVIII shows that even a deviation in  $am'_q$  of twice this size does not discernibly affect  $a\bar{M}_2$ . The situation is similar for the coarse and medium-coarse results. For the coarse ensembles, the simulation mass  $am'_q = 0.03$  differs by 0.0044 from the tuned value of  $am_s$ . Table XIX shows that  $a\bar{M}_2$  is barely affected at the  $1 - \sigma_{a\bar{M}_2}$  level as  $am'_q$  changes by over twice this size. For the medium-coarse ensembles, the simulation mass of 0.0484 differs from the tuned strange-quark mass by 0.0058. A comparison of the values of  $a\bar{M}_2$  in Table XX shows that a deviation in  $am'_q$  just under twice this size yields, at most, a  $1 - \sigma_{a\bar{M}_2}$  variation in  $a\bar{M}_2$ . Therefore, we take our results of  $a\bar{M}_2$  at  $am'_q = 0.0272$ , 0.03, and 0.0484 as the masses of the  $B_s$  and  $D_s$  on the fine, coarse and medium-coarse ensembles, respectively, with no additional error for valence-mass mistuning.

In Appendix A, we derive an expression for the discretization error in  $M_2$ ,  $M_2 = M_{\text{continuum}} + \delta M_2$ . The result, Eq. (A22), can be written as

$$\delta M_2 = \frac{\bar{\Lambda}^2}{6m_2} \left[ 5 \left( \frac{m_2^3}{m_4^3} - 1 \right) + 4w_4(m_2a)^3 \right], \quad (5.4)$$

replacing  $\langle p^2 \rangle$  of Eq. (A22) with  $\bar{\Lambda}^2$ . Expressions for the short-distance coefficients  $m_2$ ,  $m_4$ , and  $w_4$  are given in Appendix A [14,23]. To estimate the discretization error,

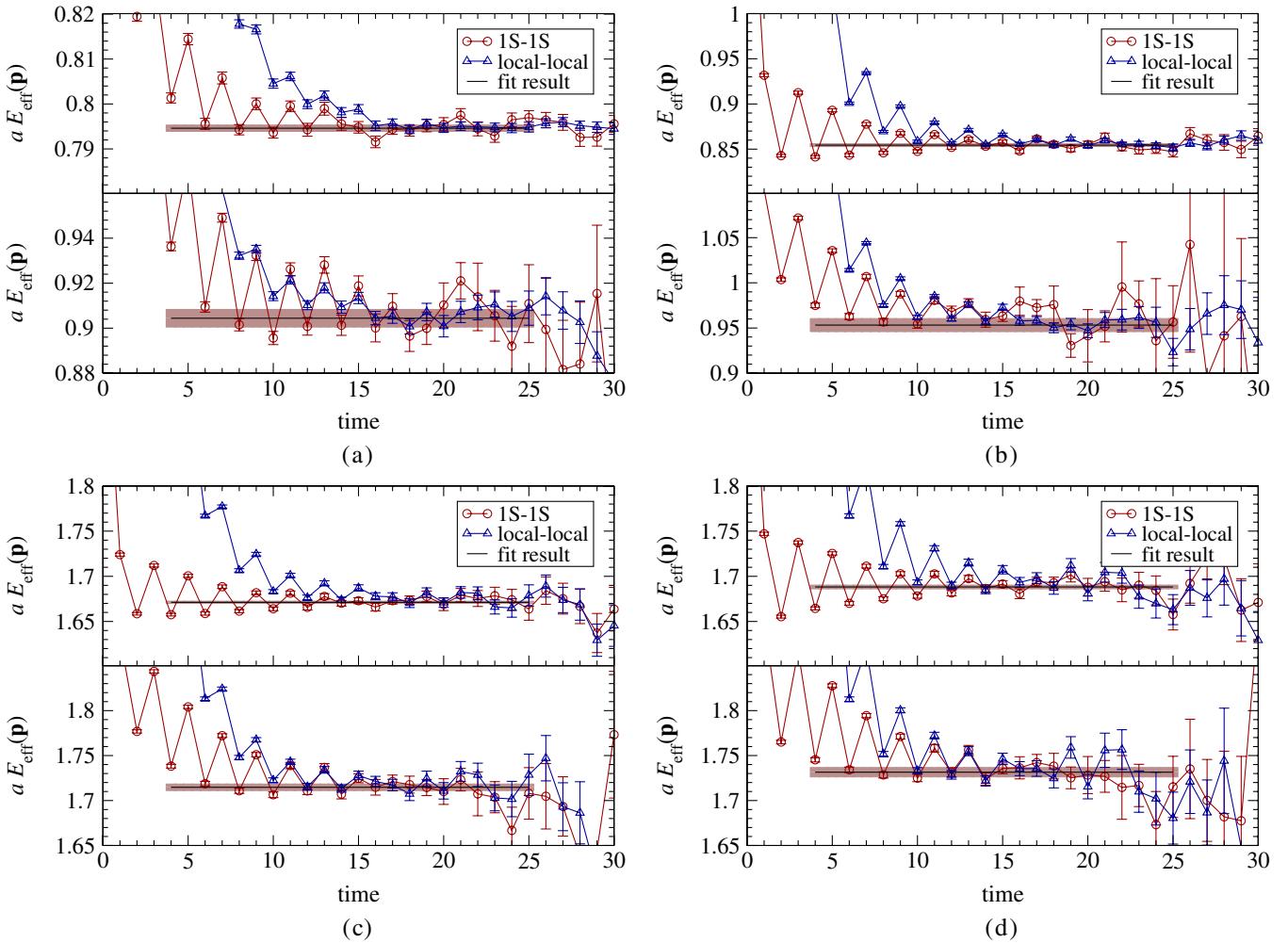


FIG. 4 (color online). Effective energy plots,  $aE_{\text{eff}}(\mathbf{p})$ , for charm-type (a) pseudoscalar and (b) vector mesons and bottom-type (c) pseudoscalar and (d) vector mesons on the  $(0.0062, 0.031)$  fine ensemble.  $\kappa = 0.127$  and  $0.093$  for charm- and bottom-type mesons, respectively;  $am'_q = 0.0272$ . The upper [lower] plot is from a fit where the meson has momentum of  $\mathbf{n} = (0, 0, 0)$  [ $\mathbf{n} = (2, 0, 0)$ ]. Open (blue) triangles mark the local correlator and open (red) circles mark the 1S-smeared correlator. Lines connecting the data points are simply to guide the eye; they are not a fit. The unadorned black line is the multicorrelator fit result and the shaded band marks the average 68% bootstrap error.

we use values of the physical (pole) quark mass (1.4 GeV for charm and 4.2 GeV for bottom) for  $m_2$  in the prefactor of Eq. (5.4), and  $\bar{\Lambda} = 0.7$  GeV. Using these values,  $u_0$  from Table II, and  $\kappa_{\text{crit}}$  from Table XVII yields the values of  $\delta M_2$  shown in Table IX. The error estimate in Eq. (5.4) pertains to the kinetic mass, but the main focus here is the tuning of  $\kappa$ . After tuning, we shall propagate this error from  $M_2$  to  $\kappa_c$  and  $\kappa_b$ .

### C. Fitting summary

The preceding subsections contain many details intended for those engaged in similar analyses. In this section, we re-emphasize the main features of the analysis. Because, in this and related [3–8] work, we are interested in the ground state, we do not dwell on the excited states here.

Our priors are guided by the data, using one ensemble to set them and (generally) other ensembles for physical results. We choose a time range such that the fit results for the ground state are stable, listed in Table VII. We also test for stability as the number  $N$  of (pairs of) exponentials grows—as shown in one example in Fig. 1—and choose the minimum value of  $N$  for which the central value is stable within errors. The errors on the ground-state amplitudes and energies are always determined by the data, not the priors, as shown in Fig. 2 and 3. (In many cases, even excited-state information is data-determined, not prior determined.) Figure 4 shows that the fits agree with the effective energies. (Note that the oscillations of  $aE_{\text{eff}}$  at small  $t$  are to be expected with staggered quarks.) In conclusion, the constrained curve fitting for  $E(\mathbf{p})$  has worked as advertised, subsuming the subjectivity of fit ranges and different choices of  $N$  into robust results for

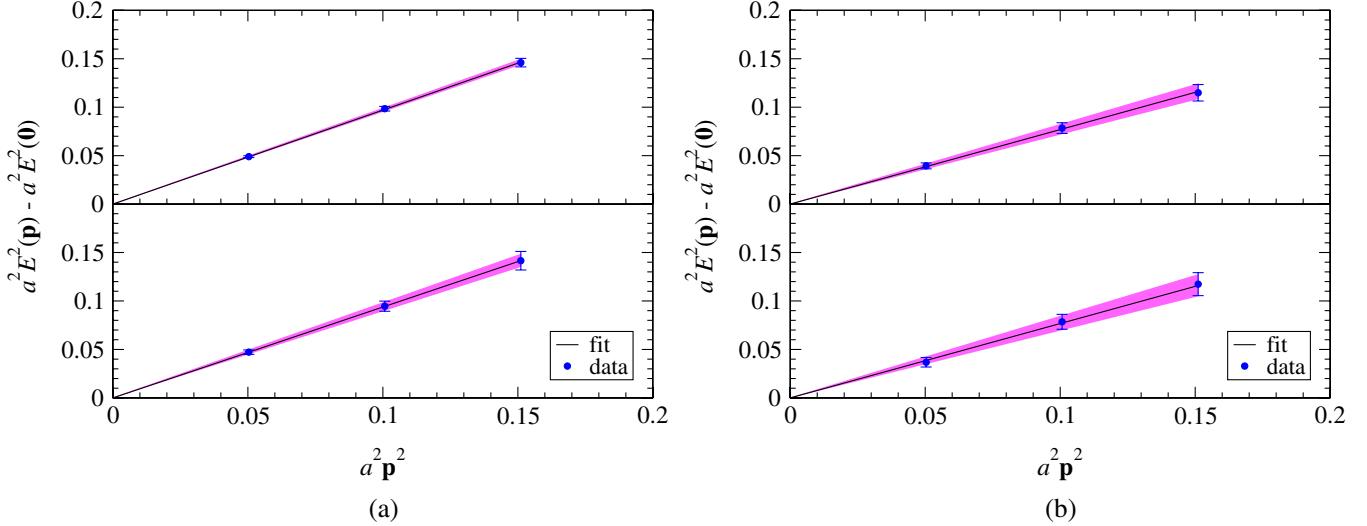


FIG. 5 (color online). Results of fits to the dispersion relation for (a) charm-type ( $\kappa = 0.127$ ) and (b) bottom-type ( $\kappa = 0.0923$ ) mesons on the (0.0062, 0.031) fine ensemble. (Blue) dots are the data. A black line shows the fit result with the (pink) shaded band showing the one-sigma error from the fit. Upper panels show results for pseudoscalars and lower for vectors.

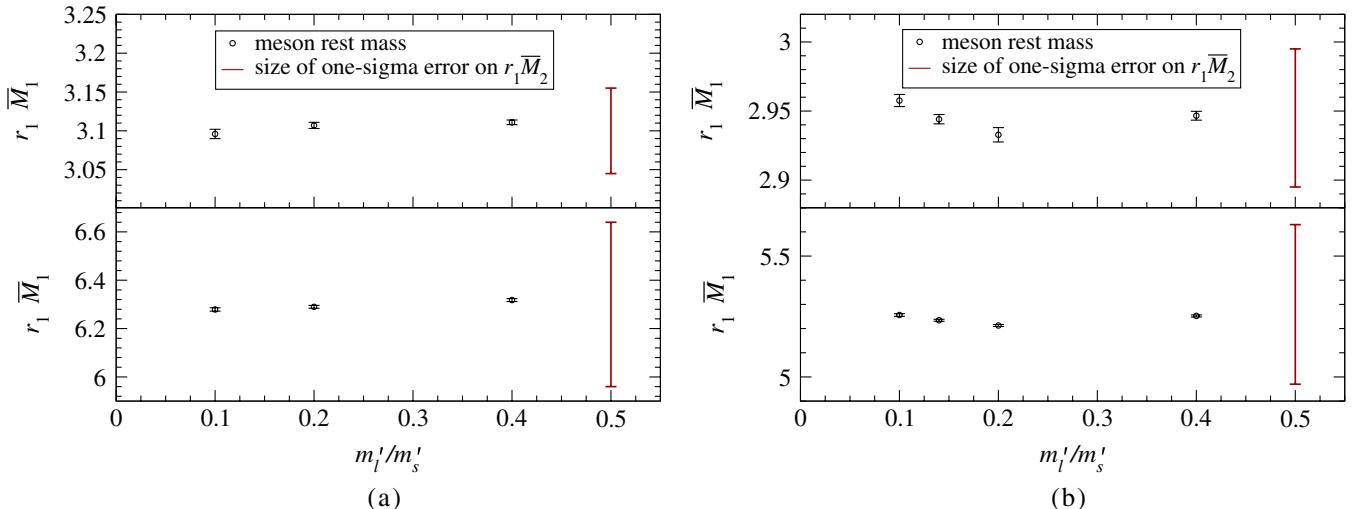


FIG. 6 (color online). The spin-averaged meson rest mass in physical units versus the ratio of the light to strange sea-quark masses  $m'_l/m'_s$  for the (a) fine and (b) coarse ensembles. Error bars are statistical only, from the average 68% bootstrap error. On the far right of the plot is a (red) bar indicating the size of the one-sigma statistical error on  $r_1 \bar{M}_2$ . The upper-panel plot is for charm-type mesons, lower is for bottom-type. Values of  $\kappa$  used are 0.127, 0.0923 on the fine ensembles with  $am'_q = 0.0272$  and  $\kappa$  0.122, 0.086 on the coarse ensembles with  $am'_q = 0.0415$ .

both the central value and error bar. Figures 5 and 6 show that, once  $E(\mathbf{p})$  is well determined, we can straightforwardly obtain the kinetic mass  $M_2$  and the hyperfine splitting.

## VI. RESULTS

In this section, we present the main results of these calculations, including our error analysis. Section VI A focuses on the tuned values of  $\kappa_c$  and  $\kappa_b$ , Sec. VI B on

the  $D_s$  and  $B_s$  hyperfine splittings, and Sec. VI C on the critical value of the hopping parameter  $\kappa_{\text{crit}}$ .

### A. The tuning of $\kappa_c$ and $\kappa_b$

As discussed in Sec. VI B, effects from nonphysical sea-quark masses and the mistuning of the valence strange-quark mass are negligible compared to the statistical error on  $a\bar{M}_2$ . In that section, we explain why taking  $a\bar{M}_2$  at certain values of  $am'_q$  is an acceptable approximation to

TABLE IX. The relative error in the tuned hopping parameter  $\delta\kappa/\kappa$  due to discretization effects in the kinetic meson mass. The ensembles used are (0.0062, 0.031), (0.007, 0.050), and (0.0097, 0.0484) for the fine, coarse, and medium-coarse lattices, respectively. Values of  $\kappa$  are 0.127 and 0.0923 on fine; 0.122 and 0.086 on coarse; and, 0.122 and 0.076 on medium-coarse. The  $[\cdot \cdot \cdot]$  denotes the quantity in brackets in Eq. (5.4). We use  $(\bar{\Lambda}^2/6m_{\text{ch}}) = 0.058\bar{\Lambda}$  and  $(\bar{\Lambda}^2/6m_{\text{bot}}) = 0.019\bar{\Lambda}$  to convert the  $[\cdot \cdot \cdot]$  to  $\delta M_2$ . Values of  $\delta\kappa/\kappa$  are given as fractions not a percentage.

Lattice spacing	$m_0a$	$[\cdot \cdot \cdot]$	Charm			$m_0a$	$[\cdot \cdot \cdot]$	Bottom		
			$\delta M_2$	$\frac{dm_2a}{dm_0a}$	$\frac{\delta\kappa}{\kappa}$			$\delta M_2$	$\frac{dm_2a}{dm_0a}$	$\frac{\delta\kappa}{\kappa}$
Fine	0.391	1.31	0.0763	0.843	-0.0086	2.08	16.8	0.327	0.880	-0.0256
Coarse	0.565	2.37	0.1384	0.831	-0.0203	2.62	23.6	0.459	0.899	-0.0440
Medium-coarse	0.682	3.18	0.1857	0.830	-0.0346	3.56	37.2	0.724	0.922	-0.0756

TABLE X. PDG values of the pseudoscalar and vector masses for the  $D_s$  and  $B_s$  mesons and the hyperfine splitting  $\Delta$  [45]. Also listed is the derived quantity  $\bar{M}$ , the spin-averaged mass.

$M$ (GeV)	$M^*$ (GeV)	$\bar{M}$ (GeV)	$\Delta$ (MeV)
$D_s$	1.968 49(34)	2.1123(5)	2.0763(4)
$B_s$	5.3661(6)	5.4120(12)	5.4005(9)

TABLE XI. Spin-averaged PDG masses converted to lattice units with an error from the uncertainty in the lattice spacing  $a$ . Values of  $a$  used in the conversion can be found in Table I.

Ensemble	$a\bar{M}_{D_s}$	$a\bar{M}_{B_s}$
Fine (0.0062, 0.031)	$0.884^{+0.009}_{-0.023}$	$2.299^{+0.023}_{-0.060}$
Coarse (0.007, 0.050)	$1.242^{+0.012}_{-0.032}$	$3.230^{+0.031}_{-0.083}$
Medium-coarse (0.0097, 0.0484)	$1.529^{+0.015}_{-0.039}$	$3.977^{+0.038}_{-0.102}$

$a\bar{M}_2$  at the tuned physical strange-quark mass. We choose to tune  $\kappa$  at those same  $am'_q$ , which are  $am'_q = 0.0272$  on the (0.0062, 0.031) fine ensemble,  $am'_q = 0.03$  on the (0.007, 0.050) coarse ensemble, and  $am'_q = 0.0484$  on the (0.0097, 0.0484) medium-coarse ensemble.

To obtain the tuned  $\kappa$  for the charm (bottom) quark,  $\kappa_c$  ( $\kappa_b$ ), we want to interpolate  $\bar{M}_2$  to the PDG value of the spin-averaged  $D_s$  ( $B_s$ ) mass [45]. In practice, it is simpler to do the interpolation with the meson mass in lattice units. Hence, we linearly interpolate  $a\bar{M}_2$  to  $a\bar{M}_{\text{PDG}}$ , the PDG value for the meson mass converted to lattice units with  $a$  from Table I. This interpolation is repeated for the entire

TABLE XIII. Final tuned results for  $\kappa_c$ ,  $\kappa_b$  with the total error.

	Fine	Coarse	Medium-coarse
$\kappa_c$	0.127(2)	$0.1219^{+0.25}_{-25}$	$0.122^{+1}_{-4}$
$\kappa_b$	0.090(5)	0.082(8)	$0.077^{+5}_{-7}$

bootstrap distribution of  $a\bar{M}_2$ . We then estimate the statistical error on  $\kappa$  as the average 68% bootstrap error described in Sec. IV A. The discretization error in  $M_2$ ,  $\delta M_2$ , is given by Eq. (5.4), and is always positive. This results in a single-sided, negative error bar on  $\kappa$ . We convert  $\delta M_2$  to the error,  $\delta\kappa$ , using  $dM_2/d\kappa \approx dm_2/d\kappa$  and expressions for  $m_0a$  and  $m_2a$  given in Appendix A. The  $\delta\kappa$  are given in Table IX. The experimental errors on the PDG values are negligible. The remaining errors to consider are those which appear in the conversion between lattice and physical units. The error in the determination of  $r_1/a$  is negligible, so we only need to consider the error in  $r_1$ , given in Eq. (3.2).

The error on  $r_1$  is propagated to an error on  $a^{-1}$  and then to an error on  $a\bar{M}_{\text{PDG}}$ , denoted  $\sigma_{\text{PDG}}$ . Table X gives the values of the PDG meson masses used in this work and tabulates their spin-averaged mass and hyperfine splitting. Table XI gives the spin-averaged mass in lattice units. The uncertainty  $\sigma_{\text{PDG}}$  is propagated to  $\kappa$  using the standard error formula  $\sigma_\kappa = \sigma_{\text{PDG}}/s$ , where  $s$  is the slope used in the interpolation. Table XII gives the error budget for  $\kappa_c$  and  $\kappa_b$ , and Table XIII lists the final tuned results.

TABLE XII. Percent errors in the tuned  $\kappa$  and the total error. For several sources of uncertainty, we determined that the error was smaller than the precision of these calculations. This is indicated by an entry of “0.0” in the table.

Uncertainty	Fine	Charm			Fine	Coarse	Medium-coarse
		Coarse	Medium-coarse	Fine			
Statistical	1.26	0.57	0.53	5.0	9.1	5.6	
Discretization	(0, -0.86)	(0, -2.0)	(0, -3.46)	(0, -2.6)	(0, -4.4)	(0, -7.56)	
Sea-quark masses	0.0	0.0	0.0	0.0	0.0	0.0	
$am_s$ mistuning	0.0	0.0	0.0	0.0	0.0	0.0	
Unit conversion ( $a$ )	(+0.90, -0.35)	(+0.49, -0.19)	(+0.77, -0.30)	(+1.7, -0.64)	(+1.9, -0.72)	(+1.76, -0.66)	
Total	(1.5, 1.6)	(+ 0.75, -2.1)	(+ 0.93, -3.5)	(+ 5.3, -5.7)	(+ 9.3, -10.1)	(+ 5.9, -9.4)	

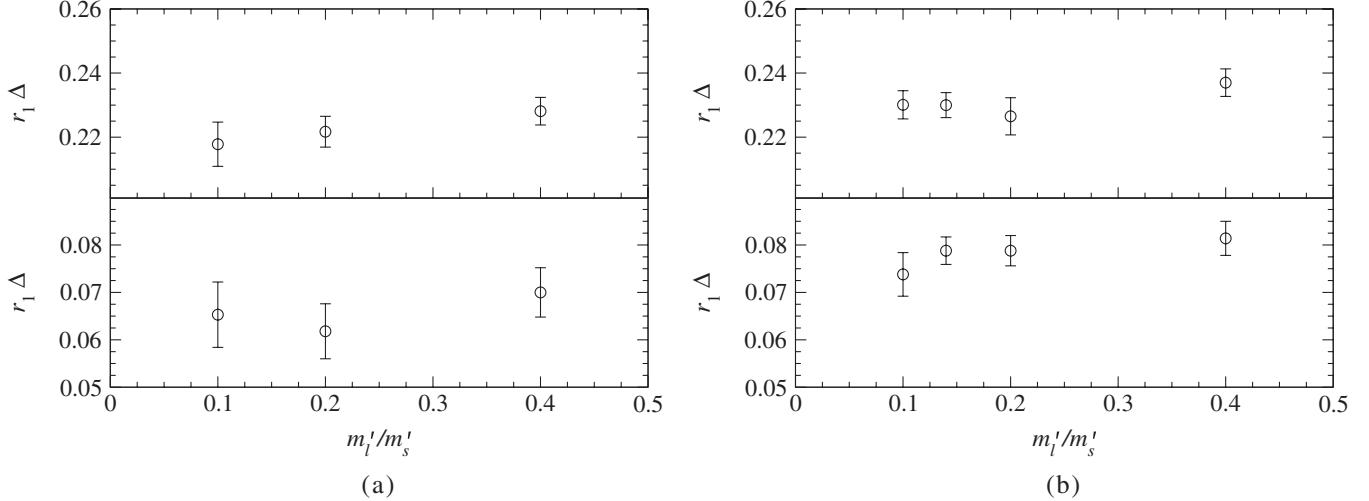


FIG. 7. The hyperfine splitting, in units of  $r_1$ , versus the ratio of the light to strange sea-quark masses  $m_l'/m_s'$  on (a) fine and (b) coarse ensembles. Errors are the average 68% bootstrap error. The upper panel in each plot is for charmlike splittings and the lower panel is for bottomlike splittings. Values of  $\kappa$  are 0.127, 0.0923 for the fine ensembles and 0.122, 0.086 for coarse ensembles. Values of  $am'_q$  are 0.0272 and 0.0415 for the fine and coarse ensembles, respectively.

### B. The rest mass and hyperfine splitting

In this section, we discuss the uncertainties in our calculation of the hyperfine splitting and compare our final results, for the  $B_s$  and  $D_s$  systems, with the PDG values. To support the discussion, we tabulate our results for the pseudoscalar and vector meson rest masses and the hyperfine splitting,  $aM$ ,  $aM^*$ ,  $a\Delta_1$ ,  $r_1\Delta_1$ , in Tables XXI, XXII, XXIII, XXIV, XXV, and XXVI in Appendix C. Statistical errors in these tables are the average 68% bootstrap errors described in Sec. IV A. The other errors we consider are the mistuning of the valence strange-quark mass, unphysical sea-quark masses, the uncertainty in the tuning of  $\kappa$ , discretization effects, and the conversion to physical units. For the central value, at each lattice spacing, we take  $a\Delta_1$  at the tuned values of  $\kappa_c$  and  $\kappa_b$ , linearly interpolating in  $\kappa$  when necessary.

PDG results for the hyperfine splitting show a weak dependence on the light-quark valence mass, so we expect the mistuning in the simulated valence strange-quark mass to have a negligible effect.<sup>3</sup> The simulation valence masses  $am'_q = 0.0272, 0.03, 0.0484$  for the fine, coarse, and medium-coarse lattices, respectively, differ from the physical  $am_s$  given in Table I by 0.0020, 0.0044, 0.0058, respectively. Tables XXI, XXII, and XXIII show that, indeed, these small mistunings have a negligible effect on the hyperfine splitting. Hence, we do not interpolate to  $am_s$ ; rather, we take  $a\Delta_1$  at the valence masses  $am'_q$  listed above as the result at the physical strange valence-quark mass and take the error for this approximation to be negligible.

<sup>3</sup>For  $X = B$  or  $D$ , the difference between the  $M_{X_s^*} - M_{X_s}$  splitting and the  $M_{X^*} - M_X$  splitting is measured to be about 1% or less [45].

To estimate the error due to the nonphysical values of the sea-quark masses we use partially quenched chiral perturbation theory. The needed expression is derived in Appendix D and we repeat Eq. (D1) here for convenience. The hyperfine splitting  $M_x^* - M_x$  of a heavy-light meson with light-valence quark  $x$  is

$$\begin{aligned} M_x^* - M_x = & \Delta - \frac{\Delta g_\pi^2}{8\pi^2 f^2} \delta_{\log} + 2\Delta^{(\sigma)}(2m_l + m_s) \\ & + 2\Delta^{(a)} m_x, \end{aligned} \quad (6.1)$$

where  $\delta_{\log}$  contains the chiral logs,  $m_l$  and  $m_s$  are the light and strange sea-quark masses, and  $\Delta^{(\sigma)}$  and  $\Delta^{(a)}$  are counter terms which must be determined from the lattice data. Working at a fixed value of  $m_x$ , we can use the difference of splittings at different values of  $m_l$  to determine  $\Delta^{(\sigma)}$ . Given  $\Delta^{(\sigma)}$ , we can find the difference between the splitting at simulation values of  $(m_l', m_s')$  and the physical values  $(m_{l,\text{phys}}, m_{s,\text{phys}})$ . We take this difference as the error due to the nonphysical sea-quark masses.

We have tabulated values of the hyperfine splitting in physical units,  $r_1\Delta_1$ , in Tables XXIII, XXIV, and XXV in Appendix C 2. Figure 7 shows how  $r_1\Delta_1$  varies with the light sea-quark mass on fine and coarse lattices. From Fig. 7, it is clear that, due to statistical variation in the splitting, using the difference in the central values of splittings from any two points will yield different values for  $\Delta^{(\sigma)}$ . For the fine and coarse ensembles, we look only at the  $am_l/am_s = 0.4$  to 0.1 and  $am_l/am_s = 0.4$  to 0.2 differences and take the one that gives the larger error; for medium coarse, we have no  $am_l/am_s = 0.1$  data and so take the error from the  $am_l/am_s = 0.4$  to 0.2 difference.

For the error estimate, we take  $f = 131$  MeV and  $g_\pi = 0.51$  [48]. We relate meson to quark masses by

$$M_{xy}^2 = B_0(m_x + m_y), \quad (6.2)$$

where  $B_0$  is determined empirically with  $r_1 B_0 = 6.38, 6.23, 6.43$  on the fine, coarse, and medium-coarse lattices, respectively. These values of  $B_0$  come from tree-level fits to MILC light-meson data, as described in Refs. [2,11,35]. We calculate  $\Delta^{(r)}$  for each meson type,  $B_s$  and  $D_s$ , at each lattice spacing. We then calculate the difference

$$(M_x^* - M_x)_{\text{sim}} - (M_x^* - M_x)_{\text{phys}}, \quad (6.3)$$

where the subscript “sim” (“phys”) denotes simulation (physical) sea-quark mass inputs ( $am_l, am_s$ ). For the physical masses, we use  $(am_{l,\text{phys}}, am_{s,\text{phys}}) = (0.000\,92, 0.0252), (0.001\,25, 0.0344), (0.001\,54, 0.0426)$  for the fine, coarse, and medium-coarse lattices, respectively. These values of the quark masses are taken from Ref. [11], after adjustment for the  $r_1$  scale used here. The simulation masses are those on the  $(0.0062, 0.031)$  fine,  $(0.007, 0.050)$  coarse, and  $(0.0097, 0.0484)$  medium-coarse ensembles. The error calculated in this manner is labeled “sea-quark masses” in Tables XIV and XV.

For the uncertainty in  $a\Delta_1$  due to the error in  $\kappa$ , recall that the non-negligible sources of the error in  $\kappa$ , from Table XII in Sec. VIA, are statistics, units conversion, and of the discretization error in  $M_2$ . Because we want to consider discretization errors separately from all others, we start by considering only the  $\kappa$ -tuning error that comes from statistics and units-conversion. To convert the error in  $\kappa$  to an error in  $a\Delta_1$ , we look at the change in  $a\Delta_1$  between two values of  $\kappa$  on the  $(0.0062, 0.031)$  fine,  $(0.007, 0.050)$  coarse, and  $(0.0097, 0.0484)$  medium-coarse ensembles; specific values can be found in Tables XXI, XXII, and XXIII. This is the error labeled “ $\kappa$  tuning” in Tables XIV and XV.

For the  $D_s(B_s)$  meson, Table XIV and XV gives the error budget for  $a\Delta_1$  at each lattice spacing, from all sources *except* discretization. These are statistics, valence-mass mistuning, unphysical sea-quark masses, and  $\kappa$  tuning. In Fig. 8, these values are plotted as black, filled dots.

We now consider the three, distinct sources of the discretization error in  $a\Delta_1$ . The first is indirect, coming from the discretization error in  $aM_2$ , which is propagated to an error on  $\kappa$  as discussed in Sec. VIA. This error can be traced to a mismatch between the spin-independent  $O(\mathbf{p}^4)$  terms in Eq. (2.8) (not given explicitly) and the corresponding terms in the effective Lagrangian for continuum QCD. These terms contribute to  $aM_2$  as discussed in Appendix A. The second source of the discretization error is a direct result of the lattice-continuum mismatch of the dimension-seven operator  $\{i\boldsymbol{\sigma} \cdot \mathbf{B}, \mathbf{D}^2\}$  [23].<sup>4</sup> The third source of the discretization error is the  $O(\alpha_s)$  mismatch in the coefficient of the  $i\boldsymbol{\sigma} \cdot \mathbf{B}$  operator in Eq. (2.10). For the discussion

<sup>4</sup>Other dimension-six and -seven operators are either redundant, loop suppressed, or known to have small coefficients [23].

TABLE XIV. Percent errors in the hyperfine splitting,  $a\Delta_1$ , of  $D_s$  *not* including discretization effects.

Uncertainty	Fine	Coarse	Medium-coarse
Statistical	2.2	1.9	1.9
$\kappa$ tuning	(8.8, -7.5)	(4.0, -3.1)	(4.0, -2.7)
Valence $m_s$	0	0	0
Sea-quark masses	3.6	5.4	6.9
Total	(10, -9)	(7, -7)	(8, -8)

TABLE XV. Percent errors in the hyperfine splitting,  $a\Delta_1$ , of  $B_s$  *not* including discretization effects.

Uncertainty	Fine	Coarse	Medium-coarse
Statistical	9.5	4.0	5.6
$\kappa$ tuning	(12, -11)	(17, -17)	(11, -10)
Valence $m_s$	0	0	0
Sea-quark masses	17	7.8	2.6
Total	(23, -22)	(19, -19)	(13, -12)

of the error estimates below, it is useful to recall that the heavy-quark dynamics associate  $m_2$  with the physical quark mass. Mismatches between  $m_2$  and the generalized masses associated with other operators capture the heavy-quark discretization effects. We now give numerical estimates of the error from each source.

Our estimate of the discretization error in  $a\bar{M}_2$  and its inclusion in the error on  $\kappa$  is discussed in Sec. VIA. In Fig. 8, the value of  $r_1\Delta_1$  with an error that includes *only* the uncertainty due to the discretization error on  $\kappa$  is shown as an open (blue) circle with a dashed error bar. Note, as described in Sec. VB, this uncertainty estimate depends on one’s choice of  $\Lambda_{\text{QCD}}$ . In this paper, we use  $\Lambda_{\text{QCD}} = 0.7$  GeV. Choosing  $\Lambda_{\text{QCD}} = 0.5$  GeV would cut the error on  $\kappa$  in half and decrease the error on  $r_1\Delta_1$ .

Next we estimate the contribution from the dimension-seven operator  $\{i\boldsymbol{\sigma} \cdot \mathbf{B}, \mathbf{D}^2\}$ . Using the notation of Ref. [23], summarized in Sec. A 4, this operator’s contribution to the hyperfine splitting has a coefficient

$$\frac{1}{(m_B' a)^3} = \frac{1}{(m_4 a)^3}, \quad (6.4)$$

where the equality holds at the tree level for the choices of parameters in our action. The difference between  $am_4$  and  $am_2$  captures the discretization error. The fractional error in the hyperfine splitting due to this mismatch is

$$(a\Lambda_{\text{QCD}})^2 2am_2 \left[ \frac{1}{(2am_4)^3} - \frac{1}{(2am_2)^3} \right]. \quad (6.5)$$

This error is plotted as a (green) dash-dot line on an X in Fig. 8. It would be added in quadrature with the error on the filled dot, if it were to be included in the total error. Again

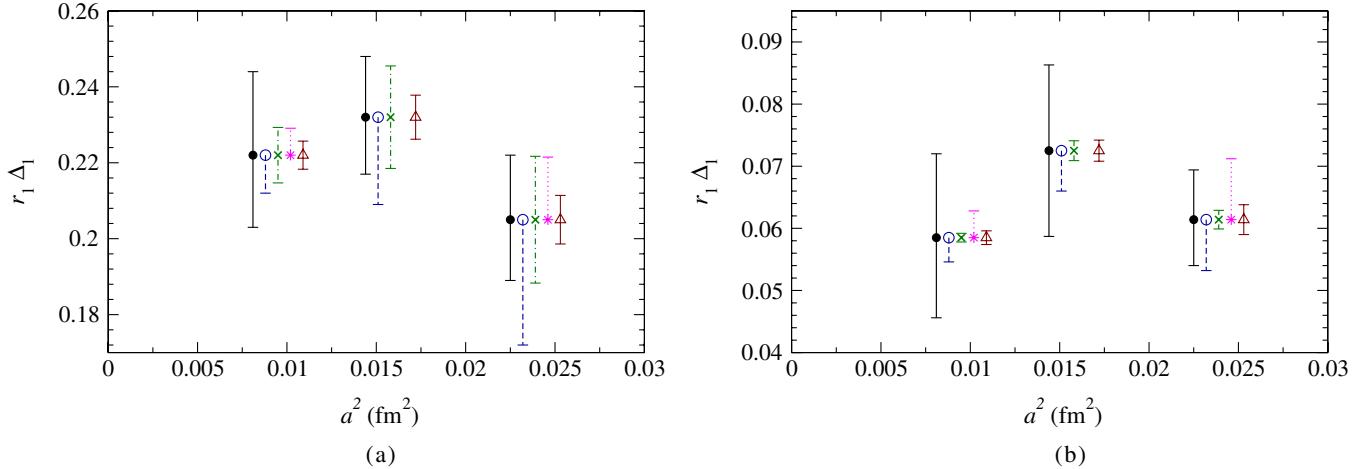


FIG. 8 (color online). Hyperfine splittings in  $r_1$  units versus the squared lattice spacing  $a^2$  ( $\text{fm}^2$ ) for the (a)  $D_s$  meson and (b)  $B_s$  meson. Filled (black) dots with a solid error bar show the splitting with an error from all sources *except* discretization. Open (blue) circles with a dashed error bar show the splitting with an error that also includes discretization error effects in  $\kappa$ . (Green) X's with dash-dotted error bars show the estimated size of discretization effects from the lattice-continuum mismatch of the dimension-seven operator  $\{i\sigma \cdot \mathbf{B}, D^2\}$ —the errors are barely visible for the  $B_s$  system. (Pink) stars with a dotted error bar show the  $O(\alpha_s)$  discretization error from the 1-loop mismatch between  $m_2$  and  $m_B$ . For the difference between the  $O(\alpha_s)$  discretization effects on the coarse lattice versus the fine and medium-coarse lattices, see the text.

we take  $\Lambda_{\text{QCD}} = 0.7 \text{ GeV}$ , but choosing  $\Lambda_{\text{QCD}} = 0.5$  would cut these error bars in half. The error from Eq. (6.5) is small for the  $D_s$  splitting at the fine lattice spacing, but increasingly large and non-negligible at the coarse and medium-coarse lattice spacings; for the  $B_s$  splitting, the error is negligible.

Finally, we turn to the effects of the  $O(\alpha_s)$  mistuning in  $c_B$ , which leads to an  $O(\alpha_s)$  mismatch between  $m_B a$  and  $m_2 a$ . Ideally,  $c_B$  should be adjusted so the coefficient of  $\bar{h}^{(+)} i\sigma \cdot \mathbf{B} h^{(+)}$  equals  $Z_B/2m_2$ , where  $Z_B$  is a coefficient with an anomalous dimension, such that  $Z_B \bar{h}^{(+)} i\sigma \cdot \mathbf{B} h^{(+)}$  is scale and scheme independent [49]. In practice,  $c_B$  is chosen in some approximation, in our case the tadpole-improved tree level of perturbation theory.

Given a value of  $c_B$ , our simulations produce

$$M_1^* - M_1 = \Delta_1 = \frac{4\lambda_2}{2m_B(c_B)}. \quad (6.6)$$

From Eq. (A26), we see that  $1/am_B$  has a contribution  $c_B/(1+m_0a)$ . Hence, to include the leading correction to the hyperfine splitting, we shift

$$4\lambda_2 a \left[ \frac{1}{2am_B(c_B)} \right] \rightarrow 4\lambda_2 a \left[ \frac{1}{2am_B(c_B)} + \frac{c_B^{\text{ideal}} - c_B}{2(1+m_0a)} \right], \quad (6.7)$$

where  $c_B^{\text{ideal}}$  is the ideal choice. (Because loop corrections to  $1/am_B$  depend on  $c_B$ , subleading corrections also exist.) To estimate the error in  $\Delta_1$ , we have to estimate  $c_B^{\text{ideal}} - c_B$ . In fact, Eq. (6.7) can also be used to shift the central value of the hyperfine splitting.

Reference [50] describes preliminary work on a calculation of the one-loop corrections to  $c_B^{[1]}$ , as a function of the bare-quark mass. For all relevant values of  $m_0a$ , the one-loop effects are a small correction to the tadpole-improved Ansatz  $c_B = u_0^{-3}$ , provided that  $u_0$  is the average link in Landau gauge. On the coarse ensembles, we chose  $u_0$  this way, and we can estimate the remaining correction directly from the calculation in Ref. [50]. Given further uncertainties from higher orders, we take this small correction as an uncertainty estimate. On the medium-coarse and fine ensembles, however, we chose  $u_0^4$  to be the average plaquette. In those cases, the leading correction to  $c_B$  comes from,

$$c_B^{\text{ideal}} - c_B = u_{0,\text{LL}}^{-3} - u_{0,\text{plaq}}^{-3}, \quad (6.8)$$

where the labels refer to “Landau-gauge link” and “plaquette.” Equation (6.8) leads to significant corrections to the hyperfine splitting, so we shift  $\Delta_1$  on the medium-coarse and fine ensembles by the amount corresponding to Eq. (6.7) and (6.8). These shifts put  $\Delta_1$  at the medium-coarse and fine lattice spacings on the same footing as those at the coarse spacing. Empirically, they flatten the lattice-spacing dependence.

For the medium-coarse and fine data, we use the values of  $u_0$  given in Table XVI to calculate the shift described

TABLE XVI. Tadpole-improvement factors for the estimate of the  $O(\alpha_s)$  discretization error shown in Fig. 8.

Ensemble	$u_{0,\text{plaq}}$	$u_{0,\text{Landau}}$
Fine (0.0062, 0.031)	0.878	0.854
Medium-coarse (0.0097, 0.0484)	0.860	0.822

above. It is displayed in Fig. 8 as a (pink) star with a single-sided, positive error bar. To obtain an error bar corresponding to the one-loop correction to  $c_B$  in Ref. [50], we take  $\alpha_s(0.09 \text{ fm}) = 1/3$  and use one-loop running to obtain values of  $\alpha_s$  for the coarse and medium-coarse lattices. These corrections are shown in Fig. 8 as a (red) triangle with a solid error bar.

In summary, discretization errors in the hyperfine splitting are small at the fine lattice spacing; therefore, we take as our final results the splittings calculated on the fine lattice. In addition, since the effect of the leading  $O(\alpha_s)$  mistuning of  $c_B$  can be quantified, we shift our final central values by this amount. All other discretization errors are included in our final error. We convert our results to physical units using the values of  $r_1/a$  and  $r_1$  as listed in Table I. After including the error from the units conversion in the total, our final results for the hyperfine splittings are

$$\Delta_{D_s} = 145 \pm 15 \text{ MeV}, \quad (6.9)$$

$$\Delta_{B_s} = 40 \pm 9 \text{ MeV}. \quad (6.10)$$

These results are in good agreement with the PDG values of  $143.9 \pm 0.4 \text{ MeV}$  and  $46.1 \pm 1.5 \text{ MeV}$ , respectively.

### C. The critical hopping parameter $\kappa_{\text{crit}}$

In principle, it is possible to carry out a suite of non-perturbative heavy-quark calculations without knowing  $\kappa_{\text{crit}}$ , but in practice  $\kappa_{\text{crit}}$  is useful. In particular, it enters the construction of improved bilinear and 4-quark operators via  $m_0 a$  Eq. (2.5). It also enters the computation of matching factors such as  $Z_V$  and  $Z_A$  [18]. Note that these all amount to small corrections, so we do not need a very precise determination of  $\kappa_{\text{crit}}$ . Equation (2.5) shows that it does not have to be much better determined than  $\kappa_c$  and  $\kappa_b$ .

TABLE XVII. Values of  $\kappa_{\text{crit}}$  by ensemble. “ $u_0$  used” gives the origin of the  $u_0$  value used in the  $\kappa_{\text{crit}}$  determination.  $\kappa_{\text{crit}}$  values are given in two columns. The first  $\kappa_{\text{crit}}$  column contains values which were determined by a fit. The second  $\kappa_{\text{crit}}$  column contains values which were estimated from fitted values at the same (approximate) lattice spacing. The last column gives the fit method used in the determination, explained in the text.

Lattice	$(am'_s, am'_s)$	$u_0$ used	Iterated fit	$\kappa_{\text{crit}}$ Direct fit	Estimated
Fine	(0.0031, 0.031)	Landau-gauge link			0.1372
	(0.0062, 0.031)	Landau-gauge link	0.1372		
	(0.0062, 0.031)	plaquette	0.1391		
	(0.0124, 0.031)	Landau-gauge link	0.1372		
Coarse	(0.005, 0.050)	Landau-gauge link			0.1379
	(0.007, 0.050)	Landau-gauge link			0.1379
	(0.010, 0.050)	Landau-gauge link	0.1379		
	(0.020, 0.050)	Landau-gauge link	0.1378		
	(0.030, 0.050)	Landau-gauge link	0.1377		
Medium-coarse	(0.0097, 0.0484)	Plaquette			0.1424
	(0.0194, 0.0484)	Plaquette			0.1424
	(0.0290, 0.0484)	Plaquette			0.1423

A nonperturbative definition of  $\kappa_{\text{crit}}$  is the value of  $\kappa$  such that the mass of a pseudoscalar meson consisting of two Wilson quarks (with the clover action) vanishes. The computation of these light-light pseudoscalar meson masses shares code with the work reported here and in Ref. [8], and it is convenient to report the analysis here. The value of  $\kappa_{\text{crit}}$  depends on  $u_0$  via our choice of clover coupling,  $c_B = c_E = u_0^{-3}$ . In this and other work [3–8],  $u_0$  has been set sometimes from the average plaquette and sometimes from the average link in Landau gauge. The prescription for  $u_0$  used in each  $\kappa_{\text{crit}}$  determination is given in column four of Table XVII.

The determination of  $\kappa_{\text{crit}}$  is carried out on a subset of the available configurations, 50–100 configurations for the fine ensembles and 400–600 for the coarse and medium-coarse. We compute two-point correlators for a range of  $\kappa$  that yields meson masses of about  $M_{\text{PS}} = 450\text{--}900 \text{ MeV}$  on the fine ensembles, 650–1100 MeV on the coarse ensembles, and 550–950 MeV on the medium-coarse ensembles. It is impractical to push to lower  $M_{\text{PS}}$  due to exceptional configurations.  $M_{\text{PS}}$  is a function of the quark mass, which we parametrize as the tree-level, tadpole-improved kinetic or rest mass. In the relevant region,  $m_{1,2}a = m_0a[1 - \frac{1}{2}m_0a] + O((m_0a)^3)$ , so both pertain equally well. The meson masses can be fit to a polynomial ansatz

$$a^2 M_{\text{PS}}^2(\kappa) = A + B a m_2(\kappa, \kappa_{\text{crit}}) + C a^2 m_2^2(\kappa, \kappa_{\text{crit}}) \quad (6.11)$$

(or  $m_1$  instead of  $m_2$ ), where  $A = 0$  when  $\kappa_{\text{crit}}$  is correctly adjusted.

We use two techniques to determine  $\kappa_{\text{crit}}$ . One method starts with a reasonable value of  $\kappa_{\text{crit}}$  and fits Eq. (6.11) to obtain  $A$ ,  $B$ , and  $C$ , which depend implicitly on  $\kappa_{\text{crit}}$ . A better trial value of  $\kappa_{\text{crit}}$  is chosen, and the process is iterated until a  $\kappa_{\text{crit}}$  is found such that  $A = 0$ . We call this the “iterated fit.”

The second method freezes  $A$  to zero, and then  $B$ ,  $C$ , and  $\kappa_{\text{crit}}$  are the fit parameters. We call this the “direct fit.” On several ensembles the  $\kappa_{\text{crit}}$  values were simply estimated from the other ensembles with the same (approximate) lattice spacing, these are labeled as “estimated.”

Table XVII contains our results for  $\kappa_{\text{crit}}$ , indicating the method used. The table does not include error bars for  $\kappa_{\text{crit}}$ , but we believe that the results are correct to the number of significant figures shown, even though the range of  $M_{\text{PS}}$  is high. We carried out several tests to verify this accuracy. We compared linear iterated fits [i.e.,  $C = 0$  in Eq. (6.11)] to the baseline quadratic. We also compared direct fits with and without the (continuum) chiral log. These test show that higher order or log contributions do not alter our values of  $\kappa_{\text{crit}}$  significantly. We fit comparable data with staggered valence quarks allowing  $(m_0 a)_{\text{crit}} \neq 0$ , thereby testing whether a range of such large  $M_{\text{PS}}$  skews the results. None of these tests suggests an error larger than a few in the fourth digit. Such errors are negligible compared to those for  $\kappa_c$  and  $\kappa_b$ —see Tables XII and XIII—when forming  $m_0 a$  with Eq. (2.5).

## VII. SUMMARY AND OUTLOOK

An accurate and precise determination of  $\kappa_c$  and  $\kappa_b$  is important for all calculations using the Fermilab action [3–8]. In this analysis, the error on  $\kappa_b$  is dominated by statistics, and the error on  $\kappa_c$  receives approximately equal contributions from statistics and discretization effects. These errors play a significant role on quantities as diverse as  $D$ - and  $B$ -meson decay constants [3] and the quarkonium hyperfine splitting [8]. Our final results for  $\kappa_c$  and  $\kappa_b$  are given in Table XIII.

Another ingredient that is useful for matrix elements [3–5] is the additive renormalization of the bare-quark mass or, equivalently,  $\kappa_{\text{crit}}$ . The improvement and matching of the operators needed to compute these matrix elements depends mildly on  $\kappa_{\text{crit}}$  via  $m_0 a$  [18]. Our final results for  $\kappa_{\text{crit}}$  are given in Table XVII.

The key ingredient needed to determine  $\kappa_c$  and  $\kappa_b$  is a computation of the pseudoscalar and vector heavy-strange meson masses. These can be combined to yield the hyperfine splitting for  $D_s$  and  $B_s$  mesons. Our final results for the hyperfine splittings are given in Eqs. (6.9) and (6.10). Both are in good agreement with the corresponding PDG averages. These results bolster confidence in the tuning of  $\kappa_c$  and  $\kappa_b$ , as well as the choice  $c_B = u_0^{-3}$ . Further tests of these choices come from related calculations of the quarkonium spectrum [8]. With detailed attention given to the connection between action parameters and mass splittings, those results are found to be consistent with experiment within the expected uncertainties.

Improved determinations of  $\kappa_c$ ,  $\kappa_b$ , and  $\kappa_{\text{crit}}$  for the medium-coarse, coarse, and fine ensembles are underway with higher statistics, as well as calculations on the new superfine ( $a \approx 0.06$  fm), and ultrafine ( $a \approx 0.045$  fm)

lattices. The increased statistics will also allow us to use higher momentum data and fit to the  $O(p^4)$  terms in the dispersion relation. Refinements in the determination and use of  $r_1/a$  are allowing for a better understanding of sea-quark effects which will be needed as the statistical error on  $aM_2$  decreases. We are also investigating the use of twisted boundary conditions [51] which will allow us to obtain data points at lower momenta.

As uncertainties in  $M_2$  and  $M_1$  decrease, there will be a need for a better understanding of the chiral behavior of these masses. One-loop,  $O(\Lambda/m_Q)$  chiral perturbation theory results exist for continuum QCD [47]. The extension to staggered chiral perturbation theory should be straightforward, and would allow us to extrapolate the light-valence mass to the physical up/down quark mass and determine the hyperfine splittings of the  $B^\pm$  and  $D^\pm$  mesons. In this paper, we have included the partially quenched expression for the hyperfine splitting in Appendix D, since it is useful in estimating uncertainties from the unphysical sea-quark masses.

In addition, tuned values of  $\kappa_c$ ,  $\kappa_b$ , and  $\kappa_{\text{crit}}$  combined with one-loop (lattice) perturbation theory can yield determinations of the pole masses  $m_1$  and  $m_2$  for both charmed and bottom quarks,<sup>5</sup> which can be converted to the potential-subtracted,  $\overline{\text{MS}}$ , and other schemes [6]. Quark masses combined with staggered chiral perturbation theory for the  $B^\pm$  and  $D^\pm$  mesons, can yield *ab initio* calculations of HQET matrix elements [22,52], which are used to calculate the Cabibbo-Kobayashi-Maskawa matrix element  $|V_{cb}|$  via inclusive decay measurements. Finally, improved determinations of the oscillating-state energy  $E^p$  could make determinations of the experimentally accessible masses of the positive parity states,  $D_{s0}^*(2317)$  and  $D_{s1}(2460)$  [53] a viable option [54].

## ACKNOWLEDGMENTS

Computations for this work were carried out on facilities of the USQCD Collaboration, which are funded by the Office of Science of the U.S. Department of Energy. This work was supported in part by the U.S. Department of Energy under Grant Nos. DE-FC02-06ER41446 (C. D., L. L.), DE-FG02-91ER40661 (S. G.), DE-FG02-91ER40677 (A. X. K., E. G., R. T. E.), DE-FG02-91ER40628 (C. B., E. D. F.), and DE-FG02-04ER-41298 (D. T.); the National Science Foundation under Grant Nos. PHY-0555243, NPHY-0757333, PHY-0703296 (C. D., L. L.), PHY-0757035 (R. S.), PHY-0704171 (J. E. H.), and PHY-0555235 (E. D. F.); the Universities Research Associates (R. T. E., E. G.), and the M. Hildred Blewett Scholarship of the American Physical Society (E. D. F.). This manuscript has been co-authored by an employee of Brookhaven Science Associates, LLC, under

<sup>5</sup>The determination of the quark mass from  $m_1$  requires a nonperturbative calculation of the binding energy as defined by  $M_1 - m_1$  [6].

Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. R. S. V. acknowledges support from BNL via the Goldhaber Distinguished Fellowship. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

## APPENDIX A: DISCRETIZATION ERROR IN THE KINETIC MESON MASS

In this appendix, we present a semiquantitative estimation of the discretization error in the kinetic mass of heavy-light hadrons. We use a formalism that applies when both quarks are nonrelativistic, even though this approximation is not good for the light quark in a heavy-light meson. *A posteriori*, we examine two ways to re-interpret the resulting formula for a relativistic light quark. Both estimates are numerically the same, so we proceed to use the formula in Sec. V B.

In what follows, the generalized masses  $m_1, m_2, m_4$  and the coefficient  $w_4$  are used to describe the discretization errors. Expressions for them when using the Fermilab action are in Ref. [14] or [23] and are given at the end of this appendix for convenience. We assume that the light quark ( $s$ ) has a mass in lattice units  $m_s a \ll 1$  and makes no significant contribution to the discretization error.

The bound state's kinetic mass can be read off from its kinetic energy (by definition). It will have a kinematic contribution, from the constituents' kinetic energy, and a dynamical contribution, from the interaction that binds the constituents. We consider each in turn.

### 1. Contributions from constituents' kinetic energy

The hadron of interest is a heavy-strange meson, a bound state of a heavy quark  $Q$  (momentum  $\mathbf{Q}$ ) and a strange antiquark  $s$  (momentum  $\mathbf{s}$ ). The nonrelativistic kinetic energy is

$$\begin{aligned} T = m_{1Q} + \frac{\mathbf{Q}^2}{2m_{2Q}} - \frac{(\mathbf{Q}^2)^2}{8m_{4Q}^3} - \frac{1}{6}w_{4Q}a^3\sum_i Q_i^4 + m_{1s} \\ + \frac{\mathbf{s}^2}{2m_{2s}} - \frac{(\mathbf{s}^2)^2}{8m_{4s}^3} - \frac{1}{6}w_{4s}a^3\sum_i s_i^4. \end{aligned} \quad (\text{A1})$$

The binding energy is communicated to the bound-state kinetic mass via the terms quartic in the momenta and via corrections to the potential, given below. In general, the lattice breaks relativistic invariance, so  $m_1 \neq m_2 \neq m_4, w_4 \neq 0$ . Rewriting the kinetic energy in center-of-mass coordinates

$$\mathbf{Q} = \frac{m_{2Q}}{m_{2Q} + m_{2s}}\mathbf{P} + \mathbf{p}, \quad (\text{A2})$$

$$\mathbf{s} = \frac{m_{2s}}{m_{2Q} + m_{2s}}\mathbf{P} - \mathbf{p}, \quad (\text{A3})$$

$$\mathbf{P} = \mathbf{Q} + \mathbf{s}, \quad (\text{A4})$$

$$\mathbf{p} = \frac{m_{2s}\mathbf{Q} - m_{2Q}\mathbf{s}}{m_{2Q} + m_{2s}}, \quad (\text{A5})$$

one finds

$$\begin{aligned} T = m_{1Q} + m_{1s} + \frac{\mathbf{P}^2}{2(m_{2Q} + m_{2s})} + \frac{\mathbf{p}^2}{2\mu_2} \\ - \frac{\mathbf{P}^2\mathbf{p}^2 + 2(\mathbf{P} \cdot \mathbf{p})^2}{4(m_{2Q} + m_{2s})^2} \left[ \frac{m_{2Q}^2}{m_{4Q}^3} + \frac{m_{2s}^2}{m_{4s}^3} \right] \\ - a^3 \sum_i \frac{P_i^2 p_i^2}{(m_{2Q} + m_{2s})^2} (w_{4Q}m_{2Q}^2 + w_{4s}m_{2s}^2) + \dots, \end{aligned} \quad (\text{A6})$$

$$\frac{1}{\mu_2} = \frac{1}{m_{2Q}} + \frac{1}{m_{2s}}. \quad (\text{A7})$$

The only quartic terms shown are those quadratic in  $\mathbf{P}$ ; the omitted terms are not smaller; they just do not contribute to the bound state's kinetic energy. The objective is to collect all terms quadratic in  $\mathbf{P}$ , because their overall coefficient will yield the bound state's kinetic mass.

### B. Contribution from the interaction: Breit equation

To obtain the two-particle system's potential energy, one has to work out the scattering amplitude from one-gluon exchange, obtaining an expression called the Breit equation [46,55].

In momentum space, for the color-singlet channel

$$\begin{aligned} V(\mathbf{K}) = -C_F g^2 D_{\mu\nu}(K) \mathcal{N}_Q(\mathbf{Q} + \mathbf{K}) \bar{u}(\xi', \mathbf{Q} + \mathbf{K}) \\ \times \Lambda_Q^\mu(Q + K, Q) u(\xi, \mathbf{Q}) \mathcal{N}_Q(\mathbf{Q}) \\ \times \mathcal{N}_s(s) \bar{v}(\xi, s) \Lambda_s^\nu(s, s - K) v(\xi', s - K) \\ \times \mathcal{N}_s(s - K), \end{aligned} \quad (\text{A8})$$

where  $D_{\mu\nu}$  is the (lattice) gluon propagator,  $\Lambda_q^\mu$  is the lattice vertex function (for  $q = Q, s$ ), and  $\mathcal{N}_q$  is an external-line factor needed with the normalization conditions on spinors employed here [14,23]. (In continuum field theory,  $\mathcal{N} = \sqrt{m/E}$ .)

To the accuracy needed here, the gluon propagator can be replaced with the continuum propagator. The heavy-quark line is

$$\begin{aligned} J_Q^4 = \mathcal{N}_Q(\mathbf{Q} + \mathbf{K}) \bar{u}(\xi', \mathbf{Q} + \mathbf{K}) \\ \times \Lambda_Q^4(Q + K, Q) u(\xi, \mathbf{Q}) \mathcal{N}_Q(\mathbf{Q}) \\ = \bar{u}(\xi', \mathbf{0}) \left[ 1 - \frac{\mathbf{K}^2 - 2i \cdot (\mathbf{K} \times \mathbf{Q})}{8m_{EQ}^2} + \dots \right] u(\xi, \mathbf{0}), \end{aligned} \quad (\text{A9})$$

$$\begin{aligned} J_Q &= \mathcal{N}_Q(Q + K)\bar{u}(\xi', Q + K) \\ &\times \Lambda_Q(Q + K, Q)u(\xi, Q)\mathcal{N}_Q(Q) \\ &= -i\bar{u}(\xi', \mathbf{0})\left[\frac{Q + \frac{1}{2}K}{m_{2Q}} + \frac{i \times K}{2m_{BQ}} + \dots\right]u(\xi, \mathbf{0}), \quad (\text{A10}) \end{aligned}$$

and, to the extent that the strange antiquark is nonrelativistic, one has a similar expression for the antiquark line  $J_s^\nu = \mathcal{N}_s(s)\bar{v}(\xi, s)\Lambda_s^\nu(s, s - K)v(\xi', s - K)\mathcal{N}_s(s - K)$ .

In the Coulomb gauge,

$$D_{44}(K) = \frac{1}{K^2}, \quad D_{ij}(K) = \frac{1}{K^2}\left(\delta^{ij} - \frac{K^i K^j}{K^2}\right), \quad (\text{A11})$$

and the other components vanish. Thus, noting that  $K_4 = i[(Q + K)^2 - Q^2]/2m_Q$  is subleading,

$$\begin{aligned} V(K) &= -C_F g^2 \left[ \frac{1}{K^2} - \left( \frac{1}{8m_{EQ}^2} + \frac{1}{8m_{Es}^2} \right) \right. \\ &\quad \left. - \frac{1}{m_{2Q} m_{2s}} \left( Q \cdot s - \frac{Q \cdot K K \cdot s}{K^2} \right) \frac{1}{K^2} \right] \\ &\quad + \text{spin-dependent terms.} \quad (\text{A12}) \end{aligned}$$

Let us discuss each part of the bracket in turn. The leading term yields, after Fourier transforming to position space, the  $1/r$  potential. The second yields a contact term proportional to  $\delta(r)$ : it is a relativistic correction to the bound state's rest mass, so it is of no further interest here. Similarly, the spin-dependent terms do not contribute to the bound state's kinetic energy, so they are not written out.

]

$$\begin{aligned} E(\mathbf{P}) &= m_{1Q} + m_{1s} + \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} - \left\langle \frac{C_F \alpha_s}{r} \right\rangle + \frac{\mathbf{P}^2}{2(m_{2Q} + m_{2s})} \left[ 1 - \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2(m_{2Q} + m_{2s})} + \frac{1}{(m_{2Q} + m_{2s})} \left\langle \frac{C_F \alpha_s}{r} \right\rangle \right] \\ &\quad + \frac{\mathbf{P}^2}{2(m_{2Q} + m_{2s})^2} \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} \left[ 1 - \mu_2 \left( \frac{m_{2Q}^2}{m_{4Q}^3} + \frac{m_{2s}^2}{m_{4s}^3} \right) \right] + \frac{P_i P_j}{(m_{2Q} + m_{2s})^2} \frac{\langle p_i p_j \rangle}{2\mu_2} \left[ 1 - \mu_2 \left( \frac{m_{2Q}^2}{m_{4Q}^3} + \frac{m_{2s}^2}{m_{4s}^3} \right) \right] \\ &\quad - a^3 \sum_i \frac{P_i^2 \langle p_i^2 \rangle}{(m_{2Q} + m_{2s})^2} (w_{4Q} m_{2Q}^2 + w_{4s} m_{2s}^2) + \dots \quad (\text{A17}) \end{aligned}$$

The first four terms of Eq. (A17) show the binding energy adding to the quarks' rest masses to form the bound state's rest mass,

$$M_1 = m_{1Q} + m_{1s} + \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} - \left\langle \frac{C_F \alpha_s}{r} \right\rangle. \quad (\text{A18})$$

The fifth term in Eq. (A17) shows the same binding energy modifying the kinetic energy. The remaining terms are discretization errors. In general they are a bit messy, but they simplify for the S-wave states we use to tune  $\kappa$ . Then  $\langle p_i p_j \rangle = \frac{1}{3} \delta_{ij} \langle \mathbf{p}^2 \rangle$ , whence

$$E(\mathbf{P}) = M_1 + \frac{\mathbf{P}^2}{2M_2} + \dots, \quad (\text{A19})$$

The remaining exhibited contributions do contribute to the bound state's kinetic energy, when  $Q$  and  $s$  are eliminated in favor of  $P$  and  $p$ .

Next we Fourier transform from  $K$  to  $r$  using

$$\int \frac{d^3 K}{(2\pi)^3} \frac{e^{ir \cdot K}}{K^2} = \frac{1}{4\pi r}, \quad (\text{A13})$$

$$\int \frac{d^3 K}{(2\pi)^3} \frac{K_i K_j e^{ir \cdot K}}{(K^2)^2} = \frac{1}{2} (\delta_{ij} + r_i \nabla_j) \int \frac{d^3 K}{(2\pi)^3} \frac{e^{ir \cdot K}}{K^2}. \quad (\text{A14})$$

Following with the substitution of  $P$  and  $p$  for  $Q$  and  $s$  this yields

$$\begin{aligned} V(\mathbf{r}, \mathbf{P}, \mathbf{p}) &= -\frac{C_F \alpha_s}{r} \left[ 1 - \frac{\mathbf{P}^2}{2(m_{2Q} + m_{2s})^2} \right] \\ &\quad - r_i \nabla_j \frac{C_F \alpha_s}{r} \frac{P_i P_j}{2(m_{2Q} + m_{2s})^2} + \dots, \quad (\text{A15}) \end{aligned}$$

where the omitted terms do not influence the bound state's kinetic energy.

Note that  $K$  changes  $p$  but not  $P$ , so  $r$  is conjugate to  $p$ . To take expectation values, we use the virial theorem

$$\langle r_i \nabla_j V(\mathbf{r}) \rangle = \frac{\langle p_i p_j \rangle}{\mu_2}, \quad (\text{A16})$$

so the total energy of the bound state,  $E(\mathbf{P}) = \langle T + V \rangle$ , is

where

$$\begin{aligned} M_2 &= m_{2Q} + m_{2s} + \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} - \left\langle \frac{C_F \alpha_s}{r} \right\rangle \\ &\quad + \frac{5}{3} \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} \left[ \mu_2 \left( \frac{m_{2Q}^2}{m_{4Q}^3} + \frac{m_{2s}^2}{m_{4s}^3} \right) - 1 \right] \\ &\quad + \frac{4}{3} a^3 \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} \mu_2 (w_{4Q} m_{2Q}^2 + w_{4s} m_{2s}^2) + \dots. \quad (\text{A20}) \end{aligned}$$

The last two lines exhibit the discretization errors, which would vanish if  $m_4 = m_2$ ,  $w_4 = 0$ .

TABLE XVIII. The kinetic meson mass for bottom- and charm-type mesons on the (0.0062, 0.031) fine ensemble from fits to  $E^2(\mathbf{p}) - E^2(0)$  using  $|\mathbf{n}| \leq \sqrt{3}$ . Fits are done to obtain  $aM_2$  and  $aM_2^*$  and the results are then spin averaged. Uncertainties are the average 68% bootstrap error.  $\chi^2/\text{dof}$  with the  $p$  value in parentheses is also given. The  $p$  value is one minus the  $\chi^2$  cumulative distribution [45].

$\kappa$	$am'_q$	$aM_2$	$aM_2^*$	$a\bar{M}_2$	$aM_2$	$aM_2^*$	$\chi^2/\text{dof}$ ( $p$ )
$am'_q = 0.0272$	0.090	2.30(17)	2.31(25)	2.31(21)	0.21 (0.81)	0.24 (0.79)	
	0.0923	2.19(15)	2.22(22)	2.21(19)	0.22 (0.80)	0.35 (0.71)	
	0.093	2.16(14)	2.19(22)	2.18(18)	0.22 (0.80)	0.37 (0.69)	
	0.1256	0.860(19)	0.936(42)	0.917(32)	0.09 (0.92)	0.14 (0.87)	
	0.127	0.819(15)	0.912(39)	0.889(30)	0.36 (0.70)	0.00 (1.0)	
$am'_q = 0.031$	0.0923	2.22(14)	2.22(21)	2.22(18)	0.18 (0.84)	0.31 (0.73)	
	0.1256	0.871(18)	0.947(38)	0.928(30)	0.14 (0.87)	0.16 (0.85)	
	0.127	0.828(15)	0.918(37)	0.895(29)	0.40 (0.67)	0.00 (1.0)	

The error can be rewritten as

$$\delta M_2 = \frac{1}{3} \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} \left\{ 5 \left[ \mu_2 \left( \frac{m_{2Q}^2}{m_{4Q}^3} + \frac{m_{2s}^2}{m_{4s}^3} \right) - 1 \right] + 4a\mu_2 [w_{4Q}(m_{2Q}a)^2 + w_{4s}(m_{2s}a)^2] \right\}, \quad (\text{A21})$$

which is equivalent to Eq. (14) of Ref. [46]. Note that the error ends up being proportional to the internal kinetic energy of the bound state,  $\langle \mathbf{p}^2 \rangle / 2\mu_2$ .

### 3. Relativistic light degrees of freedom

For asqtad light quarks, the discretization errors are  $O(\alpha_s m_s^2 a^2)$  and  $O(m_s^4 a^4)$ . So, for a semiquantitative estimate of the discretization error, it should be safe to assume  $m_{4s} = m_{2s} = m_{1s} = m_s$ ,  $a^3 w_{4s} = 0$ . Equation (A21) is then

$$\delta M_2 = \frac{1}{3m_{2Q}} \frac{\langle \mathbf{p}^2 \rangle}{2\mu_2} \mu_2 \left[ 5 \left( \frac{m_{2Q}^3}{m_{4Q}^3} - 1 \right) + 4w_{4Q}(m_{2Q}a)^3 \right]. \quad (\text{A22})$$

To use this formula we need a value for  $\langle \mathbf{p}^2 \rangle$ , and we consider two possibilities. The first is to replace  $\langle \mathbf{p}^2 \rangle$  with  $\bar{\Lambda}^2$ . The reduced mass  $\mu_2$  then cancels, yielding a sensible limit even when  $m_s \rightarrow 0$ . The second is to replace the nonrelativistic kinetic energy  $\langle \mathbf{p}^2 \rangle / 2\mu_2$  with a relativistic version, namely  $\bar{\Lambda}$ . If we take a constituent quark mass  $m_s = \frac{1}{2}\bar{\Lambda}$ , then this discretization-error estimate equals that of the first approach to  $O(m_s/m_Q)$ .

### 4. The generalized masses and $w_4$

General tree-level expressions for the quark masses and  $w_4$  were originally given in Ref. [14] and succinctly recapitulated in Ref. [23]. For convenience we give them here with parameters  $\zeta = 1 = r_s$  as in our simulations

$$m_0 a = \frac{1}{u_0} \left( \frac{1}{2\kappa} - \frac{1}{2\kappa_{\text{crit}}} \right), \quad (\text{A23})$$

$$m_1 a = \ln(1 + m_0 a), \quad (\text{A24})$$

$$\frac{1}{m_2 a} = \frac{2}{m_0 a(2 + m_0 a)} + \frac{1}{1 + m_0 a}, \quad (\text{A25})$$

$$\frac{1}{m_B a} = \frac{2}{m_0 a(2 + m_0 a)} + \frac{c_B}{1 + m_0 a}, \quad (\text{A26})$$

$$\frac{1}{4m_E^2 a^2} = \frac{1}{[m_0 a(2 + m_0 a)]^2} + \frac{c_E}{m_0 a(2 + m_0 a)}, \quad (\text{A27})$$

$$\begin{aligned} \frac{1}{m_4^3 a^3} = & \frac{8}{[m_0 a(2 + m_0 a)]^3} + \frac{4 + 8(1 + m_0 a)}{[m_0 a(2 + m_0 a)]^2} \\ & + \frac{1}{(1 + m_0 a)^2}, \end{aligned} \quad (\text{A28})$$

$$w_4 = \frac{2}{m_0 a(2 + m_0 a)} + \frac{1}{4(1 + m_0 a)}. \quad (\text{A29})$$

These expressions and Eq. (A22) are used to obtain Table IX.

## APPENDIX B: TABLES OF THE KINETIC MASS

In this appendix, we tabulate values of the pseudoscalar, vector, and spin-averaged kinetic mass,  $aM_2$ ,  $aM_2^*$ , and  $\bar{M}_2$ , respectively. Values are given for all combinations of  $\kappa$  and  $am'_q$  on the ensembles used for tuning  $\kappa$ .  $\chi^2/\text{dof}$  and the  $p$  value, one minus the  $\chi^2$  cumulative distribution [45], from the dispersion relation fits are also given.

## APPENDIX C: TABLES OF $M_1 = E(0)$ AND THE HYPERFINE SPLITTING

In this appendix, we tabulate the hyperfine splitting  $a\Delta_1$  and  $r_1\Delta_1$  discussed in Sec. VI B.

TABLE XIX. Same as Table XVIII but for mesons on the (0.007, 0.050) coarse ensemble.

	$\kappa$	$aM_2$	$aM_2^*$	$a\bar{M}$	$aM_2$	$\chi^2/\text{dof } (p)$	$aM_2^*$
$am'_q = 0.03$	0.074	3.78(49)	3.64(54)	3.67(50)	1.12 (0.33)	0.17 (0.84)	
	0.086	2.93(21)	3.03(33)	3.01(29)	0.28 (0.76)	0.21 (0.81)	
	0.093	2.50(14)	2.66(24)	2.62(21)	0.05 (0.95)	0.11 (0.90)	
	0.119	1.263(16)	1.402(43)	1.368(34)	0.84 (0.43)	0.20 (0.82)	
	0.122	1.132(17)	1.270(46)	1.236(37)	0.35 (0.70)	0.34 (0.71)	
	0.124	1.038(16)	1.161(43)	1.130(33)	0.28 (0.76)	0.52 (0.60)	
$am'_q = 0.0415$	0.074	3.66(35)	3.75(53)	3.73(48)	0.26 (0.77)	0.23 (0.79)	
	0.086	2.99(19)	3.09(28)	3.06(25)	0.46 (0.63)	0.48 (0.62)	
	0.093	2.57(13)	2.75(25)	2.70(21)	0.14 (0.87)	0.31 (0.73)	
	0.119	1.292(15)	1.456(41)	1.415(33)	0.88 (0.41)	0.38 (0.69)	
	0.122	1.157(17)	1.310(44)	1.272(36)	0.19 (0.83)	0.24 (0.79)	
	0.124	1.065(15)	1.200(43)	1.166(34)	0.15 (0.86)	0.47 (0.63)	

TABLE XX. Same as Table XVIII but for mesons on the (0.0097, 0.0484) medium-coarse ensemble.

	$\kappa$	$aM_2$	$aM_2^*$	$a\bar{M}$	$aM_2$	$\chi^2/\text{dof } (p)$	$aM_2^*$
$am'_q = 0.0484$	0.070	4.54(32)	4.53(45)	4.53(41)	0.55 (0.58)	0.78 (0.46)	
	0.080	3.79(19)	3.77(27)	3.78(24)	0.54 (0.58)	1.19 (0.31)	
	0.115	1.747(25)	1.825(47)	1.805(37)	1.32 (0.27)	0.36 (0.70)	
	0.125	1.304(12)	1.415(32)	1.387(26)	1.23 (0.29)	0.05 (0.95)	
$am'_q = 0.0387$	0.070	4.47(35)	4.44(50)	4.44(46)	0.47 (0.63)	0.72 (0.49)	
	0.080	3.73(21)	3.70(31)	3.71(28)	0.53 (0.59)	1.07 (0.34)	
	0.115	1.725(28)	1.804(58)	1.784(47)	1.06 (0.43)	0.04 (0.96)	
	0.125	1.282(13)	1.388(37)	1.361(29)	0.89 (0.41)	0.07 (0.93)	

TABLE XXI. Fine-ensemble values of the rest mass  $M_1 = E(0)$  and hyperfine splitting  $\Delta_1$ .  $am'_q = 0.0272$  and  $0.031$ . Uncertainties are the average 68% bootstrap error.

	$\kappa$	Ensemble	$aM_1$	$aM_1^*$	$a\Delta_1$
$am'_q = 0.0272$	0.090	(0.0062, 0.031)	1.7387(13)	1.7546(19)	0.0158(15)
	0.0923	(0.0031, 0.031)	1.6877(21)	1.7054(25)	0.0177(19)
	0.0923	(0.0062, 0.031)	1.6870(13)	1.7037(20)	0.0167(16)
	0.0923	(0.0124, 0.031)	1.6835(16)	1.7024(19)	0.0188(14)
	0.1256	(0.0062, 0.031)	0.8408(8)	0.8968(16)	0.0561(15)
	0.127	(0.0031, 0.031)	0.7944(9)	0.8534(19)	0.0590(19)
	0.127	(0.0062, 0.031)	0.7946(7)	0.8544(15)	0.0599(13)
	0.127	(0.0124, 0.031)	0.7901(7)	0.8514(11)	0.0613(12)
$am'_q = 0.031$	0.090	(0.0062, 0.031)	1.7441(13)	1.7601(17)	0.0159(14)
	0.0923	(0.0062, 0.031)	1.6926(12)	1.7093(18)	0.0167(14)
	0.1256	(0.0062, 0.031)	0.8470(8)	0.9030(14)	0.0560(13)
	0.127	(0.0062, 0.031)	0.8009(7)	0.8606(14)	0.0597(12)

TABLE XXII. Same as Table XXI but for the coarse ensembles with  $am'_q = 0.0415$  and 0.03.

	$\kappa$	Ensemble	$aM_1$	$aM_1^*$	$a\Delta_1$
$am'_q = 0.0415$	0.074	(0.007, 0.050)	2.2394(22)	2.2618(25)	0.0224(09)
	0.086	(0.005, 0.050)	1.9662(17)	1.9941(27)	0.0279(18)
	0.086	(0.007, 0.050)	1.9644(17)	1.9943(21)	0.0299(11)
	0.086	(0.010, 0.050)	1.9676(16)	1.9978(21)	0.0301(12)
	0.086	(0.020, 0.050)	1.9584(16)	1.9891(21)	0.0307(14)
	0.122	(0.005, 0.050)	1.0529(10)	1.1399(22)	0.0870(17)
	0.122	(0.007, 0.050)	1.0520(7)	1.1393(17)	0.0873(15)
	0.122	(0.010, 0.050)	1.0549(10)	1.1414(26)	0.0865(22)
	0.122	(0.020, 0.050)	1.0446(9)	1.1339(16)	0.0894(16)
	0.124	(0.007, 0.050)	0.9871(7)	1.0819(17)	0.0948(15)
$am'_q = 0.030$	0.074	(0.007, 0.050)	2.2241(26)	2.2466(29)	0.0225(11)
	0.086	(0.007, 0.050)	1.9488(21)	1.9787(25)	0.0299(14)
	0.122	(0.007, 0.050)	1.0339(08)	1.1220(20)	0.0881(17)

TABLE XXIII. Same as Table XXI but for medium-coarse ensembles with  $am'_q = 0.0484$  and 0.0387.

	$\kappa$	Ensemble	$aM_1$	$aM_1^*$	$a\Delta_1$
$am'_q = 0.0484$	0.076	(0.0097, 0.0484)	2.3192(27)	2.3472(36)	0.0280(17)
	0.076	(0.0194, 0.0484)	2.3153(30)	2.3424(47)	0.0270(26)
	0.076	(0.0290, 0.0484)	2.3137(23)	2.3445(21)	0.0308(16)
	0.080	(0.0097, 0.0484)	2.2298(24)	2.2606(34)	0.0308(17)
	0.122	(0.0097, 0.0484)	1.2427(8)	1.3390(21)	0.0963(19)
	0.122	(0.0194, 0.0484)	1.2397(8)	1.3400(17)	0.1004(14)
	0.122	(0.0290, 0.0484)	1.2364(7)	1.3402(17)	0.1038(14)
	0.125	(0.0097, 0.0484)	1.1565(8)	1.2634(21)	0.1069(20)
	0.076	(0.0097, 0.0484)	2.3060(31)	2.3341(40)	0.0281(19)
	0.122	(0.0097, 0.0484)	1.2271(9)	1.3237(25)	0.0966(24)

### 1. The hyperfine splitting in lattice units $a\Delta_1$

In this subsection, we tabulate values of  $a\Delta_1$  relevant to the discussion in Sec. VI B of the uncertainty in the hyperfine splitting due to statistics,  $\kappa$  tuning, and the light-valence mass.

### 2. The hyperfine splitting in physical units $r_1\Delta_1$

In this subsection, we tabulate values of  $r_1\Delta_1$  relevant to the discussion in Sec. VI B of the dependence of the hyperfine splitting on the sea-quark masses.

## APPENDIX D: PARTIALLY QUENCHED CHIRAL PERTURBATION THEORY FOR THE HEAVY-LIGHT HYPERFINE SPLITTING

For full (unquenched) QCD, Jenkins [47] has calculated the hyperfine splitting at one-loop in heavy-meson chiral perturbation theory. It is not difficult to take her result (Eq. (A.10) of Ref. [47]) and extend it to partially quenched QCD. The further step of including staggered taste-violations (*i.e.*, doing staggered chiral perturbation theory) would also be fairly straightforward, but we do not take it here because the continuum partially quenched form is

TABLE XXIV. Fine-ensemble values of the hyperfine splitting  $\Delta_1$  in units of  $r_1$ .  $am'_q = 0.0272$ . Uncertainties are the average 68% bootstrap error.

	Ensemble	$r_1\Delta_1$
$\kappa = 0.0923$	(0.0031, 0.031)	0.0653(69)
	(0.0062, 0.031)	0.0618(58)
	(0.0124, 0.031)	0.0700(52)
$\kappa = 0.127$	(0.0031, 0.031)	0.2178(69)
	(0.0062, 0.031)	0.2217(48)
	(0.0124, 0.031)	0.2281(43)

TABLE XXV. Same as Table [XXIV](#) but for the coarse ensembles with  $am'_q = 0.0415$ .

	Ensemble	$r_1 \Delta_1$
$\kappa = 0.086$	(0.005, 0.050)	0.0738(46)
	(0.007, 0.050)	0.0788(29)
	(0.010, 0.050)	0.0788(32)
	(0.020, 0.050)	0.0814(36)
$\kappa = 0.122$	(0.005, 0.050)	0.2301(44)
	(0.007, 0.050)	0.2300(39)
	(0.010, 0.050)	0.2265(58)
	(0.020, 0.050)	0.2370(43)

TABLE XXVI. Same as Table [XXIV](#) but for medium-coarse-ensembles with  $am'_q = 0.0484$ .

	Ensemble	$r_1 \Delta_1$
$\kappa = 0.076$	(0.0097, 0.0484)	0.0616(37)
	(0.0194, 0.0484)	0.0603(57)
$\kappa = 0.122$	(0.0290, 0.0484)	0.0699(37)
	(0.0097, 0.0484)	0.2117(41)
	(0.0194, 0.0484)	0.2244(30)
	(0.0290, 0.0484)	0.2357(39)

sufficient for estimating the small systematic effect due to the mistuning of sea-quark masses. Unlike Jenkins, we neglect electromagnetic and isospin-violating effects.

At the quark-flow level, the relevant diagrams are the self-energy diagrams shown in Fig. 5(a) [left] of Ref. [\[56\]](#) (the “connected diagram”) and in Figs. 5(b), (c) [left] of Ref. [\[56\]](#) (the “disconnected diagram”).<sup>6</sup> One simply needs to determine how much of Jenkins’s result comes from each of these two diagrams. This is accomplished by noting that, when the light-valence quark is a  $u$  ( $a = 1$  in Jenkins’s notation), an internal kaon only appears in the connected diagram, when the quark in the virtual loop is an  $s$ . This fixes the normalization of the connected diagram. Using the methods described in Refs. [\[56–58\]](#) (but dropping the taste violations and indeed the taste degree of freedom itself), the disconnected diagram is easily

calculated. Its normalization can then be fixed so that it supplies the remainder of the  $a = 1$  result in Ref. [\[47\]](#).

There are ample checks of this reasoning. First, the same normalizations must apply for any choice of the valence mass. The  $\eta$  contributes in each case only through the disconnected diagram, while the pion contributions come from both connected and disconnected diagrams for valence  $u$  or  $d$  ( $a = 1, 2$ ), and must be absent for valence  $s$  ( $a = 3$ ). Finally the contribution from the unphysical  $s\bar{s}$  state, which appears in each diagram for  $a = 3$ , should cancel.

It is then immediate to write down the partially quenched version. Let the light-valence quark be  $x$ , with mass  $m_x$ , and let the sea quarks be  $u, d, s$  with masses  $m_u = m_d = m_l$  and  $m_s$ . With the light-meson decay constant  $f$  normalized so that  $f \approx f_\pi \approx 130$  MeV, the hyperfine splitting  $M_x^* - M_x$  is given by

$$M_x^* - M_x = \Delta - \frac{\Delta g_\pi^2}{8\pi^2 f^2} \delta_{\log} + 2\Delta^{(\sigma)}(2m_l + m_s) + 2\Delta^{(a)}m_x, \quad (\text{D1})$$

where  $\Delta$  is the splitting in the (three-flavor) chiral limit, and  $\Delta^{(\sigma)}$  and  $\Delta^{(a)}$  are low-energy constants that start at order  $1/m_Q$  in the heavy-quark expansion. The nonanalytic chiral logarithms  $\delta_{\log}$  are

$$\delta_{\log} = \sum_{F=u,d,s} \ell(M_{xF}^2) - \frac{1}{3} R_X^{[2,2]}(\{m\}, \{\mu\}) \tilde{\ell}(M_X^2) - \frac{1}{3} \sum_{j=X,\eta} D_{j,X}^{[2,2]}(\{m\}, \{\mu\}) \ell(M_j^2). \quad (\text{D2})$$

Here  $M_X$  is the mass of the valence  $x\bar{x}$  meson, and  $M_{xF}$  is the mass of the mixed valence-sea  $x\bar{F}$  meson. The residue functions  $R_j^{[n,k]}$  and  $D_{j,i}^{[n,k]}$ , as well as the chiral logarithm functions  $\ell(m^2)$  and  $\tilde{\ell}(m^2)$ , are defined in Refs. [\[57,58\]](#). The term with the sum over  $F$  comes from the connected diagram, while those with the residue functions come from the disconnected diagram, which has a double pole at  $M_X^2$  in the partially quenched case. The denominator ( $\{m\}$ ) and numerator ( $\{\mu\}$ ) mass sets are

$$\{m\} = \{M_X, M_\eta\}, \quad \{\mu\} = \{M_U, M_S\} \quad (\text{D3})$$

with  $M_U$  and  $M_S$  the masses of the  $u\bar{u}$  and  $s\bar{s}$  mesons, respectively.

<sup>6</sup>One should ignore the solid square in each of the figures from Ref. [\[56\]](#) because it represents a current insertion, not relevant here.

- [1] C.T.H. Davies *et al.* (HPQCD, MILC, and Fermilab Lattice Collaborations), *Phys. Rev. Lett.* **92**, 022001 (2004).
- [2] C. Aubin *et al.* (HPQCD, MILC, and UKQCD Collaborations), *Phys. Rev. D* **70**, 031504(R) (2004); C. Aubin *et al.* (MILC Collaboration), *Phys. Rev. D* **70**, 114501 (2004).
- [3] C. Aubin *et al.* (Fermilab Lattice, MILC, and HPQCD Collaborations), *Phys. Rev. Lett.* **95**, 122002 (2005); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LATTICE2008* (2008) 278 [[arXiv:0904.1895](#)].
- [4] C. Aubin *et al.* (Fermilab Lattice, MILC, and HPQCD Collaborations), *Phys. Rev. Lett.* **94**, 011601 (2005); M. Okamoto *et al.* (Fermilab Lattice and MILC Collaborations), *Nucl. Phys. B, Proc. Suppl.* **140**, 461 (2005); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **79**, 014506 (2009); J.A. Bailey *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **79**, 054507 (2009); C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **80**, 034026 (2009).
- [5] R.T. Evans *et al.* (Fermilab Lattice and MILC collaborations), *Proc. Sci., LATTICE 2008* (2008) 052; R.T. Evans, E. Gámiz, A. El-Khadra, and A.S. Kronfeld (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LATTICE 2009* (2009) 245 [[arXiv:0911.5432](#)].
- [6] E.D. Freeland, A.S. Kronfeld, J.N. Simone, and R.S. Van de Water (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LAT2007* (2007) 243 [[arXiv:0710.4339](#)].
- [7] I.F. Allison *et al.* (HPQCD and Fermilab Lattice Collaborations), *Phys. Rev. Lett.* **94**, 172001 (2005).
- [8] T. Burch *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **81**, 034508 (2010).
- [9] C.W. Bernard *et al.*, *Phys. Rev. D* **64**, 054506 (2001).
- [10] C. Aubin *et al.*, *Phys. Rev. D* **70**, 094505 (2004).
- [11] A. Bazavov *et al.*, *Rev. Mod. Phys.* **82**, 1349 (2010).
- [12] C.W. Bernard *et al.*, *Phys. Rev. D* **62**, 034503 (2000); R. Sommer, *Nucl. Phys.* **B411**, 839 (1994).
- [13] T. Blum *et al.*, *Phys. Rev. D* **55**, R1133 (1997); K. Orginos and D. Toussaint (MILC Collaboration), *Phys. Rev. D* **59**, 014501 (1998); J.F. Lagaë and D.K. Sinclair, *Phys. Rev. D* **59**, 014511 (1998); G.P. Lepage, *Phys. Rev. D* **59**, 074502 (1999); K. Orginos, D. Toussaint, and R.L. Sugar (MILC Collaboration), *Phys. Rev. D* **60**, 054503 (1999); C.W. Bernard *et al.* (MILC Collaboration), *Phys. Rev. D* **61**, 111502(R) (2000).
- [14] A.X. El-Khadra, A.S. Kronfeld, and P.B. Mackenzie, *Phys. Rev. D* **55**, 3933 (1997).
- [15] B. Sheikholeslami and R. Wohlert, *Nucl. Phys.* **B259**, 572 (1985).
- [16] K.G. Wilson, in *New Phenomena in Subnuclear Physics*, edited by A. Zichichi (Plenum, New York, 1977).
- [17] A.S. Kronfeld, *Phys. Rev. D* **62**, 014505 (2000).
- [18] J. Harada, S. Hashimoto, K.I. Ishikawa, A.S. Kronfeld, T. Onogi, and N. Yamada, *Phys. Rev. D* **65**, 094513 (2002); **71**, 019903(E) (2005); J. Harada, S. Hashimoto, A.S. Kronfeld, and T. Onogi, *Phys. Rev. D* **65**, 094514 (2002).
- [19] K. Symanzik, in *Recent Developments in Gauge Theories*, edited by G. 't Hooft *et al.* (Plenum, New York, 1980); in *Mathematical Problems in Theoretical Physics*, edited by R. Schrader (Springer, New York, 1982); *Nucl. Phys.* **B226**, 187 (1983) **B226**, 205 (1983).
- [20] G.P. Lepage and P.B. Mackenzie, *Phys. Rev. D* **48**, 2250 (1993).
- [21] B.P.G. Mertens, A.S. Kronfeld, and A.X. El-Khadra, *Phys. Rev. D* **58**, 034505 (1998).
- [22] A.S. Kronfeld and J.N. Simone, *Phys. Lett. B* **490**, 228 (2000); **495**, 441(E) (2000).
- [23] M.B. Oktay and A.S. Kronfeld, *Phys. Rev. D* **78**, 014504 (2008).
- [24] M.G. Alford, W. Dimm, G.P. Lepage, G. Hockney, and P.B. Mackenzie, *Phys. Lett. B* **361**, 87 (1995).
- [25] C.W. Bernard *et al.* (MILC Collaboration), *Phys. Rev. D* **58**, 014503 (1998); M. Lüscher and P. Weisz, *Commun. Math. Phys.* **97**, 59 (1985); **98**, 433(E) (1985); *Phys. Lett. B* **158B**, 250 (1985).
- [26] Y. Shamir, *Phys. Rev. D* **71**, 034509 (2005); **75**, 054503 (2007).
- [27] C. Bernard, *Phys. Rev. D* **73**, 114503 (2006).
- [28] C. Bernard, M. Golterman, and Y. Shamir, *Phys. Rev. D* **77**, 074505 (2008).
- [29] S. Dürr, *Proc. Sci., LAT2005* (2006) 021 [[arXiv:hep-lat/0509026](#)]; S.R. Sharpe, *Proc. Sci., LAT2006* (2006) 022 [[arXiv:heplat/0610094](#)]; M. Golterman, *Proc. Sci., CONFINEMENT8* (2008) 014 [[arXiv:0812.3110](#)].
- [30] A.S. Kronfeld, *Proc. Sci., LAT2007* (2007) 016 [[arXiv:0711.0699](#)].
- [31] For  $m_s$  and  $r_1/a$ , we use the fitting methods described in Ref. [11] as applied to the data available in June, 2007 [32]. Specifically, to smooth  $r_1/a$ ,  $\ln(r_1/a)$  is fit to a polynomial in  $\beta$  and  $2am'_l + am'_s$ . Values of  $u_0$  can also be found in this reference.
- [32] C. Bernard *et al.* (MILC Collaboration), *Proc. Sci., LAT2007* (2007) 137 [[arXiv:0711.0021](#)].
- [33] A. Gray, I. Allison, C.T.H. Davies, E. Gulez, G.P. Lepage, J. Shigemitsu, and M. Wingate, *Phys. Rev. D* **72**, 094507 (2005).
- [34] C. Bernard *et al.* (MILC Collaboration), *Proc. Sci., LAT2005* (2006) 025 [[arXiv:hep-lat/0509137](#)].
- [35] C. Bernard *et al.*, *Proc. Sci., LAT2007* (2007) 090 [[arXiv:0710.1118](#)].
- [36] C.T.H. Davies, E. Follana, I.D. Kendall, G.P. Lepage, and C. McNeile, *Phys. Rev. D* **81**, 034506 (2010).
- [37] D.P. Menscher, Ph.D. thesis, University of Illinois, 2005.
- [38] J.L. Richardson, *Phys. Lett.* **82B**, 272 (1979).
- [39] M. Wingate, J. Shigemitsu, C.T.H. Davies, G.P. Lepage, and H.D. Trottier, *Phys. Rev. D* **67**, 054505 (2003).
- [40] M.F.L. Golterman, *Nucl. Phys.* **B278**, 417 (1986).
- [41] F. Gliozzi, *Nucl. Phys.* **B204**, 419 (1982).
- [42] H. Kluberg-Stern, A. Morel, O. Napoly, and B. Petersson, *Nucl. Phys.* **B220**, 447 (1983).
- [43] G.P. Lepage, B. Clark, C.T.H. Davies, K. Hornbostel, P.B. Mackenzie, C. Morningstar, and H. Trottier, *Nucl. Phys. B, Proc. Suppl.* **106**, 12 (2002); C. Morningstar, *Nucl. Phys. B, Proc. Suppl.* **109**, 185 (2002).
- [44] For a pedagogical introduction see D.S. Sivia, *Data Analysis: A Bayesian Tutorial* (Oxford University Press, USA, 1996); A review can be found in K. Nakamura *et al.* (Particle Data Group), *J. Phys. G* **37**, 075021 (2010).

- [45] W.M. Yao *et al.* (Particle Data Group), *J. Phys. G* **33**, 1 (2006) and 2007 partial update for edition 2008 (URL: <http://pdg.lbl.gov>). The 2009-10 updates for the  $D_s$  hyperfine splitting have not changed its value. For the  $B_s$  hyperfine splitting, the “average value” of 46.1(1.5) has remained consistent; the fit value has increased slightly to 49.0(1.5).
- [46] A.S. Kronfeld, *Nucl. Phys. B*, Proc. Suppl. **53**, 401 (1997).
- [47] E.E. Jenkins, *Nucl. Phys.* **B412**, 181 (1994).
- [48] C.M. Arnesen, B. Grinstein, I.Z. Rothstein, and I.W. Stewart, *Phys. Rev. Lett.* **95**, 071802 (2005).
- [49] E. Eichten and B.R. Hill, *Phys. Lett. B* **243**, 427 (1990).
- [50] M. Nobes and H. Trottier, Proc. Sci., LAT2005 (2006) 209 [[arXiv:hep-lat/0509128](https://arxiv.org/abs/hep-lat/0509128)]; M. Nobes, Ph.D. thesis, Simon Fraser University, 2004.
- [51] C.T. Sachrajda and G. Villadoro, *Phys. Lett. B* **609**, 73 (2005); P.F. Bedaque, *Phys. Lett. B* **593**, 82 (2004).
- [52] E.D. Freeland, A.S. Kronfeld, J.N. Simone, and R.S. Van de Water (for the Fermilab Lattice and MILC Collaborations), Proc. Sci., LAT2006 (2006) 083 [[arXiv:hep-lat/0610108](https://arxiv.org/abs/hep-lat/0610108)].
- [53] B. Aubert *et al.* (BABAR Collaboration), *Phys. Rev. D* **74**, 032007 (2006).
- [54] M. Di Pierro *et al.*, *Nucl. Phys. B*, Proc. Suppl. **129**, 328 (2004).
- [55] V.I. Berestetskii, E.M. Lifshitz, and L.P. Pitaevskii, *Relativistic Quantum Theory* (Pergamon, Oxford, 1971).
- [56] C. Aubin and C. Bernard, *Phys. Rev. D* **73**, 014515 (2006).
- [57] C. Aubin and C. Bernard, *Phys. Rev. D* **68**, 034014 (2003).
- [58] C. Aubin and C. Bernard, *Phys. Rev. D* **68**, 074011 (2003).

# web2py for Scientific Applications

*The needs of scientists to communicate effectively, collaborate over the Internet, and organize and present data in a structured and accessible manner have become critically important. The web2py framework offers rapid development and prototyping of secure database-driven Web applications and was created specifically with the scientific and academic communities in mind.*

**D**iscussions about scientific applications often refer to large-scale numerical computations. In recent times, things have changed, and the scientific community's needs have broadened. In many sectors, scientists' need to communicate effectively, increase access, and organize and present data are more important than their need to write fast code.

For example, using Secure Copy (SCP) and Remote Sync (RSync) protocols to transfer data from one place to another was a good solution when a handful of scientists were working on it, but it's no longer effective when hundreds are collaborating. Data must be tagged, access tracked, and errors logged. Applications that access the data must be given a face in the form of an accessible Web-based user interface or a Web service interface. Through the Web interface, users can more easily and efficiently interact with the data. Grid tools provide a solution to some of these problems, but such tools have a steep learning curve. Moreover, there's not one solution that fits all problems.

Web frameworks are a relatively new class of tools that have come to the rescue of scientists. These frameworks are libraries that allow rapid development of Web-based applications. Most of the frameworks provide APIs to handle concurrency and scalability (to serve multiple users

simultaneously), security (authentication, authorization, cross-site scripting, and injection prevention), storage (cookies, sessions, and database access via a database abstraction layer), and interface with third-party programs via standard communication protocols.

A Web framework can turn a scientific software library into a complex interactive application. Web frameworks already find wide applications in science. They've been used to build Web interfaces for data acquisition systems, computing clusters, and computer algebra systems. Different communities have developed Web frameworks in almost every programming language. Common open source ones include Struts for Java, Symphony for PHP, Ruby on Rails for Ruby, and Django for Python.

Here, I describe web2py (<http://web2py.com>),<sup>1</sup> a new Web framework created in Python and programmable in Python. The only framework born in an academic environment, web2py is used, for example, in projects supported by the US Department of Energy,<sup>2</sup> and it counts more than 1,700 registered users. Here, I'll use web2py to show how we can build a Web application that stores a database of DNA strings and looks for similarities using Needleman-Wunsh plots.<sup>3</sup> This article is based on a tutorial I gave at Supercomputing 2009 in Portland; the code presented is a complete web2py application.

## Framework Overview

Three things distinguish web2py from many existing frameworks. First, its main objective is not simply to provide a utility for Web development,

but to make Web development as easy as possible. To do this, web2py forces developers to follow good practice (such as form self-submission), relieves developers from security-related decisions, and provides everything in one package, including a Web-based integrated development environment (IDE).

Second, web2py is the only framework to provide a database abstraction layer that works on both relational databases and Google App Engine (support for other nonrelational databases is in the works). Third, web2py is written in Python, which lets users leverage power scientific libraries, including NumPy, SciPy, matplotlib, PyTable/HDF5, and others.

Because web2py is a model-view-controller framework, applications built with the framework are divided into files organized by role. The three most important roles are

- *models*, which describe how data should be represented in the system;
- *views*, which describe how data should be presented to users; and
- *controllers*, which describe the application workflow and run the core algorithms.

web2py also handles other types of files:

- regular Python extension modules;
- static files, including images and other files that don't change;
- sessions, which store each system user's current state;
- cache;
- databases, for long-term memory;
- uploads (that is, static files uploaded by visitors);
- crontab files, which describe tasks that must be executed on a fixed schedule; and
- languages, which contain the application translation to support internationalization.

Files are grouped under subfolders of the application folder. The web2py framework enforces this structure and other good software engineering practices. It also provides a Web-based IDE and Web-based testing and debugging tools.

## Example Application

As an example, let's say we need a Web application that lets users post and retrieve DNA strings, compare them using the Needleman-Wunsh algorithm,<sup>3</sup> and plot the result. Users must be authenticated to use the system. For plotting, we'll use the matplotlib library.

Figure 1. First steps in creating the example dna application. The process starts by creating an empty Web-based integrated development environment application that's prepopulated with scaffolding files that are editable.

```
db.define_table('dna',
    Field('name',unique=True,notnull=True)
    Field('sequence', 'text',
        requires=IS_MATCH('([ATGC]*)')))
```

Figure 2. A new model file models/db\_dna.py. The define\_table dynamically generates and executes the SQL code to create the dna table.

To create our dna application, we first easily create a new empty application via the Web-based IDE. This application comes prepopulated with scaffolding files that can be edited. (I'll make it clear when we're editing a scaffolding file or creating a new file.) Once web2py is installed and running, we can access the Web-based IDE at URL <http://127.0.0.1:8000/admin>. Figure 1 shows a screenshot.

## The Model

We start by creating a new model file models/db\_dna.py (see Figure 2).

The function define\_table dynamically generates and executes the SQL code—for any of the supported back-ends, including SQLite, PostgreSQL, MySQL, MSSQL, FireBird, Oracle, Informix, DB2, SyBase, and the Google App Engine—to create the dna table (or alter the table if it has different field content). The dna table has two fields, name and sequence, that have requirements that the framework will enforce at the form level:

- name must be non-empty and not already in database, and

```

if not db(db.dna.id>0).count():
    import random, uuid
    genes = []
    for k in range(10):
        bases = ['ATGC'[random.randint(0,3)] for i in range(20)]
        genes.append(''.join(bases))
    for k in range(100):
        sequence = ''.join([genes[random.randint(0,9)] for i in range(50)])
        db.dna.insert(name=uuid.uuid4(), sequence=sequence)

```

Figure 3. The code to append to the `models/db_dna.py` file. The `if` statement checks whether the table contains any record; if not, the code inserts 100 new random DNA sequences in the table.

The screenshot shows the `appadmin` interface for the `db` database. The title is "database db select". Below it is a button "[insert new dna]". A section titled "Rows in table" contains a form with fields: "Query:" (containing `db.dna.id>0`), "Update:" (empty), and "Delete:" (empty). Below the form is a note: "The "query" is a condition like "db.table1.field1=='value'". Something like "db.table1.field1=='value'" Use (...)&(...) for AND, (...)|(...) for OR, and ~(...) for NOT to build more complex queries" and "update" is an optional expression like "field1='newvalue'". You cannot update or delete". A "submit" button is present. Below the form is a table titled "100 selected" with columns `dna.id`, `dna.name`, and `dna.sequence`. The table lists 100 rows of DNA sequence data.

	<code>dna.id</code>	<code>dna.name</code>	<code>dna.sequence</code>
1	4fdbffdb-6e46...	GTTTGGACTGAAG...	
2	ddc5b894-475c...	TGTGTCCCCACG...	
3	9aed74b0-f02d...	ATACGGGGTGCA...	
4	1812fdcc-86c1...	TTCGAAGTCATGC...	
5	ca1e6049-bf1b...	ATACGGGGTGCA...	
6	2213fd1e-d502...	TTCGAAGTCATGC...	

Figure 4. The `appadmin` interface. This interface lets users browse through the generated sequences.

- `sequence` must contain only the ATGC symbols.

When users simply type this text in the model file, web2py triggers, creates the table, and generates a Web-based interface (`appadmin`) to perform database inserts, updates, deletes,

and selects. Users can access `appadmin` at `http://127.0.0.1:8000/appadmin`; Figure 2 shows a screenshot (I assume web2py is running on 127.0.0.1:8000).

Next, we need to generate dummy data to populate the database. We'll assume each DNA sequence is comprised of 50 genes, each picked at random from 10 possible variations. Each gene consists of 20 random bases (ATGC). We then name each sequence with a random universal unique identifier (uuid). Figure 3 shows the code to append to the `models/db_dna.py` file.

The `if` statement checks whether the table contains any record; if not, the code inserts 100 new random DNA sequences in the table. Ideally, real data should be used to populate the database. Users can browse generated sequences through the `appadmin` interface (see Figure 4).

### The Algorithm

Figure 5 shows the core of our application: the Needleman-Wunsh algorithm.

The Needleman-Wunsh algorithm is implemented in the `model/db_dna.py` file. The algorithm takes two strings, `a` and `b` (which in our case are two DNA sequences), and returns a table `z`,

```

def needleman_wunsch(a,b,p=0.97):
    z=[]
    for i,r in enumerate(a):
        z.append([])
        for j,c in enumerate(b):
            if r==c: z[-1].append(z[-1][-1]+1 if i*j>0 else 1)
            else: z[-1].append(p*max(z[-1][-1] if i>0 else 0,
                                      z[i][j-1] if j>0 else 0))
    return z

```

Figure 5. The Needleman-Wunsh algorithm. This algorithm forms the application's core and is implemented in the `model/db_dna.py` file.

where  $z[i][j]$  represent a similarity factor between the substrings  $a[:i]$  and  $b[:j]$ . This empirical algorithm works by defining the similarity between  $a[:i]$  and  $b[:j]$  as the similarity between  $a[:i-1]$  and  $b[:j-1]$  times  $p = 0.97$ , if  $a[i] == b[j]$  or, otherwise, as the maximum of the similarities between  $a[:i], b[:j-1]$  and  $a[:i-1], b[:j]$ . The application visualizes  $z$  so users can locate similar genes in each couple of DNA sequences (see Figure 6).

### The Controller

The controller's role is to implement the application workflow, decide which functions and algorithms to expose, and call these algorithms. In our case, we edit the scaffolding controller `controllers/default.py` and replace the `index` function with the function shown in Figure 7.

This function defines a form that contains two fields (`s1` and `s2`). Each of them is a reference to `db.dna` record. It must be represented by its `%(name)s` as opposed to its `id`. The function, which returns a form, is called by visiting `http://127.0.0.1:8000/dna/default/index` (see Figure 6). When we pick two sequences and submit the form, the form is accepted (`if form.accepts(...)`) and returns the variable `image` set to the dynamically generated URL calling another function (`needleman_wunsch_plot`) that generates the Needleman-Wunsh plot (which we've not yet implemented). The form variables `s1` and `s2` are also passed to that function via `vars=form.vars`.

The decorator `@auth.requires_login()` guarantees that only users who have logged in can visit the page. All other visitors must register and go through a login page provided automatically by the framework (web2py implements the role-based access control mechanism with multiple authentication modules).

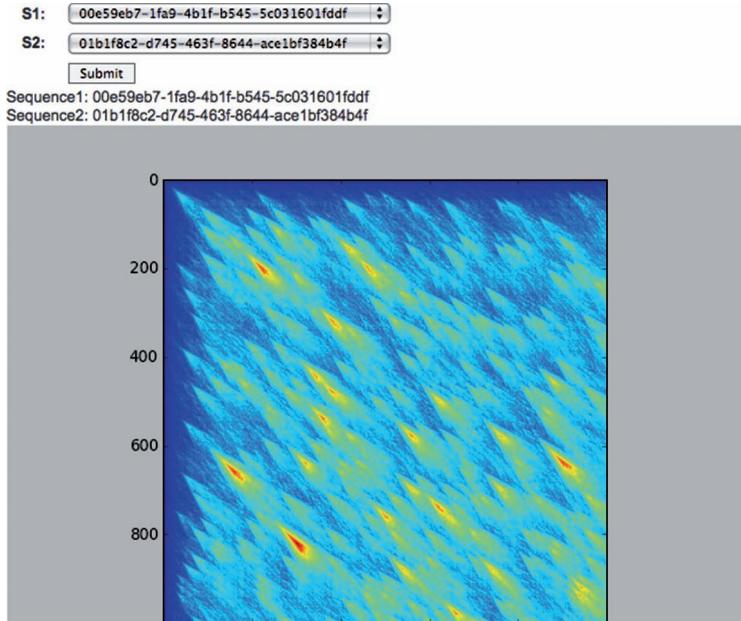


Figure 6. Visualizing  $z$ . Offering this visualization lets users locate similar genes in DNA sequences.

### The View

web2py provides a default presentation for the controller functions' output, but we can customize it in multiple ways. The easiest way is to provide a view for the `index` function by editing the file `views/default/index.html` (see Figure 8).

In this view, everything in `{{ ... }}` is pure Python code and everything outside is HTML (because this is a `.html` view). Here, we extended the default `layout.html`. This is a generic HTML file that ships with the scaffolding application. Its role is to give all pages a consistent look and feel, while also allowing those pages to be customized. We embedded the form in the page `{{=form}}` and also embed the image pointing to the image's URL `{{if image:}}...{{pass}}`.

```
@auth.requires_login()
def index():
    form = SQLFORM.factory(
        Field('s1', db.dna, requires=IS_IN_DB(db, 'dna.id', '%(name)s'),
              Field('s2', db.dna, requires=IS_IN_DB(db, 'dna.id', '%(name)s')))
    if form.accepts(request.vars):
        image = URL(r=request, f='needleman_wunsch_plot', vars=form.vars)
    else:
        image = None
    return dict(form=form, image=image)
```

Figure 7. The function that replaces the `index` function. This new function defines a form that contains the two fields that reference the `db.dna` record.

```

{{extend 'layout.html'}}
{{=form}}
{{if image:}}
Sequence1: {{=db.dna[request.vars.s1].name}}<br/>
Sequence2: {{=db.dna[request.vars.s2].name}}<br/>

{{pass}}

```

Figure 8. Customizing the controller functions' output. The easiest way to achieve this is to edit the file `views/default/index.html` to provide a view for the `index` function.

```

@auth.requires_login()
def needleman_wunsch_plot():
    import cStringIO
    from matplotlib.backends.backend_agg import *
    from matplotlib.figure import *
    ### 1) fetch sequences
    dna1 = db.dna[request.vars.s1].sequence
    dna2 = db.dna[request.vars.s2].sequence
    ### 2) run algorithm
    z = needleman_wunsch(dna1,dna2)
    ### 3) make figure and plot on canvas
    fig = Figure()
    fig.add_subplot(111).imshow(z)
    ### 4) print canvas to memory mapped file
    stream = cStringIO.StringIO()
    FigureCanvasAgg(fig).print_png(stream)
    ### 5) return content of memory mapped file
    response.headers['Content-Type']='image/png'
    return stream.getvalue()

```

Figure 9. The complete implementation. The `needleman_wunsch_plot` is implemented in `controllers/default.py` and has several tasks, including to retrieve actual sequences from the database and call the Needleman-Wunsh algorithm.

### Plotting

Finally, we implement the function `needleman_wunsch_plot` in `controllers/default.py`. This function has five primary tasks:

- retrieve the actual sequences from the database,
- call the Needleman-Wunsh algorithm,
- print the data as a plot on a canvas in memory,
- print the canvas in a memory-mapped PNG file (`cStringIO`), and
- return the PNG content.

Figure 9 shows the complete implementation.

The `matplotlib` APIs are described in detail at <http://matplotlib.sourceforge.net>. The yellow

and red patterns in Figure 6 let us easily find gene pairs (identified by their  $x$  and  $y$  coordinates) that contain similar DNA subsequences.

### Web Services

At this point, our program is complete and fully functional, so it's time to add a Web service. Specifically, we want to expose the ability to retrieve a DNA sequence from its `uuid` name via an XML Remote Procedure Calling (RPC) service. We can do this simply by placing the following code in `controllers/default.py`:

```

@Service.xmlrpc
def get_sequence(uuid):
    return db.dna(name=uuid).sequence

```

The decorator (`@service.xmlrpc`) exposes the function (the function's name, `get_sequence`, is arbitrary). The service is available at `http://127.0.0.1:8000/dna/default/call/xmlrpc` and can be called from any programming language supporting XML-RPC. Because all database access is performed via the database abstraction layer's API, there's no need to write any SQL.

### Linked Data

Linked Data (LD; <http://linkeddata.org>) is a set of best practices for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using Universal Resource Identifiers and a Resource Description Framework (RDF).

RDF is an XML-based document file format. World Wide Web inventor Tim Berners-Lee proposed the LD, which consists of tagging database tables, fields, and relations using a Web Ontology Language (OWL) and exposing the data together with metadata that describe its meaning and relations. To support LD, `web2py` lets users tag data with OWL and automatically create an RDF service. More details are available at <http://web2py.com/semantic>.

**B**y providing tools to help them expose their data and algorithms to the Web for both human and machine consumption, Web frameworks can help scientists increase transparency, awareness, and efficiency in their work. The `web2py` framework, which is released under the GPL2 license, was developed specifically with the scientific and academic communities in mind.

## Acknowledgments

I thank David Angulo for reviewing this article and for insightful conversations about the algorithm used here.

## References

1. M. Di Pierro, *web2py Manual*, 3rd ed., lulu.com, 2010.
2. M. Di Pierro, "mc4qcd," *Proc. Advanced Computing and Analysis Techniques, Proc. Science*, 2010; [http://pos.sissa.it/archive/conferences/093/054/ACAT2010\\_054.pdf](http://pos.sissa.it/archive/conferences/093/054/ACAT2010_054.pdf).
3. S.B. Needleman and C.D Wunsch, "A General Method Applicable to the Search for Similarities in

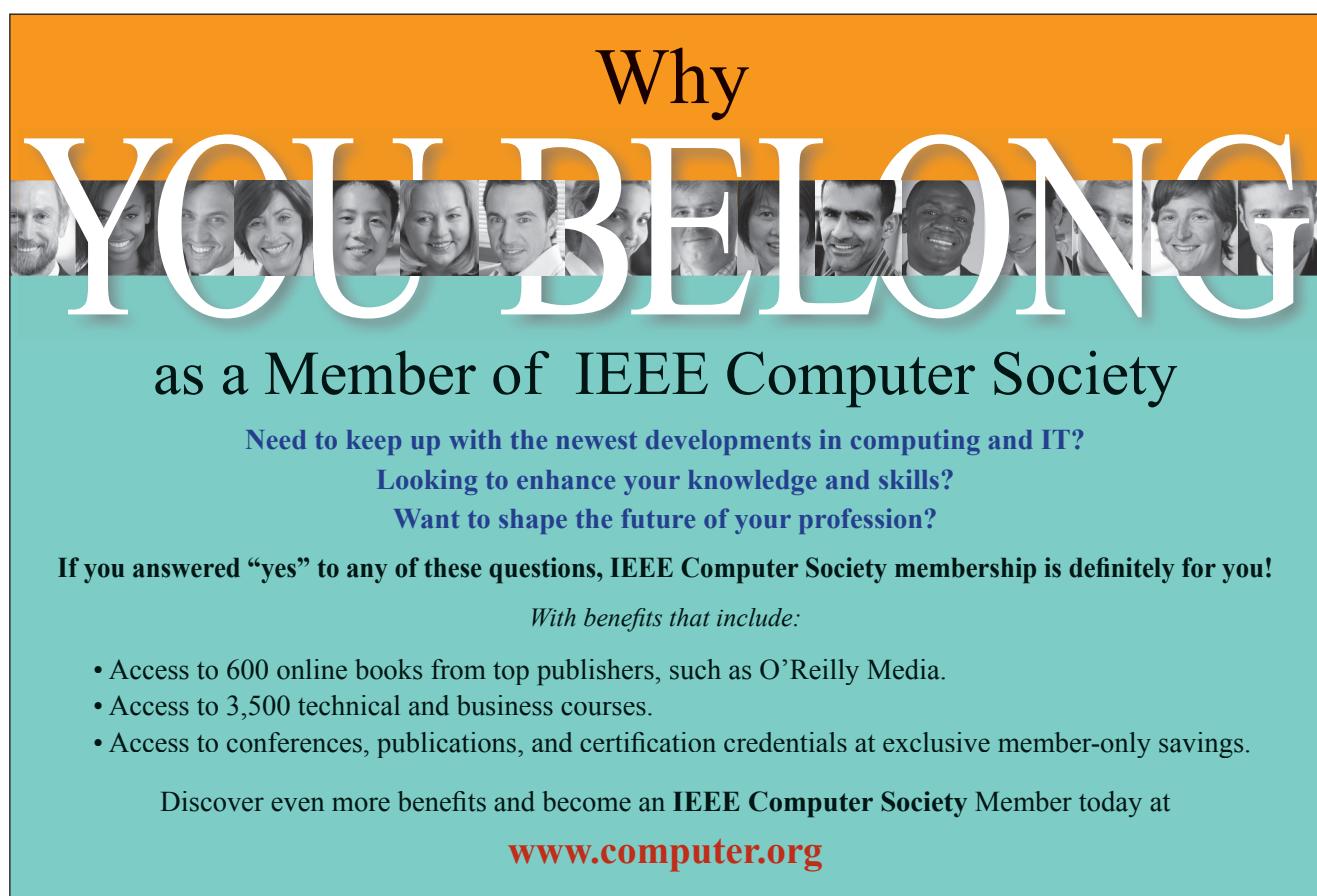
the Amino Acid Sequence of Two Proteins," *J. Molecular Biology*, vol. 48, no. 3, 1970, pp. 443–53.

**Massimo Di Pierro** is an associate professor at the School of Computing of DePaul University in Chicago. His main expertise is in high-performance numerical algorithms for scientific and financial applications. Di Pierro has a PhD in high energy physics from the University of Southampton, UK. Contact him at mdipierro@cs.depaul.edu.

**cn** Selected articles and columns from IEEE Computer Society publications are also available for free at <http://ComputingNow.computer.org>.



The cover of the IEEE Internet Computing magazine features a blue background with white and teal circular patterns. The title "Engineering and Applying the Internet" is at the top left, followed by the IEEE logo and "Internet Computing". Below the title is a large graphic of a stylized infinity symbol formed by a series of overlapping circles. To the right of the infinity symbol, the text reads: "IEEE Internet Computing reports emerging tools, technologies, and applications implemented through the Internet to support a worldwide computing environment." At the bottom right, it says: "For submission information and author guidelines, please visit [www.computer.org/internet/author.htm](http://www.computer.org/internet/author.htm)".



The advertisement features a large, bold, white "YOU BELONG" text where the letters are filled with small, diverse portraits of people's faces. Above this, the word "Why" is written in a smaller, serif font. Below the main title, the text "as a Member of IEEE Computer Society" is displayed in a large, serif font. Three bullet points below the title list reasons for joining: "Need to keep up with the newest developments in computing and IT?", "Looking to enhance your knowledge and skills?", and "Want to shape the future of your profession?". The text "If you answered ‘yes’ to any of these questions, IEEE Computer Society membership is definitely for you!" is in bold. Below that, "With benefits that include:" is followed by a bulleted list of benefits: "Access to 600 online books from top publishers, such as O'Reilly Media.", "Access to 3,500 technical and business courses.", and "Access to conferences, publications, and certification credentials at exclusive member-only savings.". At the bottom, it says "Discover even more benefits and become an IEEE Computer Society Member today at" and provides the website [www.computer.org](http://www.computer.org).

# Concurrency in Modern Programming Languages



In the last 10 years, CPU clock speeds haven't increased significantly, but the computing power of CPUs has continued to grow according to Moore's law. Researchers achieved this by increasing the number of

computing cores per chip. This process has made parallel computing relevant to a much broader community.

Writing computer programs that take advantage of multicores is a challenge because of the diversity of architectures and because of the inherent difficulty of writing parallel programs. In today's multicore world, the means to express parallelism and the effectiveness in achieving parallel performance are increasingly important.

Traditionally, there have been two approaches to parallel programming: threads that communicate via shared memory and processes that

communicate via message passing. Modern programming languages introduce different approaches, some of which are discussed in this special issue of *CiSE*.

## About the Articles

In “Clojure for Number Crunching on Multi-core Machines,” Martin Kalin and David Miller describe the Clojure language, which was first released to the public by Rich Hickey in 2007. Clojure—like Lisp, Erlang, and Haskell—is a functional language. In functional languages, the output of any function depends only on the input, and there’s no global state. This approach alone should make functional languages appealing to scientists. The input and output consist of immutable data structures that can safely be accessed concurrently without the need for locking.

If a problem is decomposable, the functions that constitute the intermediate steps to the solution can be executed in parallel, and the underlying virtual machine (VM) can automatically allocate them to different cores. Yet not everything is always so simple and some objects must be mutable. Different languages deal with mutability in different ways.

Clojure uses transactional memory. With some limitations, if two functions reference the same mutable object and try to modify it in parallel (think about incrementing a counter), the conflict is detected, and effectively the access to the object is serialized without need for explicit locking. Clojure runs on the Java VM and can access all of the Java APIs.

Steve Vinoski’s article “Concurrency and Message Passing in Erlang” describes Erlang and the Open Telecom Platform (OTP) released by Erlang in 1996. Erlang is a functional language originally developed with the goal of making software development more reliable, easier, and less expensive.

In Erlang, applications consist of many lightweight processes that are managed by a scheduler in the VM and dispatched to the available cores automatically. These lightweight processes have no overhead, and one VM can run millions of them. They can only communicate via message passing, thus avoiding the need for explicit locking.

Duncan Coutts and Andres Löh’s article “Deterministic Parallel Programming with Haskell” describes the Haskell language. Haskell is a modern functional programming language that, instead of using concurrent threads, provides a library (Repa) to allow writing concise high-level parallel programs that are guaranteed to be deterministic.

Haskell doesn’t use a VM, but it compiles to native code. Like Erlang, it supports lightweight threads, but it doesn’t use explicit message passing. Instead, it determines data dependencies from the data structures. For example, a Haskell library called Repa defines the notion of a parallel array. Parallel array operations are automatically parallelized into as many chunks as there are CPU cores.

Coutts and Löh also mention ongoing development efforts to let Haskell support grids, GPUs, and CPU single-instruction, multiple-data (SIMD) instructions. The goal is to allow different parallel programming models within a single programming language.

## A Parallel Assignment

To better compare Clojure, Erlang, and Haskell, we assigned the authors the same problem: parallelize a naive solver for a 1D Poisson equation:

$$\nabla^2 \phi(x) = \rho(x).$$

Once discretized ( $x \rightarrow x_i = x_0 + b_i$ ,  $\phi(x) \rightarrow \phi_i$ ,  $\rho(x) \rightarrow \rho_i$ ) this equation turns into an iterative expression:

$$\forall i \quad \phi_i \leftarrow \frac{\phi_{i-1} + \phi_{i+1}}{2} - \frac{b^2}{2} \rho_i,$$

where  $\rho$  and  $\phi$  can be thought of as input and output, respectively. Here,  $b$  is the discretization step. The expression must be iterated until convergence of the array  $\phi$ . Each iteration can be parallelized in the integer variable  $i$ .

There are three issues of importance here: ease of implementation of the parallel approach; parallel scaling with the number of cores; and overall language speed. We leave the discussion of the first two issues to the authors.

A truly unbiased speed comparison is impossible, and real-life performance is problem-specific. Still yet, we refer the reader to *The Computer Language Benchmarks Game* (<http://shootout.alioth.debian.org>), from which we report benchmarks (see Table 1).

The first column of Table 1 shows the average running time of a battery of benchmarks on a single core of a Q6600 Intel CPU normalized to C++ running time. This measures the language speed but not its scaling. The second column shows the results of a specific benchmark, called *thread-ring*, on four cores. Thread-ring creates 503 threads and passes a token around them in a ring, while performing no computation. This measures the language overhead in dealing with concurrency. In both cases, smaller numbers indicate better performance.

**Table 1. Computer language benchmarks.**

Language	One core (average)	Four cores (thread-ring)
C++ (GNU C++)	1.00	1.00
Java	1.49	1.86
Haskell (Glasgow Haskell Compiler)	1.58	0.05
Clojure	3.31	0.51
Erlang (High-Performance Erlang)	7.83	0.19

**T**hese languages aren't for everyone, and we don't expect the readers to immediately switch to using them. Yet their expressiveness and their ability to handle concurrency in a simpler way will be appealing to some. Moreover, they might provide an indication as to the future of programming languages.

In fact, some of the ideas presented in these articles aren't unique to functional languages. An example is the Go language (<http://golang.org>) proposed by Google as a possible C replacement. As with Erlang and Haskell, Go implements lightweight "goroutines" that communicate via buffered channels, thus providing a low-level and efficient message passing interface to concurrent tasks. Another example is the D language (<http://dlang.org>), which is a systems programming language similar to C++. D's approach to

concurrency is also to use isolated threads or processes that communicate via messages, although it also supports old-style synchronization of critical sections using explicit mutex locks.

The proliferation of new languages and paradigms can at first appear confusing, but it shows a clear trend toward increasing the expressiveness and readability of languages while decoupling the coding of the algorithm from parallelization and concurrency optimizations. 

**Massimo Di Pierro** is an associate professor at the School of Computing of Digital Media at DePaul University. His main research interests are in lattice quantum chromodynamics, quantitative risk management, and Web development. Massimo has a PhD in physics from the University of Southampton, United Kingdom. Contact him at [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu).

**David Skinner** currently leads the Scientific Discovery through Advanced Computing (SciDAC) Outreach Center at Lawrence Berkeley Laboratory, which provides information and services that support SciDAC's outreach, training, and research objectives. His research focuses on the performance analysis of computing architectures and HPC applications. Skinner has a PhD in theoretical chemistry from the University of California, Berkeley. Contact him at [deskinner@lbl.gov](mailto:deskinner@lbl.gov).



[www.aip.org](http://www.aip.org)

The American Institute of Physics (AIP) is a not-for-profit membership corporation chartered in New York State in 1931 for the purpose of promoting the advancement and diffusion of the knowledge of physics and its application to human welfare. Leading societies in the fields of physics, astronomy, and related sciences are its members.

In order to achieve its purpose, AIP serves physics and related fields of science and technology by serving its member societies, individual scientists, educators, students, R&D leaders, and the general public with programs, services, and publications—*information that matters*.

The Institute publishes its own scientific journals as well as those of its member societies; provides abstracting and indexing services; provides online database services; disseminates reliable information on physics to the public; collects and analyzes statistics on the profession and on physics education; encourages and assists in the documentation and study of the history and philosophy of physics; cooperates with other organizations on educational projects at all levels; and collects and analyzes information on federal programs and budgets.

The scientists represented by the Institute through its member societies number more than 134 000. In addition, approximately 6000 students in more than 700 colleges and universities are members of the Institute's Society of Physics Students, which includes the honor society Sigma Pi Sigma. Industry is represented through the membership of 37 Corporate Associates.

**Governing Board:** Louis J. Lanzerotti\* (chair), Richard Baccante, Barry Barish, Malcolm R. Beasley, G. Fritz Benedict, J. Daniel Bourland,\* Terri Braun, Robert Byer, Timothy A. Cohn, Beth Cunningham,\* Bruce H. Curran,\* Robert Doering, Michael D. Duncan,\* H. Frederick Dylla\*(ex officio), David Ernst, Janet Fender, Judith Flippin-Anderson,\* Brian J. Fraser,\* Jaime Fucugauchi, A. Jeffrey Giacomini,\* Mark Hamilton, John Haynes, Paul L. Kelley, Angela R. Keyser, James T. Kirby Jr, Kate Kirby, Rudolf Ludeke,\* Jim Marshall, Kevin B. Marvel,\* Christine McEntee, Catherine O'Riordan, Elizabeth A. Rogan, Charles E. Schmid,\* Joseph Serene,\* Benjamin B. Shavely\* (ex officio), David Sokoloff, Scott Sommerfeldt, Gene Sprouse, Gay Stewart, Hervey (Peter) Stockman, Michael Turner.

\*Executive Committee member.

**Management Committee:** H. Frederick Dylla, Executive Director and CEO; Richard Baccante, Treasurer and CFO; Theresa C. Braun, Vice President, Human Resources; John S. Haynes, Vice President, Publishing; Catherine O'Riordan, Vice President, Physics Resources; Benjamin B. Shavely, Secretary.

# Neutral $B$ -meson mixing from three-flavor lattice quantum chromodynamics: Determination of the $SU(3)$ -breaking ratio $\xi$

A. Bazavov,<sup>1</sup> C. Bernard,<sup>2</sup> C. M. Bouchard,<sup>3,4,5</sup> C. DeTar,<sup>6</sup> M. Di Pierro,<sup>7</sup> A. X. El-Khadra,<sup>3</sup> R. T. Evans,<sup>3,8</sup> E. D. Freeland,<sup>3,9</sup> E. Gámiz,<sup>4,10,\*</sup> Steven Gottlieb,<sup>11</sup> U. M. Heller,<sup>12</sup> J. E. Hetrick,<sup>13</sup> R. Jain,<sup>3</sup> A. S. Kronfeld,<sup>4</sup> J. Laiho,<sup>14</sup> L. Levkova,<sup>6</sup> P. B. Mackenzie,<sup>4</sup> E. T. Neil,<sup>4</sup> M. B. Oktay,<sup>6</sup> J. N. Simone,<sup>4</sup> R. Sugar,<sup>15</sup> D. Toussaint,<sup>16</sup> and R. S. Van de Water<sup>1</sup>

(Fermilab Lattice and MILC Collaborations)

<sup>1</sup>*Physics Department, Brookhaven National Laboratory, Upton, New York 11973, USA*

<sup>2</sup>*Department of Physics, Washington University, St. Louis, Missouri 63130, USA*

<sup>3</sup>*Physics Department, University of Illinois, Urbana, Illinois 61801, USA*

<sup>4</sup>*Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA*

<sup>5</sup>*Department of Physics, The Ohio State University, Columbus, Ohio 43210, USA*

<sup>6</sup>*Physics Department, University of Utah, Salt Lake City, Utah 84112, USA*

<sup>7</sup>*School of Computing, DePaul University, Chicago, Illinois 60604, USA*

<sup>8</sup>*Department of Nuclear Engineering, North Carolina State University, Raleigh, North Carolina 27695, USA*

<sup>9</sup>*Department of Physics, Benedictine University, Lisle, Illinois 60532, USA*

<sup>10</sup>*CAFPE and Departamento de Física Teórica y del Cosmos, Universidad de Granada, Granada, Spain*

<sup>11</sup>*Department of Physics, Indiana University, Bloomington, Indiana 47405, USA*

<sup>12</sup>*American Physical Society, Ridge, New York 11961, USA*

<sup>13</sup>*Physics Department, University of the Pacific, Stockton, California 95211, USA*

<sup>14</sup>*SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow, Scotland, United Kingdom*

<sup>15</sup>*Department of Physics, University of California, Santa Barbara, California 93106, USA*

<sup>16</sup>*Department of Physics, University of Arizona, Tucson, Arizona 85721, USA*

(Received 1 June 2012; published 14 August 2012)

We study  $SU(3)$ -breaking effects in the neutral  $B_d$ - $\bar{B}_d$  and  $B_s$ - $\bar{B}_s$  systems with unquenched  $N_f = 2 + 1$  lattice quantum chromodynamics (QCD). We calculate the relevant matrix elements on the MILC collaboration's gauge configurations with asqtad-improved staggered sea quarks. For the valence light-quarks ( $u$ ,  $d$ , and  $s$ ) we use the asqtad action, while for  $b$  quarks we use the Fermilab action. We obtain  $\xi = f_{B_s} \sqrt{B_{B_s}} / f_{B_d} \sqrt{B_{B_d}} = 1.268 \pm 0.063$ . We also present results for the ratio of bag parameters  $B_{B_s}/B_{B_d}$  and the ratio of Cabibbo-Kobayashi-Maskawa matrix elements  $|V_{td}|/|V_{ts}|$ . Although we focus on the calculation of  $\xi$ , the strategy and techniques described here will be employed in future extended studies of the  $B$  mixing parameters  $\Delta M_{d,s}$  and  $\Delta \Gamma_{d,s}$  in the standard model and beyond.

DOI: 10.1103/PhysRevD.86.034503

PACS numbers: 12.38.Gc, 13.20.He

## I. INTRODUCTION

The observation of new particles at high-energy colliders is not the only way for new physics to be discovered. It can also be unveiled through the observation of deviations from the standard model (SM) via high-precision measurements of low-energy observables in high-luminosity experiments. This requires matching precision in the theoretical SM predictions for these observables. In principle, such a comparison could reveal the exchange of virtual, new heavy particles involving scales much higher than those that can be achieved in direct production at high-energy colliders.

Heavy-flavor physics and, in particular, neutral-meson mixing are potentially very sensitive to these virtual

effects. Neutral-meson mixing occurs at loop level in the SM, see Fig. 1, and it is further suppressed by small Cabibbo-Kobayashi-Maskawa (CKM) matrix elements, so the effect of new particles in the internal loops could be noticeable in the parameters describing the mixing. Indeed, there are several measurements for which there is a  $2 - 3\sigma$  difference from the SM prediction. These include  $\sin(2\beta)$  [1], the like-sign dimuon charge asymmetry [2], and unitarity triangle (UT) fits [3–7]. It has been argued that these differences may be due to physics beyond the standard model (BSM) affecting the neutral  $B$ -meson mixing processes [3,4].

In the  $B_s^0$  system, the relative phase between the decay amplitudes with and without mixing,  $\beta_s$ , could also show BSM effects, as pointed out in Ref. [8] and later hinted at in a Tevatron measurement [9]. Although new measurements at the CDF [10] and D0 detector [11] are in better

\*megamiz@ugr.es

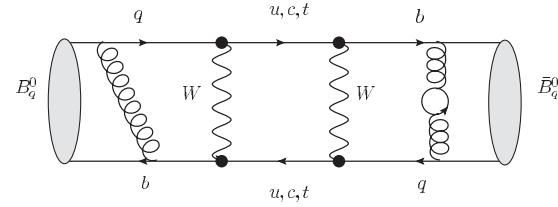
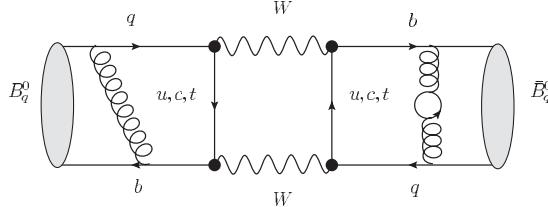


FIG. 1. Box diagrams contributing to  $B^0 - \bar{B}^0$  mixing in the SM. Gluon exchanges shown in the plot are just representative of the QCD corrections.

agreement with the SM, reducing the difference from  $\sim 3\sigma$  to  $\sim 1\sigma$ , there is still room for a large deviation of  $\beta_s$  from SM values.

The main parameters describing mixing in the  $B_s^0$  and the  $B_d^0$  systems are the mass differences,  $\Delta M_{s(d)}$ , and the decay width differences,  $\Delta\Gamma_{s(d)}$ , between the heavy and light  $B_{s(d)}^0$  mass eigenstates, and the  $CP$  violating phases  $\phi_{s(d)}$ . The phases  $\phi_{s(d)}$  are defined as the argument of the ratio of the dispersive and absorptive off-diagonal elements of the time evolution matrix which describes the mixing [12]. The existence of new, heavy particles in loops could affect the value of the mass differences, given by the dispersive part of the time evolution matrix. The mass differences  $\Delta M_s$  [13–15] and  $\Delta M_d$  [16] have been measured with an accuracy better than 1%. Improving the theoretical control on these quantities is thus crucial in order to fully exploit the potential of  $CP$  violating observables to search for nonstandard physics. In addition, the theoretical calculation of BSM contributions to mixing and the experimental measurement of  $B^0$  mixing parameters can help in constraining BSM parameters and understanding new physics [6]. Several recent studies have addressed that task [3–7, 17–24], finding that one of the main limitations to further constraining the parameter space in BSM theories is the error associated with the theoretical calculation of the nonperturbative inputs.

The most interesting quantity to analyze in  $B^0$  mixing phenomena is the  $SU(3)$ -breaking ratio  $\xi$ , which measures the difference between the mixing parameters in the  $B_s^0$  and the  $B_d^0$  systems, and enters the relation between the ratio of mass differences and CKM matrix elements as

$$\left| \frac{V_{td}}{V_{ts}} \right| = \xi \sqrt{\frac{\Delta M_d M_{B_s}}{\Delta M_s M_{B_d}}} \quad (1.1)$$

Its value, together with the experimental measurement of the mass differences  $\Delta M_{s,d}$ , determines the ratio of CKM matrix elements  $|V_{td}/V_{ts}|$ , which constrains one side of the unitarity triangle [25, 26]. Thus,  $\xi$  is one of the key ingredients in UT analyses [3, 5–7].

In the SM, mixing is due to box diagrams with the exchange of two  $W$ -bosons, like those in Fig. 1. These box diagrams can be rewritten in terms of an effective Hamiltonian with four-fermion operators describing processes with  $\Delta B = 2$ . In BSM theories, mixing processes

can receive contributions from additional diagrams due to the exchange of new, heavy particles. These can also be parametrized in terms of four-fermion effective operators built with SM degrees of freedom. The most general effective Hamiltonian describing processes with  $\Delta B = 2$  was given in Refs. [27, 28], and can also be found in Ref. [29]. There are a total of five independent operators (plus parity conjugates) in the Hamiltonian, but only three of them contribute to mixing in the SM

$$\mathcal{H}_{\text{eff,SM}}^{\Delta B=2} = \sum_{i=1}^3 C_i \mathcal{O}_i, \quad (1.2)$$

with

$$\begin{aligned} \mathcal{O}_1^q &= (\bar{q}^i \gamma^\nu L b^i)(\bar{q}^j \gamma^\nu L b^j), \\ \mathcal{O}_2^q &= (\bar{q}^i L b^i)(\bar{q}^j L b^j), \\ \mathcal{O}_3^q &= (\bar{q}^i L b^j)(\bar{q}^j L b^i), \end{aligned} \quad (1.3)$$

where  $i$  and  $j$  are color indices, and  $L$  and  $R$  are the Dirac projection operators  $\frac{1}{2}(1 - \gamma_5)$  and  $\frac{1}{2}(1 + \gamma_5)$ , respectively. The fields  $q$  denote strange or down fields for  $B_s^0$  and  $B_d^0$  mixing, respectively, and  $b$  represents the bottom field.

The matrix element of the first operator in Eq. (1.3),  $\mathcal{O}_1^q$ , provides the mass difference in the SM:

$$\Delta M_q^{\text{SM}} = \frac{G_F^2 M_W^2}{6\pi^2} |V_{tq}^* V_{tb}|^2 \eta_2^B S_0(x_t) M_{B_q} f_{B_q}^2 \hat{B}_{B_q}, \quad (1.4)$$

where  $S_0(x_t)$  is the Inami-Lim function [30], which depends on the top quark mass through  $x_t = m_t^2/M_W^2$ , and the quantity  $\eta_2^B$  is a perturbative quantum chromodynamics correction factor. The products  $f_{B_q}^2 \hat{B}_{B_q}$  parametrize the hadronic matrix elements in the effective theory by

$$\langle \bar{B}_q^0 | \mathcal{O}_1^q | B_q^0 \rangle(\mu) = \frac{2}{3} M_{B_q}^2 f_{B_q}^2 B_{B_q}(\mu). \quad (1.5)$$

The factors  $f_{B_q}$  are the  $B_q^0$  decay constants. The renormalization group invariant bag parameters  $\hat{B}_{B_q}$  in Eq. (1.4) are related to the scheme and scale dependent bag parameters in Eq. (1.5) at next-to-leading order (NLO) by

$$\hat{B}_{B_q} = [\alpha_s(\mu)]^{-6/23} \left[ 1 + \frac{\alpha_s(\mu)}{4\pi} J_5 \right] B_{B_q}(\mu), \quad (1.6)$$

where  $J_5$  is known in both  $\overline{\text{MS}}$ -NDR (naive dimensional regularization) and  $\overline{\text{MS}}\text{-HV}$  ('t Hooft-Veltman) schemes [31]. Bag parameters have traditionally been used to measure the deviation of the four-fermion operator matrix elements from their vacuum insertion values,  $B_B = 1$ .

The  $SU(3)$ -breaking parameter  $\xi$  can be written in terms of decay constants and bag parameters

$$\xi = \frac{f_{B_s} \sqrt{B_{B_s}}}{f_{B_d} \sqrt{B_{B_d}}}. \quad (1.7)$$

Many of the uncertainties that affect the theoretical calculation of the decay constants and bag parameters cancel totally or partially in this ratio, leaving the chiral extrapolation as the dominant error. Hence,  $\xi$  and the combination of CKM matrix elements related to it, can be determined with a significantly smaller error than the individual matrix elements.

The hadronic matrix elements in Eq. (1.5) encode the nonperturbative physics of the problem and are best calculated using lattice QCD. Our current knowledge of them, limits the accuracy with which the CKM matrix elements appearing in Eq. (1.4) can be determined from the experimental measurements of  $\Delta M_{s(d)}$ . In particular, the uncertainty associated with the calculation of  $\xi$  is one of the main limiting factors in UT analyses, so improvement in the knowledge of  $\xi$  is crucial to disentangle the origin of the  $2 - 3\sigma$  tension.

There are two  $2 + 1$  unquenched lattice calculations of the ratio  $\xi$  in the literature. One is by the HPQCD collaboration [32], which quotes the value  $\xi = 1.258(33)$ . The other is an exploratory study by the RBC and UKQCD collaborations [33] on a single lattice spacing and using the static limit for the bottom quark; their result is  $\xi = 1.13(12)$ . In this paper, we report a lattice calculation of  $\xi$  at the few-percent level.

Preliminary results related to the work here were presented in Refs. [34–37]. In Ref. [34], the simulation and correlator fitting methods were described using data for one lattice spacing, while Refs. [35,36] focused on the discussion of statistical and fitting errors, and the chiral extrapolation method. In Ref. [37], we studied the matching method and the heavy-quark discretization errors.

The primary difference between this work and the HPQCD calculation in Ref. [32] is the treatment of the valence  $b$  quarks. The HPQCD collaboration uses lattice NRQCD [38] while we employ the clover action [39] with the Fermilab interpretation [40]. An advantage of the Fermilab method is that it can also be efficiently used to simulate charm quarks, so the analysis performed in this work can be easily extended to the study of the short-distance contributions to  $D^0\bar{D}^0$  mixing. Although in the case of neutral  $D$  mixing the long-distance contributions are believed to be dominant, a calculation of the short-distance contributions can, nevertheless, provide valuable constraints on extensions of the SM [41].

In order to achieve the few-percent level of precision required by phenomenology, we use lattice QCD simulations with realistic sea quarks. In particular, we employ a subset of the magnetic induced laser cooling (MILC) configurations with  $2 + 1$  flavors of asqtad sea quarks [42–44]. In the valence sector, we use the same staggered asqtad action to simulate the light quarks. The configurations we use in this analysis were generated using the fourth-root procedure for eliminating extra degrees of freedom originating from fermion doubling. Despite the nonlocal violations of unitarity of the rooted theory at non-zero lattice spacing [45,46], there are strong theoretical arguments [47–50], as well as other analytical and numerical evidence [51–54], that the local, unitary theory of QCD is recovered in the continuum limit. This gives us confidence that the rooting procedure yields valid results. We also explicitly tested the rooting procedure as well as improvements in our heavy action by calculating the spin-dependent hyperfine splittings for  $B_s$  and  $D_s$  mesons in Ref. [55].

Our collaboration has already successfully used the asqtad MILC ensembles in similar calculations of other quantities involving  $B$  mesons, as part of a broad program of calculating matrix elements: for example, the extraction of the CKM matrix elements  $|V_{ub}|$  and  $|V_{cb}|$  from the calculation of, respectively, the semileptonic form factors describing the processes  $B \rightarrow \pi l \nu$  [56] and  $B \rightarrow D^* l \nu$  [57,58]; or, more recently, the calculation of the  $f_B$  and  $f_{B_s}$  decay constants [59] and the form factor ratios between the semileptonic decays  $\bar{B} \rightarrow D^+ l^- \bar{\nu}$  and  $\bar{B}_s \rightarrow D_s^+ l^- \bar{\nu}$  [60].

This paper is organized as follows. In Sec. II, we describe the actions and parameters used in our numerical simulations, as well as the construction of the mixing operators and correlation functions. Section III presents the renormalization method using one-loop mean field improved lattice perturbation theory. We include a discussion of the errors associated with the matching and numerical values of the matching coefficients used. Next, in Sec. IV, we give the details of the procedure for the correlator fits. Section V is devoted to the chiral-continuum extrapolation, which is performed within the framework of rooted staggered chiral perturbation theory [61–65]. We describe and discuss the choice of the functional form used in the extrapolation, the different fitting methods tested, and the choice of parameters and parametrization. In Sec. VI, we list and estimate the different systematic errors. Finally, Sec. VII compiles our final results for the parameter  $\xi$  as well as for  $|V_{td}|/|V_{ts}|$ , and the ratio of bag parameters  $B_{B_s}/B_{B_d}$ . We also discuss planned future improvements in the calculation of  $B^0$  mixing parameters by our collaboration. In Appendix A, we provide the explicit formulas for the chiral fit functions used in the chiral fits described in Sec. V. In Appendix B, we compile the functions needed to estimate the heavy-quark discretization errors in our calculation. Finally, Appendix C

TABLE I. Parameters of the ensembles analyzed in this work. The first two rows show the approximate lattice spacing and the volume.  $am_l$  and  $am_h$  are the light and strange sea quark masses, respectively.  $N_{\text{confs}}$  is the number of configurations analyzed from each ensemble, and  $am_q$  are the light valence quark masses. The  $r_1/a$  values are obtained by fitting the calculated  $r_1/a$  to a smooth function [72], as explained in Ref. [66].

$\approx a$ (fm)	$(\frac{L}{a})^3 \times \frac{T}{a}$	$am_l/am_h$	$N_{\text{confs}}$	$am_q$	$r_1/a$
0.12	$24^3 \times 64$	0.005/0.05	529	0.005, 0.007, 0.01, 0.02, 0.03, 0.0415	2.64
0.12	$20^3 \times 64$	0.007/0.05	833	0.005, 0.007, 0.01, 0.02, 0.03, 0.0415	2.63
0.12	$20^3 \times 64$	0.01/0.05	592	0.005, 0.007, 0.01, 0.02, 0.03, 0.0415	2.62
0.12	$20^3 \times 64$	0.02/0.05	460	0.005, 0.007, 0.01, 0.02, 0.03, 0.0415	2.65
0.09	$28^3 \times 96$	0.0062/0.031	557	0.0031, 0.0044, 0.062, 0.0124, 0.0272, 0.031	3.70
0.09	$28^3 \times 96$	0.0124/0.031	534	0.0031, 0.0042, 0.062, 0.0124, 0.0272, 0.031	3.72

discusses our choices for prior central values and widths for the correlator fits.

## II. NUMERICAL SIMULATIONS

### A. Parameters of the simulations

The  $n_f = 2 + 1$  MILC ensembles [66] used in our calculation include the effect of three sea-quark flavors: two degenerate light quarks corresponding to the up and down quarks (although with larger masses than the physical ones), and one heavier quark corresponding to the strange quark. These dynamical quarks are simulated using the asqtad improved staggered action with errors starting at  $O(\alpha_s a^2)$  [67]. The gluon action is a Symanzik improved and tadpole improved action, with  $O(\alpha_s a^2)$  errors coming from the gluon loops removed [68,69]. The couplings needed to remove the  $O(\alpha_s a^2)$  errors coming from quark loops [70] were available only after the generation of configurations was well advanced, so these effects are not accounted for in the MILC ensembles. Thus, the dominant errors in the gauge action are also of  $O(a^4, \alpha_s a^2)$ .

The valence light-quark propagators are generated using the asqtad action and converted to naive quark propagators using the relation [71]

$$S_{\text{naive}}(x, y) = \Omega(x)\Omega^\dagger(y)S_{\text{staggered}}(x, y), \quad (2.1)$$

where  $\Omega(x) = \gamma_0^{x_0} \gamma_1^{x_1} \gamma_2^{x_2} \gamma_3^{x_3}$ .

For the heavy bottom quarks we use the Sheikholeslami-Wohlert action [39] with the Fermilab interpretation for heavy-quark systems [40]. This interpretation retains the full mass dependence of the theory within the parameters of the lattice action. A tree-level matching to QCD is then performed via heavy quark effective theory (HQET), after which it can be shown that the errors in the action begin at  $O(\alpha_s \Lambda_{\text{QCD}} a, \Lambda_{\text{QCD}}^2 a^2)$  times bounded functions of  $m_b a$ , the  $b$ -quark mass in lattice units.

We perform our analysis at two different values of the lattice spacing,  $a \approx 0.12, 0.09$  fm, and for a variety of sea-quark masses. The values used are shown in Table I. The mass of the heavy  $b$  quark is fixed to its physical value by computing the spin-averaged  $B_s$  kinetic mass [55]. This determines the  $b$  quark's hopping parameters,  $\kappa_b = 0.0860$

for the  $a \approx 0.12$  fm lattice and  $\kappa_b = 0.0923$  for the  $a \approx 0.09$  fm lattice [55], and thus the bare  $b$  quark mass. We simulate the  $B$  mesons with the six different values of light-valence quark mass listed in Table I, the smallest of which is around  $m_s/8$ , in order to facilitate the extrapolation/interpolation to the physical down/strange quark masses.

### B. Correlators: The open-meson propagator

As described in the introduction, the study of the  $SU(3)$ -breaking ratio  $\xi$  requires the calculation of the hadronic matrix elements<sup>1</sup>  $\langle \mathcal{O}_1^q \rangle$  and  $\langle \mathcal{O}_2^q \rangle$ , the latter of which mixes with  $\langle \mathcal{O}_1^q \rangle$  under renormalization, for both  $q = d, s$ . The matrix elements are obtained from three-point correlation functions with zero spatial momentum

$$C_{\mathcal{O}_i^q}(t_x, t_y) = \sum_{x,y} \langle \bar{B}_q^0(t_y, \mathbf{y}) \mathcal{O}_i^q(0) B_q^0(t_x, \mathbf{x})^\dagger \rangle, \quad (2.2)$$

where  $i = 1$  or  $2$ , and the  $B$ -meson creation operator  $B_q^0(t, \mathbf{x})^\dagger = \sum_{\mathbf{x}'} \bar{b}(t, \mathbf{x}') S(\mathbf{x}, \mathbf{x}') \gamma_5 q(t, \mathbf{x})$ , with  $q(t, \mathbf{x})$  the naive light quark field, whose propagator is constructed from the staggered propagator *via* Eq. (2.1), and with  $S(\mathbf{x}, \mathbf{x}')$  a smearing function. Our choice of smearing function is discussed in Sec. . The structure of the functions in Eq. (2.2) is depicted in Fig. 2. The four-fermion operators  $\mathcal{O}_i^q$  are placed at the origin while  $B$ -mesons are positioned at  $x$  and  $y$ . This layout allows us to perform the three-point function fits over both  $t_x$  and  $t_y$ , maximizing the information included in the fits. In order to extract the relevant matrix elements from Eq. (2.2), we need to determine the overlap of the  $B$ -meson creation operator with the ground state. Therefore, we also need the pseudoscalar two-point correlator with zero spatial momentum

$$C_{\text{PS}}(t) = \sum_x \langle B_q^0(t, \mathbf{x}) B_q^0(0, 0)^\dagger \rangle. \quad (2.3)$$

The calculation of both three-point and two-point correlators can be organized into convenient structures. Starting with a general correlator with Dirac structure  $\Gamma_1 \times \Gamma_2$ ,

<sup>1</sup>To simplify the notation, we define  $\langle \mathcal{O}_i^q \rangle \equiv \langle \bar{B}_q^0 | \mathcal{O}_i^q | B_q^0 \rangle$  for  $i = 1, 2$ .

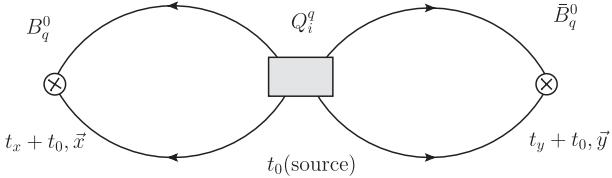


FIG. 2. Structure of the three-point correlators. A  $B_q^0$  is created at rest at  $t_x + t_0 < t_0$ . At time  $t_0$ , it oscillates into a  $\bar{B}_q^0$  via the operator  $O_i^q$ , which is subsequently annihilated at  $t_y + t_0 > t_0$ .

which accommodates a full set of  $\Delta B = 2$  four-quark operators, including those in Eq. (1.3),

$$C_3(t_x, t_y) = \sum_{x,y} \langle \bar{B}_q^0(t_y, y) \bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0) B_q^0(t_x, x)^\dagger \rangle, \quad (2.4)$$

and performing the four possible Wick contractions, we obtain

$$\begin{aligned} C_3(t_x, t_y) &= \sum_{x,y} \{ \text{tr}[\gamma_5 L_q(x, 0) \Gamma_1 H_b(0, x)] \text{tr}[\gamma_5 L_q(y, 0) \Gamma_2 H_b(0, y)] \\ &\quad + \text{tr}[\gamma_5 L_q(y, 0) \Gamma_1 H_b(0, y)] \text{tr}[\gamma_5 L_q(x, 0) \Gamma_2 H_b(0, x)] \\ &\quad - \text{tr}[\gamma_5 L_q(x, 0) \Gamma_1 H_b(0, y) \gamma_5 L_q(y, 0) \Gamma_2 H_b(0, x)] \\ &\quad - \text{tr}[\gamma_5 L_q(x, 0) \Gamma_2 H_b(0, y) \gamma_5 L_q(y, 0) \Gamma_1 H_b(0, x)] \} \end{aligned} \quad (2.5)$$

$$\begin{aligned} &= \sum_{x,y} \{ \text{tr}[L_q(x, 0) \Gamma_1 \gamma_5 H_b^\dagger(x, 0)] \text{tr}[L_q(y, 0) \Gamma_2 \gamma_5 H_b^\dagger(y, 0)] \\ &\quad + \text{tr}[L_q(y, 0) \Gamma_1 \gamma_5 H_b^\dagger(y, 0)] \text{tr}[L_q(x, 0) \Gamma_2 \gamma_5 H_b^\dagger(x, 0)] \end{aligned} \quad (2.6)$$

$$\begin{aligned} &\quad - \text{tr}[L_q(x, 0) \Gamma_1 \gamma_5 H_b^\dagger(y, 0) L_q(y, 0) \Gamma_2 \gamma_5 H_b^\dagger(x, 0)] \\ &\quad - \text{tr}[L_q(x, 0) \Gamma_2 \gamma_5 H_b^\dagger(y, 0) L_q(y, 0) \Gamma_1 \gamma_5 H_b^\dagger(x, 0)] \}, \end{aligned} \quad (2.7)$$

where  $L_q$  is the (naive) light-quark propagator, and  $H_b$  is the heavy-quark propagator. The traces in Eq. (2.5) run over spin and color indices.

These correlators can be rewritten as

$$\begin{aligned} C_3(t_x, t_y) &= \Gamma_1^{\beta\alpha} E_{aa}^{\alpha\beta}(t_x) \Gamma_2^{\tau\sigma} E_{cc}^{\sigma\tau}(t_y) \\ &\quad + \Gamma_1^{\beta\alpha} E_{aa}^{\alpha\beta}(t_y) \Gamma_2^{\tau\sigma} E_{cc}^{\sigma\tau}(t_x) \\ &\quad - \Gamma_1^{\beta\alpha} E_{ac}^{\alpha\sigma}(t_x) \Gamma_2^{\sigma\tau} E_{ca}^{\tau\beta}(t_y) \\ &\quad - \Gamma_1^{\beta\alpha} E_{ac}^{\alpha\sigma}(t_y) \Gamma_2^{\sigma\tau} E_{ca}^{\tau\beta}(t_x), \end{aligned} \quad (2.8)$$

where summation over repeated indices is implied, and we have introduced the basic objects

$$E_{ac}^{\alpha\beta}(t) = \gamma_5^{\alpha\sigma} H_{b,da}^{*\tau\sigma}(t, 0) L_{q,dc}^{\tau\beta}(t, 0), \quad (2.9)$$

with Dirac indices labeled as  $\alpha, \beta, \sigma, \tau$  and color indices labeled as  $a, c, d$ . We call the combination of propagators  $E_{ad}^{\alpha\beta}(t_x)$  defined in Eq. (2.9) “open-meson propagator”. Once the open-meson propagators have been computed and saved, all correlation functions needed for  $B$ -meson mixing, including BSM operators, can be immediately constructed by contracting them with the appropriate Dirac structures. As shown in Eq. (2.8), the three-point correlators are obtained by combining two open-meson propagators, while for the two-point correlators we only need one open-meson propagator.

### C. Doubler modes' effect on the correlation functions

The remnant doubling degeneracy of staggered fermions leads to contributions of scalar states, in addition to pseudoscalar states, in correlation functions with external pseudoscalar particles. The scalar contamination yields oscillating terms in the correlation functions [71]. In this section, we extend the analysis of Ref. [71] for two-point correlation functions, to the three-point functions introduced in Sec. II B. We conclude that the effect of the doubler modes on the three-point functions can be removed at leading order in the lattice spacing through appropriate fits of the Euclidean time-dependence.

The *doubling* symmetry of the original naive action under the transformation

$$\psi(x) \rightarrow e^{ix \cdot \pi_g} M_g \psi(x), \quad \bar{\psi}(x) \rightarrow e^{ix \cdot \pi_g} \bar{\psi}(x) M_g^\dagger, \quad (2.10)$$

where

$$M_g = \prod_{\mu \in g} i \gamma_5 \gamma_\mu, \quad (2.11)$$

$$G = \{g : g = (\mu_1, \mu_2, \dots), \mu_1 < \mu_2 < \dots\}, \quad (2.12)$$

$$(\pi_g)_\mu = \begin{cases} \frac{\pi}{a} & \text{if } \mu \in g \\ 0 & \text{otherwise} \end{cases} \quad (2.13)$$

generates sixteen equivalent species of quarks, referred to as *tastes*, that can be reduced to four by staggering the quark field [73]. Each element of  $G$  is a list of up to four indices, e.g., (2), (0,3), and (0,1,2,3) are elements of  $G$ , as is the empty set  $\emptyset$ . Different  $g$ 's label different doubler modes, or tastes.

Consider the general three-point function in momentum space,

$$\begin{aligned}
C_{\Gamma_1 \times \Gamma_2}(t_x, t_y) &\equiv \sum_{x,y} \langle \bar{b}(x) \gamma_5 q(x) [\bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0)] \bar{b}(y) \gamma_5 q(y) \rangle \\
&= \int_{-\pi/a}^{\pi/a} \frac{d^3 p}{(2\pi)^3} \frac{d^3 k}{(2\pi)^3} \langle \bar{b}(\mathbf{p}, t_x) \gamma_5 \bar{q}(\mathbf{p}, t_x) \\
&\quad \times [\bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0)] \bar{b}(\mathbf{k}, t_y) \gamma_5 \bar{q}(\mathbf{k}, t_y) \rangle, \quad (2.14)
\end{aligned}$$

where  $\Gamma_1 \times \Gamma_2$  denotes the Dirac structure of the four-fermion operators in Eq. (1.3). For simplicity of notation, we omit the smearing function from the  $B$ -meson operator and write it as  $\bar{b}(x) \gamma_5 q(x)$ . It would be straightforward (but not particularly instructive) to generalize the expressions of Eqs. (2.14), (2.15), (2.16), (2.17), (2.18), (2.19), and (2.20) to include the smearing function.

For now the bracketed four-quark operator is left in position space and  $\bar{b}$ ,  $\bar{q}$  are the spatial momentum-space bottom and strange/down fermion fields. Because of the

doubling symmetry, we can integrate over the central half of the Brillouin zone and sum over the spatial doublers

$$\begin{aligned}
C_{\Gamma_1 \times \Gamma_2}(t_x, t_y) &= \sum_{g_s, g'_s} \int_{-\pi/2a}^{\pi/2a} \frac{d^3 p}{(2\pi)^3} \frac{d^3 k}{(2\pi)^3} \\
&\quad \times \langle \bar{b}(\mathbf{p} + \boldsymbol{\pi}_{g_s}, t_x) \gamma_5 \bar{q}(\mathbf{p} + \boldsymbol{\pi}_{g_s}, t_x) \\
&\quad \times [\bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0)] \\
&\quad \times \bar{b}(\mathbf{k} + \boldsymbol{\pi}_{g'_s}, t_y) \gamma_5 \bar{q}(\mathbf{k} + \boldsymbol{\pi}_{g'_s}, t_y) \rangle, \quad (2.15)
\end{aligned}$$

where  $g_s$  denotes a particular spatial doubler mode. Due to the high momentum that is imparted to the heavy quark when  $g_s \neq \emptyset$ , such states are far off shell and have a negligible effect on the correlation function. The taste of the temporal modes can now be considered by Fourier transforming the light quarks' temporal component, and then again restricting the Brillouin zone and summing over the doublers

$$\begin{aligned}
C_{\Gamma_1 \times \Gamma_2}(t_x, t_y) &= \int_{-\pi/2a}^{\pi/2a} \frac{d^3 p}{(2\pi)^3} \frac{d^3 k}{(2\pi)^3} \int_{-\pi/2a}^{\pi/2a} \frac{dp_0}{(2\pi)} \frac{dk_0}{(2\pi)} e^{ip_0 t_x + ik_0 t_y} \langle \bar{b}(\mathbf{p}, t_x) \gamma_5 [\bar{q}'(\mathbf{p}, p_0) + (-1)^{t_x} \bar{q}'(\mathbf{p}, p_0 + \pi/a)] \\
&\quad \times [\bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0)] \bar{b}(\mathbf{k}, t_y) \gamma_5 [\bar{q}'(\mathbf{k}, k_0) + (-1)^{t_y} \bar{q}'(\mathbf{k}, k_0 + \pi/a)] \rangle. \quad (2.16)
\end{aligned}$$

With the momentum space spinors  $\tilde{f}'^s$  defined as

$$\tilde{f}'^s(p) = \prod_{\mu \in g} i \gamma_5 \gamma_\mu \bar{q}'(p + \boldsymbol{\pi}_g), \quad (2.17)$$

so that

$$\bar{q}'(\mathbf{p}, p_0) = \tilde{f}'(\mathbf{p}, p_0), \quad \bar{q}'(\mathbf{p}, p_0 + \pi/a) = i \gamma_5 \gamma_0 \tilde{f}'^0(\mathbf{p}, p_0), \quad (2.18)$$

the three-point function can be written as

$$\begin{aligned}
C_{\Gamma_1 \times \Gamma_2}(t_x, t_y) &= \int_{-\pi/2a}^{\pi/2a} \frac{d^3 p}{(2\pi)^3} \frac{d^3 k}{(2\pi)^3} \langle \bar{b}(\mathbf{p}, t_x) \gamma_5 [\tilde{f}(\mathbf{p}, t_x) + (-1)^{t_x} i \gamma_5 \gamma_0 \tilde{f}^0(\mathbf{p}, t_x)] [\bar{q}(0) \Gamma_1 b(0) \bar{q}(0) \Gamma_2 b(0)] \\
&\quad \times \bar{b}(\mathbf{k}, t_y) \gamma_5 [\tilde{f}(\mathbf{k}, t_y) + (-1)^{t_y} i \gamma_5 \gamma_0 \tilde{f}^0(\mathbf{k}, t_y)] \rangle, \quad (2.19)
\end{aligned}$$

where the superscript 0 indicates a temporal taste and no superscript is the null taste at the center of the Brillouin zone.

After Fourier transforming, the bracketed four-quark operator has no restrictions on the tastes that contribute to it. However, it must be contracted with the external quark fields to form the propagators. Because the asqtad action is used, contractions between tastes of different types are suppressed to  $O(a^2 \alpha_s^2)$ . The three-point function then takes the form

$$\begin{aligned}
C_{\Gamma_1 \times \Gamma_2}(t_x, t_y) &= \int_{-\pi/2a}^{\pi/2a} \frac{d^3 p}{(2\pi)^3} \frac{d^3 k}{(2\pi)^3} \langle \bar{b}(\mathbf{p}, t_x) \gamma_5 [f(\mathbf{p}, t_x) + (-1)^{t_x} i \gamma_5 \gamma_0 f^0(\mathbf{p}, t_x)] \\
&\quad \times [(\bar{f}(0) + i \gamma_5 \gamma_0 \bar{f}^0(0)) \Gamma_1 b(0) (\bar{f}(0) + i \gamma_5 \gamma_0 \bar{f}^0(0)) \Gamma_2 b(0)] \bar{b}(\mathbf{k}, t_y) \gamma_5 [f(\mathbf{k}, t_y) + (-1)^{t_y} i \gamma_5 \gamma_0 f^0(\mathbf{k}, t_y)] \rangle, \quad (2.20)
\end{aligned}$$

where higher order terms coming from contractions between quarks of different taste give terms of  $O(a^2 \alpha_s^2)$  that are not considered here. The effects of such terms are comparable to NLO terms in staggered chiral perturbation theory and need to be considered at that order. They give rise to the “wrong spin” terms discussed below. According to Eq. (2.20), the leading-order correlation functions have

contributions from both the pseudoscalar and the scalar states. The latter ones are known as oscillating states, since the sign of their contribution oscillates with time.

The fit ansatz for our correlators must model both regular and oscillating contributions, so that we can remove the latter and extract the physical matrix elements. This is done using the form

$$C_{\mathcal{O}_i^q}(t_x, t_y) = \sum_{\alpha, \beta=0}^{N_{\text{states}}-1} Z_\alpha Z_\beta \frac{O_{\alpha\beta}^i}{\sqrt{2E_\alpha 2E_\beta}} (-1)^{(t_x+1)\alpha + (t_y+1)\beta} e^{-E_\alpha t_x - E_\beta t_y}, \quad (2.21)$$

where the sum is over a finite number of states  $N_{\text{states}}$ . The time  $t_x$  in Eq. (2.21) and in the discussion on fitting in Sec. IV is the number of time slices between the initial state and the operator, and thus it is a positive number, unlike the time  $t_x$  defined in Fig. 2. The oscillations in Euclidean time given by the factor  $(-1)^{(t_x+1)\alpha + (t_y+1)\beta}$  reflect the contribution from the scalar states in Eq. (2.20). The matrix elements of interest are given by the three-point amplitude of the ground state  $\alpha = \beta = 0$ ,  $O_{00}^i$ . Analogously, we incorporate regular and oscillating contributions to the description of the two-point correlators by using the following functional form in the fits

$$C_{\text{PS}}(t) = \sum_{\alpha=0}^{N_{\text{states}}-1} |Z_\alpha|^2 (-1)^{(t+1)\alpha} (e^{-E_\alpha t} + e^{-E_\alpha(T-t)}), \quad (2.22)$$

where  $T$  is the temporal size of the lattice. Three- and two-point functions are fit simultaneously in our analysis, as described in Sec. IV.

## D. Improving the heavy-light four-quark operator

In addition to the discretization errors in the heavy-quark action, the mixing operator also has discretization errors due to the difference in the small-momentum behavior of lattice and continuum heavy quarks. In this section,

---


$$\langle q(p'_q), b(p'_b) | \bar{q}\Gamma_1 b \bar{q}\Gamma_2 b | q(p_q), b(p_b) \rangle_{\text{lat}} = N_q(p'_q) N_q(p_q) N_b(p'_b) N_b(p_b) \bar{u}(p'_q) \Gamma_1 u_h^{\text{lat}}(p'_b) \bar{u}(p_q) \Gamma_2 u_h^{\text{lat}}(p_b) + (\text{additional contractions}), \quad (2.25)$$

where  $N_q(p)$  and  $N_b(p)$  are normalization factors for the  $q$  and  $b$  one-particle states. Following the Fermilab interpretation in Ref. [40], we demand that lattice and continuum amplitudes match through  $O(\mathbf{ap})$ ,

$$\begin{aligned} & \mathcal{Z} \left( \bar{u}(\chi, \mathbf{p}'_q) \Gamma_1 e^{-am_1^b/2} \left[ 1 - \frac{i\gamma \cdot \mathbf{p}'_b a}{2 \sinh m_1^b} \right] u(\chi, 0) \bar{u}(\chi, \mathbf{p}_q) \Gamma_2 e^{-am_1^b/2} \left[ 1 - \frac{i\gamma \cdot \mathbf{p}_b a}{2 \sinh m_1^b} \right] u(\chi, 0) \right) + a \mathcal{Z} D_n Q_n \\ &= \bar{u}(\chi, \mathbf{p}'_q) \Gamma_1 \left[ 1 - \frac{i\gamma \cdot \mathbf{p}'_b}{2m_b} \right] u(\chi, 0) \times \bar{u}(\chi, \mathbf{p}_q) \Gamma_2 \left[ 1 - \frac{i\gamma \cdot \mathbf{p}_b}{2m_b} \right] u(\chi, 0) + O((\mathbf{pa})^2), \end{aligned} \quad (2.26)$$


---

where  $Q_n$  are dimension-seven lattice operators and  $D_n$  their corresponding coefficients. These operators and coefficients are straightforward to identify ( $m_b$  must be identified with the Fermilab kinetic mass,  $M_2$  [74]). There are two dimension-seven operators contributing to the matching,  $Q_1 = \bar{q}\Gamma_1 \gamma \cdot \mathbf{D} b \bar{q}\Gamma_2 b$  and  $Q_2 = \bar{q}\Gamma_1 b \bar{q}\Gamma_2 \gamma \cdot \mathbf{D} b$ , which will remove the  $\mathbf{pa}$  discrepancy for appropriate values of the Wilson coefficients and the normalization constant. From the relation above, we find

we describe how the lowest order of operator discretization errors are removed in our calculation. We first show that the errors start at  $O(\mathbf{ap})$  and then discuss how the error at this order can be removed by a “rotation” of the heavy-quark field.

To begin, consider the small  $\mathbf{ap}$  expansion of the spinor for the Wilson-like fermion

$$\begin{aligned} u^{\text{lat}}(\chi, \mathbf{p}) &= \frac{\gamma_0 \text{sign} \chi \sinh Ea - i\gamma_j \sin(p_j a) + L}{\sqrt{2L(L + \sinh Ea)}} u(\chi, 0) \\ &= e^{-m_1 a/2} \left[ 1 - \frac{i\gamma \cdot \mathbf{p} a}{2 \sinh m_1 a} + O((\mathbf{ap})^2) \right] u(\chi, 0), \end{aligned} \quad (2.23)$$

where  $\chi$  labels spin and particle vs antiparticle,  $\hat{p} = (2/a) \sin(pa/2)$ , and for the clover action  $L = 1 + m_0 a + \frac{1}{2} \hat{p}^2 a^2 - \cos p_0 a$ . The continuum spinor has the expansion

$$u^{\text{cont}}(\chi, \mathbf{p}) = \left[ 1 - \frac{i\gamma \cdot \mathbf{p}}{2m} + O((\mathbf{ap})^2) \right] u(\chi, 0). \quad (2.24)$$

The mismatch between the small-momentum terms can be easily removed by “rotating” the lattice heavy quark as was done to heavy-light bilinear operators in Ref. [40]. The light lattice spinors for the staggered formulation have the same small-momentum behavior as in the continuum up to  $O((\mathbf{pa})^2, (m_q a)^2)$  and need not be matched.

The analysis of Ref. [40] can be generalized from bilinears to four-fermion operators with Dirac structure  $\Gamma_1 \times \Gamma_2$ . We can take one of the terms in the contraction of the lattice operator

$$D_1 = D_2 = \left[ \frac{1}{\sinh m_1^b} - \frac{1}{2am_2^b} \right] \quad (2.27)$$

and

$$\mathcal{Z} = e^{am_1^b}. \quad (2.28)$$

Here,  $m_1$  and  $m_2$  are again the Fermilab rest and kinetic masses defined in Ref. [40].

However, through  $O(\mathbf{ap})$ , adding these operators has the same effect as inducing a “rotation” of the heavy field

$$b^r(x) = [1 + ad_1 \gamma \cdot \mathbf{D}]b(x), \quad (2.29)$$

where

$$d_1 = D_1 = D_2. \quad (2.30)$$

This removes  $O(\Lambda_{\text{QCD}}a)$  discretization errors in the operator. The leading errors are then  $O((\Lambda_{\text{QCD}}a)^2)$  and  $O(\alpha_s \Lambda_{\text{QCD}}a)$ . In the way we have set up the calculation, the open-meson propagators, Eq. (2.9), include the rotation.

### III. MATCHING OF THE LATTICE MATRIX ELEMENTS

In order to cancel the scheme and scale dependence of the Wilson coefficients in the effective Hamiltonian, we must relate the bare hadronic matrix elements of the lattice operators in Eq. (1.3) to a continuum scheme. We perform that renormalization and matching perturbatively at one-loop. In the lattice part of this renormalization calculation we use mean field improved lattice perturbation theory [75] to improve the convergence of the theory by resumming the tadpole contributions.

Already at one-loop, even in the continuum, the operators in Eq. (1.3) mix with each other under renormalization. To extract the renormalized value of  $\langle \mathcal{O}_1^q \rangle$ , we use the following matching relation

$$\begin{aligned} \langle \mathcal{O}_1^q \rangle^{\text{renor}}(\mu) &= \mathcal{C} \{ [1 + \alpha_s \cdot \zeta_{11}(\mu, m_b, am_b)] \langle \mathcal{O}_1^q \rangle^{\text{lat}} \\ &\quad + \alpha_s \cdot \zeta_{12}(\mu, m_b, am_b) \langle \mathcal{O}_2^q \rangle^{\text{lat}} \} \\ &\quad + O(\alpha_s^2, \alpha_s \Lambda_{\text{QCD}}a), \end{aligned} \quad (3.1)$$

where the renormalization coefficients  $\zeta_{ij}$  are the difference between the renormalizations in the continuum and on the lattice,  $\zeta_{ij} = Z_{ij}^{\text{cont}} - Z_{ij}^{\text{lat}}$ . The continuum renormalization scale at which we perform the matching is  $\mu$ . The lattice spacing is  $a$  and  $\mathcal{C}$  is a factor which absorbs the lattice field normalization conventions. The values of  $Z_{ij}^{\text{cont}}$  are listed in Ref. [76] and a detailed description of the calculation of the lattice renormalization coefficients will be given in Ref. [77]. Table II lists the tadpole-improved renormalization coefficients relevant for the lattice data analyzed in this paper. For each lattice spacing and  $b$  quark mass we show the infrared (IR) finite part of the  $Z_{ij}^{\text{lat}}$ 's as

TABLE II. Values of the finite part of the lattice one-loop renormalization coefficients  $Z_{ij}^{\text{lat}}$ , the difference of the continuum and lattice one-loop coefficients  $\zeta_{ij}$  needed in Eq. (3.1) for the 0.12 fm and 0.09 fm lattices, and the coupling  $\alpha_s$  used in the matching relation. The continuum ( $\overline{\text{MS}}$ -NDR) scale used in the matching is  $\mu = m_b$ .

$\approx a$ (fm)	$am_b$	$Z_{11}^{\text{lat, finite}}$	$Z_{12}^{\text{lat, finite}}$	$\zeta_{11}^{\overline{\text{MS}}-\text{NDR}}$	$\zeta_{12}^{\overline{\text{MS}}-\text{NDR}}$	$\alpha_s$
0.12	2.1881	-0.726	-0.325	0.1998	-0.312	0.32
0.09	1.7728	-0.945	-0.369	0.3041	-0.268	0.26

well as the corresponding  $\zeta_{ij}$  (in the  $\overline{\text{MS}}$ -NDR continuum scheme). The  $\zeta_{ij}$  are IR finite since the IR divergent contributions to  $Z_{ij}^{\text{lat}}$  and  $Z_{ij}^{\text{cont}}$  cancel in the difference. All the coefficients  $Z_{ij}^{\text{lat}}$  in Table II are between 0.3 and 1, which indicates a sensible behavior of the lattice perturbation series.

In order to apply the matching relation Eq. (3.1), we need to choose both a scale  $\mu$  and a value for the strong coupling constant  $\alpha_s$ . For the scale, we use the bottom quark mass and, in that way, eliminate higher-order logarithmic contributions that come in powers of  $\log(\mu/m_b)$ . For the strong coupling constant, we use the renormalized coupling in the  $V$ -scheme [69] evaluated at a scale  $q^*$ , as in Ref. [78]. The scale  $q^*$  should be the size of a typical gluon loop momentum in this process and can be calculated using the methods outlined in Refs. [69, 79]. Here, we use  $q^* = 2/a$  which is close to the calculated value for heavy-light currents using the same actions we are employing [69, 80]. This is justified since the contributions coming from the current renormalization are larger than the intrinsic four-quark contributions [77]. The values of  $\alpha_V$  are determined from the static-quark potential in a manner similar to that described in Ref. [78] and are also given in Table II.

### IV. FITTING METHOD AND STATISTICAL ERRORS

The correlation functions are calculated at four different time sources  $t$ , and then averaged over time sources. For the  $a \approx 0.12$  fm ensembles,  $t_0 = 0, 16, 32, 48$ , and for the  $a \approx 0.09$  fm ensembles,  $t_0 = 0, 24, 48, 72$ . The statistical errors in the data and fits decrease with each additional time source by approximately what is expected, suggesting that the correlators from different time sources are weakly correlated and statistical power is gained by averaging.

In order to extract the renormalized matrix elements, we tried two methods for the correlator fits. In the first method, we fit the bare correlators and combine the results afterwards with the matching coefficients in Sec. III to get the renormalized matrix elements. In the second method, we first apply the matching coefficients to the correlators for each configuration and then perform the fits to obtain the renormalized matrix elements. The central values are nearly identical with both methods, but the errors are slightly better and the fits more stable with the latter, so for the rest of this article we discuss only the results obtained with the second method.

#### A. Description of the fitting method and stability tests

The heavy quark in the two-point and three-point correlation functions is always rotated at the source as explained in Sec. II D. For three-point functions, we smear the heavy quarks at the sink using a function based on the quarkonium 1S wave function [81, 82]. For two-point functions, at the sink we either rotate local heavy quarks, or we smear

them with a 1S wave function. Smearing greatly improves the overlap with the ground state. The additional rotation at the sink is to ensure that the local-local meson correlator is positive-definite. The naive light-quark propagator is always local at source and sink.

The two-point and three-point correlators used to determine the matrix element  $\langle \mathcal{O}_1^q \rangle$  on a particular ensemble and for a particular choice of valence-mass  $m_q$  are fit simultaneously using the Bayesian fitting approach described in Refs. [83,84]. For the matrix elements on the coarse ensembles, we find the smallest errors and greatest stability using three correlators (two two-point correlation functions and one three-point correlation function):

- (i)  $C_{PS}$  in Eq. (2.22) with local source and local sink.
- (ii)  $C_{PS}$  with local source and 1S smeared sink.
- (iii)  $C_{\mathcal{O}_1^q}$  in Eq. (2.21) with local source and 1S smeared sink.

For the fine ensembles, the best results are obtained with only one two-point function and one three-point function:

- (i)  $C_{PS}$  with 1S smeared source and 1S smeared sink.
- (ii)  $C_{\mathcal{O}_1^q}$  with local source and 1S smeared sink.

The prior central values function as the initial starting guesses for our fits. Hence, we choose ground-state values guided by our data to help the fits converge. The prior central values for the ground-state masses are obtained from effective mass plots. For the overlap factors  $Z_0^d$  and  $Z_0^{1S}$ , where superscripts  $d$  and 1S denote factors corresponding to local or 1S smeared sources/sinks, we examine the amplitude of the  $B$ -meson propagator with the exponential of the ground state removed. We do the same for  $O_{00}$ , where the  $Z_0^{1S}$  amplitudes are accounted for. The prior widths are taken to be large compared with the statistical error of the parameters as reported by the fitter to avoid influencing our fit results by our choice of priors. For the higher states' overlap factors, the prior width is chosen based on the expectation that the overlaps should not be larger than the corresponding ground state ones. The energy differences have prior central values and widths that allow them to vary from  $\Delta E_{i+1,i} \equiv a(E_{i+1} - E_i) \approx 0.14\text{--}0.37$ , where experimental values [16] have been used as a guide. We checked that the prior widths for all fitting parameters are large enough so they do not influence the central value of relevant quantities extracted from the fits.

The same priors are used for all ensembles, except for the masses of the regular and oscillating ground states,  $E_0$  and  $E'_0$ , respectively. These parameters are strongly determined by the data, and very different at each lattice spacing, therefore the prior choice must also be lattice-spacing dependent. Appendix C contains a list of the prior central values and widths we use in the calculation.

Statistical errors are estimated with the bootstrap method. Specifically, for each ensemble and valence mass, 500 bootstrap ensembles are constructed from the

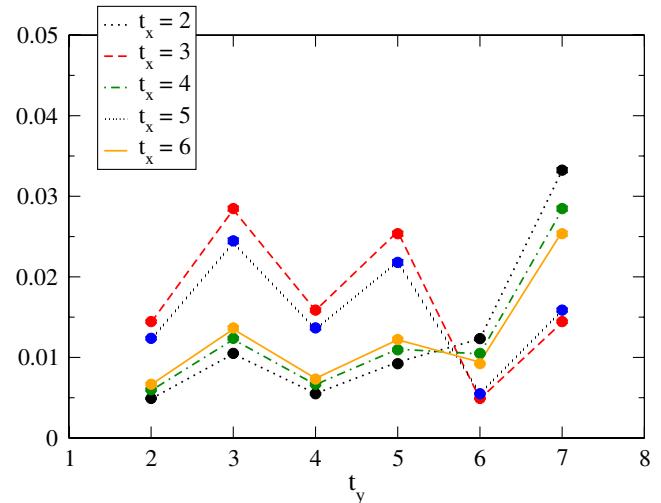


FIG. 3 (color online). Comparison of a fit with correlator data for the correlation function  $C_{\mathcal{O}_1^q}(t_x, t_y)$  at fixed  $t_x = 2\text{--}6$  (labeled by line type) and as a function of  $t_y$ , for the  $a \approx 0.09$  fm ensemble with quark masses 0.0124/0.031 and valence quark mass  $am_q = 0.031$ . The lines connect the fit function results for integer values of  $t_x$ ,  $t_y$  coming from the same single fit and evaluated at specific  $t_x$ , and the dots are the average over simulation data. Statistical errors on the plot symbols are smaller than the plot symbols. The fit results describe very well the oscillation in time shown by the data.

original ensemble by sampling with replacement. A fit is then performed to each ensemble. We find that, as long as the bootstrap ensembles are larger than  $\sim 100$ , the estimated error is independent of bootstrap ensemble size. For fitting methodology checks and plotting purposes in Figs. 3–6, statistical errors in the parameters are estimated by the average 68% bootstrap error, which is defined as half of the distance between the two points at which 16% of the distribution has a higher (lower) value.

Autocorrelations necessarily exist between correlation functions calculated on different configurations within an ensemble and can be minimized by binning the data. The autocorrelations are observable only in a few ensembles and valence masses. In many ensembles and mass combinations, the noise is large enough that the autocorrelations are not observable. We choose a conservative bin size of 4.

The number of states included in the sum in Eq. (2.21) and the time ranges we use in the fits are shown in Table III. The minimum time slice is fixed to be the same for all the correlators in the fit, three-point as well as two-point. However, the maximum time is fixed separately for the two- and three-point functions. Following Ref. [83], the number of states are determined by first performing the fit using 1 + 1 states (1 regular state + 1 oscillatory state) starting at large time slices, where the higher-energy states no longer contribute significantly and a good  $\chi^2$  per degree of freedom (d.o.f.) is obtained ( $\approx 1$ ). The fit is then performed using one lower time slice as the starting time

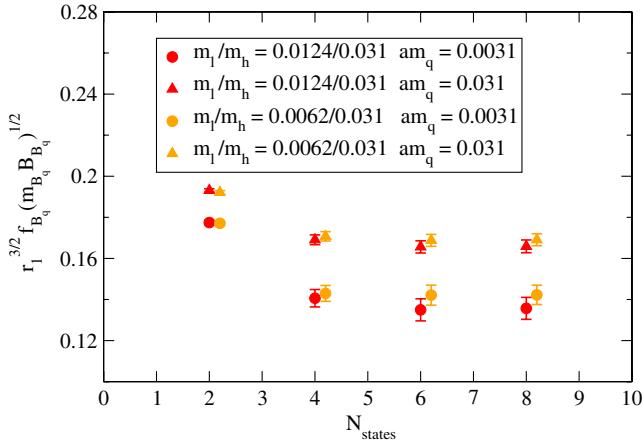


FIG. 4 (color online).  $r_1^{3/2} f_{B_q} \sqrt{M_{B_q} B_{B_q}}$  for the two  $a \approx 0.09$  fm ensembles for  $am_q = 0.031, 0.0031$  as a function of the number of states  $N_{\text{states}}$ . The fit results for  $f_{B_q} \sqrt{M_{B_q} B_{B_q}}$  reach a plateau for  $N_{\text{states}} \geq 4$ .

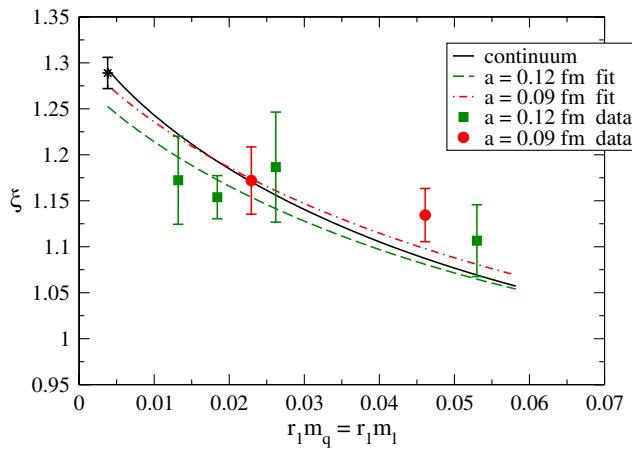


FIG. 5 (color online). Fits results using the NLO rHMS $\chi$ PT, first line in Eq. (5.6). The (black) star is the physical value of  $\xi$  in both panels. The top panel shows only the full QCD data, while the bottom one shows all the data included in the fits. The (green) squares and (red) circles and lines in the upper panel represent the 0.12 fm and 0.09 fm data and fit results, respectively. In the bottom panel, each color (symbol) labels a different ensemble.

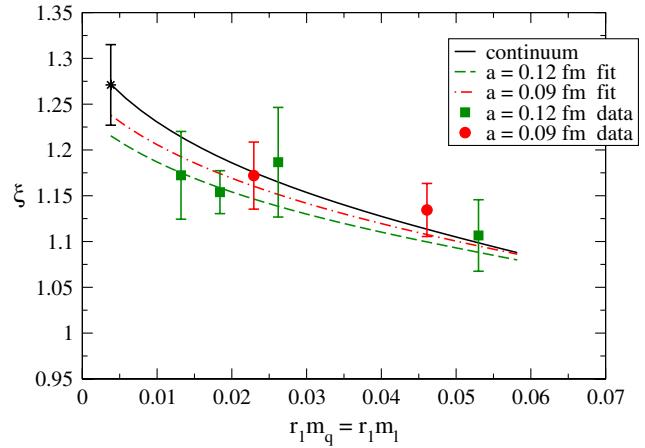


FIG. 6 (color online). Fits results using the NNLO rHMS $\chi$ PT in Eq. (5.6). The (black) star is the physical value of  $\xi$  in both panels. The top panel shows only the full QCD data, while the bottom one shows all the data included in the fits. The (green) squares and (red) circles and lines in the upper panel represent the 0.12 fm and 0.09 fm data and fit results, respectively. In the bottom panel, each color (symbol) labels a different ensemble.

$t_{\min}$ , and this is repeated, reducing  $t_{\min}$  until the  $\chi^2/\text{d.o.f.}$  is no longer reasonable,  $\geq 1.5$ . Then an additional pair of states is added to the model function, and the process iterated. Once the time slice  $t = 2$  can be included, that number of states is used in our central-value fits.<sup>2</sup>

For the three-point function, we fit using  $N_{\text{states}} = N_{\text{regular}} + N_{\text{oscillating}} = 2 + 2 = 4$  and time slices  $t_x, t_y \in [2, 10]$  for all ensembles. The two-point functions are fit for  $t \in [2, 20]$  using  $3 + 3$  states. The output of these fits successfully describe the oscillations in the correlation functions as can be seen in Fig. 3, which shows the typical behavior in one of the ensembles analyzed.

<sup>2</sup>Time slices  $t = 0$  and  $t = 1$  contain unconstrained contamination from higher-energy states. At  $t = 0$  all states contribute because they have the same exponential weighting, and  $t = 1$  is contaminated by higher-energy states because the degrees of freedom of staggered fermions spread over two time slices.

TABLE III. Number of states and time ranges used for each correlator in the fits for both the  $a \approx 0.12$  fm and  $a \approx 0.09$  fm ensembles. For the number of states, the first value indicates the number of regular states and the second one, the number of oscillating states. The labels between parentheses in the first column indicate the type of source/sink in that correlator.

Correlator	Number of states	Time range
$C_{PS}$ (local/local)	3 + 3	2–20
$C_{PS}$ (local/1S)	3 + 3	2–20
$C_{Q_q^i}$ (local/1S)	2 + 2	(2 – 10) × (2 – 10)

In order to check that this number of states is sufficient, we add more states and examine the stability of the fits. Stability plots over numbers of states for the  $a \approx 0.09$  fm ensembles, which illustrate the typical behavior of our fits, are shown in Fig. 4. The stability of central values and errors is very good for  $N_{\text{states}} \geq 4$  in all cases.

## V. CHIRAL PERTURBATION THEORY

The light sea- and valence-quark masses that are used in our lattice simulations have unphysically large values, with our lightest pion mass  $\approx 240$  MeV. To obtain information about the quark-mass dependence of the relevant matrix elements, which allows us to extrapolate our results to the physical masses, we perform our calculation at six sea  $\times$  six valence quark masses, thus including numerous partially quenched data points. In addition, the leading-order taste violations, which arise at  $O(a^2 \alpha_s)$ , are included in the theory and then removed when the extrapolation is performed using rooted heavy-meson staggered chiral perturbation theory (rHMS $\chi$ PT) [61,62,65]. The  $\chi$ PT expression for  $\langle \mathcal{O}_1^q \rangle$ , as well as for the matrix elements of all the other operators in the  $\Delta B = 2$  effective Hamiltonian, was first described in Ref. [85] for partially quenched Wilson-type quarks in the framework of continuum heavy-meson  $\chi$ PT (HM $\chi$ PT). With staggered fermions, we also must include the effects of taste-violating interactions, using rHMS $\chi$ PT [65].

With the four-quark operators, a careful examination of the Fierz properties shows that there are additional operators with both wrong taste and spin, i.e., wrong ( $\Gamma_1, \Gamma_2$ ). As far as we know, this property of local heavy-staggered four-quark operators has not been discussed in the literature before. The needed rHMS $\chi$ PT expressions are derived in Ref. [86]. We became aware of these contributions after our analysis was nearly complete, so we have not included them in the chiral fit functions used here. We do, however, estimate the associated systematic error on  $\xi$  in our error budget (cf. Sec. V C). Explicit expressions for the chiral fit functions used in this work are given in Appendix A.

The NLO rHMS $\chi$ PT in Eq. (A1) and subsequent equations in the appendix can be schematically written as

$$\begin{aligned} \langle \bar{B}_q | \mathcal{O}_1^q | B_q \rangle &= \frac{2}{3} M_{B_q}^2 f_{B_q}^2 B_q \\ &= M_{B_q} \alpha [1 + (\mathcal{W}_q + \mathcal{T}_q + \mathcal{Q}_q) + L_v m_q \\ &\quad + L_s (2m_l + m_h) + L_a a^2], \end{aligned} \quad (5.1)$$

where  $\alpha, L_v, L_s$ , and  $L_a$  are low-energy constants (LECs) to be determined from fitting the data, the factor of  $M_{B_q}$  comes from the HQET normalization of states, and the masses  $m_q, m_l$ , and  $m_h$  are the light valence, light sea, and strange sea-quark masses, respectively. The light sea quarks are treated as degenerate, and the isospin average is used, i.e.,  $\hat{m} = (m_u + m_d)/2$ . For staggered quarks, the taste-nonsinglet pseudoscalar meson masses are split

$$M_{ij,\rho}^2 = \mu(m_i + m_j) + a^2 \Delta_\rho, \quad (5.2)$$

where  $m_i$  and  $m_j$  are the quark masses and the sixteen meson masses are labeled by their taste representation,  $\rho = P, A, T, V, I$ . The parameters  $\mu$  and the  $\Delta_\rho$ 's are determined from lattice calculations for pions and kaons [87]. Their values are collected in Table IV.

The chiral logarithms,  $\mathcal{W}_q, \mathcal{T}_q, \mathcal{Q}_q$ , stem from wave function renormalization, tadpole, and sunset diagrams,

TABLE IV. Inputs for the priors of the free parameters and for the fixed parameters in the fits. The NLO low-energy constants  $L_v, L_s$ , and  $L_a$  are not constrained in the fits. The parameter  $s$  is given by the quantity  $1/(8\pi^2(r_1 f_\pi)^2)$ . We do not consider errors on the slope  $\mu r_1$  or the taste splittings  $r_1^2 a^2 \Delta_\rho$  because those have negligible effect on the final results. In the right-hand side table, the two last columns correspond to lattice spacings  $a \approx 0.12$  fm and  $a \approx 0.09$  fm. See the text for explanations of the choices of parameters.

Fit parameters (central value $\pm$ width)		
$a \approx 0.12$ fm		$(a \approx 0.09$ fm)
$\beta$		$1 \pm 1$
$g_{B^* B \pi}$		$0.51 \pm 0.20$
$r_1^2 a^2 \delta'_V$	$0.0 \pm 0.07$	$(0.0 \pm 0.07) \times 0.35$
$r_1^2 a^2 \delta'_A$	$-0.28 \pm 0.06$	$(-0.28 \pm 0.06) \times 0.35$
$L_v$		Unconstrained
$L_s$		Unconstrained
$L_a$		Unconstrained
$Q_{1-4}$	$0 \pm s^2$	
$P_{1-3}$	$0 \pm s^2$	

Input (fixed) parameters		
$a \approx 0.12$ fm		$a \approx 0.09$ fm
$f_\pi r_1$		0.2106
$\mu r_1$	6.234	6.382
$r_1^2 a^2 \Delta_P$	0	0
$r_1^2 a^2 \Delta_V$	0.439	0.152
$r_1^2 a^2 \Delta_T$	0.327	0.115
$r_1^2 a^2 \Delta_A$	0.205	0.0706
$r_1^2 a^2 \Delta_I$	0.537	0.206

respectively. The explicit expressions can be found in Appendix A.

In this work, we do not include the effects of the hyperfine splitting  $\Delta^*$  or the light-flavor splittings  $\delta_k$  defined in the Appendix. The wrong spin terms contribute to the tadpole and sunset diagrams [86].

When extrapolating to the physical point, we set the parameters  $\Delta_\rho$  and  $\delta'_{A,V}$  (which describe discretization effects) and the lattice spacing to zero, and set the sea quark masses to their physical values,  $m_l \rightarrow (m_u + m_d)/2$ , and  $m_h \rightarrow m_s$ . We then obtain  $\langle \bar{B}_d^0 | \mathcal{O}_1^d | B_d^0 \rangle$  or  $\langle \bar{B}_s^0 | \mathcal{O}_1^s | B_s^0 \rangle$  by setting  $m_q = m_d$  or  $m_s$ . Thus, it is an extrapolation to the  $u$ - and  $d$ -quark masses and an interpolation to the  $s$ -quark mass.

An additional consideration is that  $SU(3)$  NLO  $\chi$ PT may not be valid for data with masses as large as the strange quark's. It would be desirable to include next-to-next-to-leading order (NNLO) contributions to test the validity of the NLO expression, but the effort needed to calculate the NNLO logs is prohibitive. It is reasonable instead to test the chiral expansion by including just NNLO analytic contributions. Wherever the quark masses or splittings are large enough for such analytic NNLO terms to be significant, the NNLO logarithms should be slowly varying and well approximated by the analytic terms. We follow this strategy and supplement the NLO rHMS $\chi$ PT expressions with NNLO analytic terms in our fits, with prior constraints estimated based on  $\chi$ PT power counting as explained in the next section.

### A. Parametrization of the chiral expression

Dimensionful quantities are extracted first in units of the lattice spacing and then converted to their physical values using the  $r_1$  scale [88,89]. This absolute scale is defined as  $r_1^2 F(r_1) = 1.0$ , where  $F(r_1)$  is the force between static quarks. In our chiral fits, all parameters are first converted to units of  $r_1$  by multiplying by the relative scale  $r_1/a$ . The values for  $r_1/a$  on every MILC ensemble used in our calculation are listed in Table I. After the chiral-continuum extrapolation, we convert from  $r_1$  units with a physical value of  $r_1$ . We take the result obtained by combining the 2009 MILC determination of  $r_1 f_\pi$  [90] and the PDG value of  $f_\pi$  [16]. Following Ref. [59], the error for  $r_1$  is determined by averaging the MILC value with the HPQCD value in Ref. [91] and then by inflating the uncertainty to take conservatively into account the possible correlations coming from the use of the same configurations in both determinations. The final value we use is  $r_1 = 0.3117(22)$  [59]. The error associated with  $r_1$  has a very small effect on the dimensionless quantity  $\xi$ .

The dominant lattice artifacts to take into account in our rHMS $\chi$ PT expressions are expected to be taste-violating contributions of  $O(a^2 \alpha_s^2)$ , since the  $O(a^2 \alpha_s)$  taste-violating effects are absent for asqtad quarks. We parametrize these effects in our fits by defining a quantity  $A_a^2$ ,

which is the ratio of the size of taste violations on lattices with spacing  $a$  to those on the  $a \approx 0.12$  fm lattices. Thus  $A_{0.12 \text{ fm}}^2 = 1$  and

$$A_{0.09 \text{ fm}}^2 \equiv \frac{(\alpha_s^2 a^2)_{0.09 \text{ fm}}}{(\alpha_s^2 a^2)_{0.12 \text{ fm}}} \sim 0.35. \quad (5.3)$$

The NLO rHMS $\chi$ PT function in Eq. (5.1) can then be rewritten as

$$\begin{aligned} \beta_q &\equiv \sqrt{\frac{3}{2} \langle \bar{B}_q | \mathcal{O}_1^q | B_q \rangle / M_{B_q}} \\ &= f_{B_q} \sqrt{M_{B_q} B_{B_q}} \\ &= \beta^\chi \left[ 1 + \frac{1}{2} (\mathcal{Q}_q + \mathcal{W}_q + \mathcal{T}_q) + \frac{L_v}{2} M_{qq}^2 \right. \\ &\quad \left. + \frac{L_s}{2} (2M_{ll}^2 + M_{hh}^2) + \frac{L_a}{2} A_a^2 \right], \end{aligned} \quad (5.4)$$

where  $\beta^\chi = \sqrt{\alpha}$ . The masses  $M_{ij}$  are defined in Eq. (5.2), but here we disregard  $a^2$  corrections in the masses since they can be absorbed by a redefinition of the low-energy constants at higher order in the chiral expansion. To the NLO expression above we add, inside the square brackets, the allowed NNLO analytic terms, which contribute with seven more unknown LECs

$$\begin{aligned} &Q_1 M_{qq}^4 + Q_2 (2M_{ll}^2 + M_{hh}^2)^2 + Q_3 M_{qq}^2 (2M_{ll}^2 + M_{hh}^2) \\ &+ Q_4 (2M_{ll}^4 + M_{hh}^4) + P_1 A_a^2 M_{qq}^2 + P_2 A_a^2 (2M_{ll}^2 + M_{hh}^2) \\ &+ P_3 A_a^4. \end{aligned} \quad (5.5)$$

In the above expressions, we have suppressed the factors of  $r_1$  for simplicity. They can be deduced using dimensional considerations. In these expressions, which are the ones we use as fit functions, we write the analytic terms for convenience as functions of the pseudoscalar masses  $M_{ij}$  rather than the quark masses.

The ratio  $\xi$  can be extracted by first interpolating  $\beta_q$  to  $m_q = m_s$  and extrapolating to  $m_q = m_d$  separately according to expressions (5.4) and (5.5), and then forming the ratio  $\beta_s/\beta_d$ . Alternatively, one can consider the ratio of chiral expressions and expand up to NNLO to obtain

$$\begin{aligned} \xi' &= \frac{\beta_{q'}}{\beta_q} = \xi \frac{\sqrt{M_{B_{q'}}}}{\sqrt{M_{B_q}}} \\ &= 1 + \frac{1}{2} (\mathcal{Q}_{q'} + \mathcal{W}_{q'} + \mathcal{T}_{q'} - \mathcal{Q}_q - \mathcal{W}_q - \mathcal{T}_q) \\ &\quad + \frac{L_v}{2} (M_{q'q'}^2 - M_{qq}^2) + Q_1 (M_{q'q'}^4 - M_{qq}^4) \\ &\quad + Q_3 (M_{q'q'}^2 - M_{qq}^2) (2M_{ll}^2 + M_{hh}^2) \\ &\quad + P_1 (M_{q'q'}^2 - M_{qq}^2) A_a^2, \end{aligned} \quad (5.6)$$

with  $m_{q'}$  fixed to the value closest to  $m_s$ . In Eq. (5.6) we disregard the NNLO terms coming from squaring the NLO terms in the denominator, since they are not necessary to obtain good fits and they are difficult to disentangle from those already included. We can then interpolate/extrapolate to  $m_{q'} = m_s$  and  $m_q = m_d$ . We call these two strategies for the chiral and continuum extrapolation of  $\xi$  the indirect and direct methods, respectively. Many of the fit parameters cancel in the chiral expression for  $\xi'$  in Eq. (5.6), improving the reliability and stability of the fits. In addition, discretization errors of  $O(\alpha_s \Lambda_{\text{QCD}} a, (\Lambda_{\text{QCD}} a)^2)$  from the heavy-quark action that are not included in the chiral perturbation theory, partially cancel in the ratio. We thus choose this method as our preferred fitting strategy.

## B. Results from the chiral fits

In order to perform the chiral fits, we first create 200 bootstrap samples of  $\beta_q$  for each sea- and valence-quark mass combination from the two- and three-point correlator fits. The bootstrap data is then fit to the chiral expression using Bayesian techniques. The fits are simultaneously performed to all ensembles in Table I.

The input and fit parameters are set as in Table IV. We do not impose any constraint on the NLO low-energy constants. For the NNLO LECs we use prior widths based on a simple power counting argument. The NLO analytic terms should be of magnitude similar to the NLO logs, which are  $\sim m_\pi^2/(8\pi^2 f_\pi^2) = s(r_1 m_\pi)^2$ , (with  $s \equiv 1/(8\pi^2(r_1 f_\pi)^2)$ ). Hence, the NNLO terms are  $\sim (m_\pi^2/(8\pi^2 f_\pi^2))^2 = s^2(r_1 m_\pi)^4$ . The taste-violating hairpin parameters,  $\delta'_V$  and  $\delta'_A$ , were also determined from lattice calculations for pions and kaons in Ref. [66]. We constrain the parameters  $\delta'_V$  and  $\delta'_A$  in our fits using the results of Ref. [66] as prior central values and widths. We also take the effective coupling of the  $B^* B \pi$  interaction,  $g_{B^* B \pi}$ , as a fit parameter in our analysis. The prior central value and width we use for this parameter, shown in Table IV, covers the main ranges of determinations of  $g_{B^* B \pi}$  [92–98], as discussed in Ref. [57]. A more recent, precise value of  $g_{B^* B \pi}$ , obtained with  $N_f = 2 + 1$  domain-wall fermions and static  $b$  quarks [99], was not yet available when this stage of the analysis was carried out. Nevertheless, the result obtained by the authors in Ref. [99],  $0.449 \pm 0.047 \pm 0.019$ , falls well within the prior central value and width considered here. For the pion decay constant, we use the PDG value,  $f_\pi = (130.41 \pm 0.20)$  MeV [16].

The fit results for  $\xi$  using different ansätze for the fitting function and the direct and indirect methods explained in Sec. VA are listed in Table V. The results and errors obtained using the direct and indirect methods agree very well, especially when NNLO terms are included. This constitutes a good check of how well our results are encompassing higher-order terms in the chiral expansion, which are different in these two methods.

TABLE V. Results from the rHMS $\chi$ PT fits. Errors are only statistical and obtained from 200 bootstrap samples. For a full discussion of systematic errors, see Sec. VI.

Ansatz	$\chi^2/\text{d.o.f.}$	$\xi$ Direct	$\chi^2/\text{d.o.f.}$	$\xi$ Indirect
NNLO	0.45	$1.268^{+0.035}_{-0.044}$	0.23	$1.255^{+0.034}_{-0.041}$
NLO	0.78	$1.284^{+0.018}_{-0.016}$	0.49	$1.262^{+0.008}_{-0.012}$

In Figs. 5 and 6, we show the NLO and NNLO fit results for  $\xi$  from the direct method as a function of the light valence mass in  $r_1$  units,  $r_1 m_q$ . The top panels in both figures show only the full QCD points,  $m_q = m_l$ , while the bottom panels show all (partially quenched) data included in the fits (see Table I). The fit curve is the same in both panels of each figure. The black lines show the results of the fit in the continuum limit, after the dominant lattice artifacts are removed using rHMS $\chi$ PT, and after interpolating the physical sea and valence strange-quark masses to the physical value, as a function of the valence light-quark mass. The black point is our result for  $\xi$  at the physical masses, and includes statistical errors.

From the spread of data in the bottom panels of both figures (same data), one can see that the light sea-quark mass dependence is mild; all different sea-quark masses (squares or triangles at a particular axis value) agree within one statistical  $\sigma$ . The discretization errors are also small, as can be seen in both the data and the extrapolation lines in the upper panels.

We obtain fits that match the data well and have good  $\chi^2/\text{d.o.f.}$  with only the inclusion of NLO terms, as shown in Fig. 5. When we add the NNLO terms, the central values for  $\xi$  are also within one statistical  $\sigma$ , although errors are significantly larger. This is to be expected, since the NNLO LECs are poorly known. Related to this is the fact that the  $\chi^2/\text{d.o.f.}$  for the NLO fits are larger than for the NNLO fits. At NNLO, we are including extra degrees of freedom with large prior widths that are poorly determined by the fit, so, in practice, we are dividing the same  $\chi^2$  by a larger number of degrees of freedom. In fact, the NNLO fits seem to give a slightly better description of the data, as can be seen in the full QCD plots. The chiral extrapolation for  $\xi$  is also milder in the NNLO case. Based on these arguments and, as mentioned above, the fact that direct and indirect fits agree better at NNLO, we choose the direct NNLO fit for our central value and statistical error. The systematic error associated with our choice of fit function is discussed in Sec. VID.

## VI. ERROR ANALYSIS

In this section, we discuss all sources of systematic uncertainty affecting our calculation of  $\xi$ . The systematic errors have to be added to the statistical uncertainty listed in Table V, which also encompasses our imperfect knowledge about chiral parameters such as  $g_{B^* B \pi}$  and  $\delta'_{V,A}$ .

### A. Heavy-quark mass uncertainty

The mixing parameters depend on the  $b$  quark mass used in our simulations through the hopping parameter  $\kappa_b$ , which is tuned so that the kinetic meson mass,  $M_2$ , agrees with the experimental result. The dispersion relation for a heavy particle can be written for low-momentum as

$$E(\mathbf{p}) = M_1 + \frac{\mathbf{p}^2}{2M_2} - \frac{a^3 W_4}{6} \sum_j p_j^4 - \frac{(\mathbf{p}^2)^2}{8M_4^3} + O(\mathbf{p}^6), \quad (6.1)$$

where  $\kappa_b$  enters into the definitions of  $M_1$  and  $M_2$  [40].  $W_4$  and the deviation of  $M_4$  from  $M_2$  capture lattice artifacts. We calculated two-point functions for the pseudoscalar and vector mesons at several momenta and extract the energy,  $E(\mathbf{p})$ , for each particle at each momentum. A fit to the dispersion relation then determines  $M_2$  and the spin average of the results is taken. The  $\kappa_b$  value is then adjusted until  $M_2$  agrees with the spin-averaged  $B_s$  meson mass.

In this work, we use the values for  $\kappa_b$  on the  $a \approx 0.12$  fm and  $a \approx 0.09$  fm ensembles tuned this way in Ref. [55]. The error in the determination translates into a systematic error in the mixing matrix elements. However, in the ratio  $\xi$  the effect of the uncertainty is minimal, since the corrections go in the same direction in both denominator and numerator, and, thus, largely cancel. In addition, most of the remaining dependence is encoded in the decay constants rather than in the bag parameters, which are very insensitive to the exact values of the quark masses. In Ref. [59], we studied the decay constants with the same choice of actions, parameters, and configurations as here. We expect systematic errors to be very similar in both analyses. We therefore, adopt the error due to the uncertainty in the  $b$  quark mass obtained in Ref. [59] for the ratio of decay constants  $f_{B_s}/f_{B_d}$ , namely, 0.4%, as a good estimate of this systematic error for  $\xi$ .

### B. Higher-order effects in the perturbative matching

The most straightforward and conservative way to estimate the effects of the missing higher order terms in the perturbative matching is to assume two-loop coefficients of order 1 and to multiply the central value by  $\alpha_s^2 = \alpha_V^2(2/a)$ . This estimate gives an error  $\sim 5\%$  for  $f_B \sqrt{B_B}$  on the  $a \approx 0.12$  fm lattices and  $\sim 3.6\%$  on the  $a \approx 0.09$  fm lattices, becoming the main source of uncertainty for this quantity [37]. If there was no mixing between  $\langle \mathcal{O}_1 \rangle$  and  $\langle \mathcal{O}_2 \rangle$  under renormalization, there would be an exact cancellation of the renormalization coefficients for the ratio  $\xi = (f_{B_s} \sqrt{B_{B_s}})/(f_{B_d} \sqrt{B_{B_d}})$ , as long as the valence light-quarks are taken to be massless in the renormalization calculation. The mixing under renormalization prevents this exact cancellation from happening, but the renormalization corrections in the ratio are still largely suppressed, by a factor of  $\langle \mathcal{O}_2^s \rangle / \langle \mathcal{O}_1^s \rangle - \langle \mathcal{O}_2^d \rangle / \langle \mathcal{O}_1^d \rangle$ , with respect to those for a single matrix element. We estimate this suppression factor via the

ratio  $(m_s - m_d)/\Lambda_{\text{QCD}}$  and multiply the perturbative error for  $f_B \sqrt{B_B}$  given above by it. As a result, the perturbative matching uncertainty for  $\xi$  from this estimate is 0.2–0.5% (for  $\Lambda_{\text{QCD}} = 700$  MeV).

The ratio  $\xi$  changes by 0.2% when the one-loop renormalization is omitted entirely, supporting our power counting argument. Another way of estimating  $O(\alpha_s^2)$  effects is by varying the scale  $q^*$  at which  $\alpha_V$  is evaluated. If we change  $q^*$  from our central value of  $2/a$  to  $1/a$  and  $3/a$  we find that the extrapolated  $\xi$  changes between 0.2 and 0.4%.

Since the initial estimate yields the largest uncertainty, 0.5%, we take this as the error associated with the missing higher order terms in the perturbative renormalization. Hence, this source of uncertainty is subdominant in our determination of  $\xi$ .

### C. Mixing with wrong spin four-fermion operators

As mentioned in Sec. V, there are contributions at NLO in rHMS $\chi$ PT originating from the mixing of  $\langle \mathcal{O}_1 \rangle$  with the matrix elements of four-fermion operators of different spin and taste. We have omitted these contributions from our chiral fits, because we discovered these terms after this stage of the analysis was complete. From Figs. 5 and 6, one can see that the effect of the wrong spin mixing is unlikely to be very large, perhaps being mostly absorbed into the LECs.

We cannot include the effects of the wrong spin contributions, because they require the matrix elements of  $\mathcal{O}_3$ , which we have not computed here. Fortunately, however, we have started a more comprehensive analysis of  $B$ - $\bar{B}$  mixing on a larger set of higher-statistics ensembles, including  $\mathcal{O}_3$ . We have added the wrong spin operators to that analysis and find that their inclusion tends to increase the slope of the continuum extrapolated chiral fit function for  $\langle \mathcal{O}_1 \rangle$  and, hence,  $\xi$ . For example, taking priors and widths similar to those in Table IV, we find a 2% increase in  $\xi$ , while for other reasonable choices of the priors the variation is not larger than 3.2%. We add a 3.2% systematic error to account for the missing terms in our chiral extrapolation functions.

### D. Chiral-extrapolation systematics and light-quark discretization

The errors due to the choice of fit ansatz and light-quark discretization effects cannot be disentangled, because every fit ansatz necessarily treats the discretization errors differently. So any estimate of the systematic uncertainty associated with the choice of ansatz also accounts for the light-quark discretization errors left over after removing the dominant ones using rHMS $\chi$ PT.

In Fig. 7, we show the distribution of values for  $\xi$  obtained with the NNLO direct fit for the 200 bootstrap samples analyzed. We check that 200 bootstrap samples is enough to obtain a (nearly) Gaussian distribution, as can be seen in the panel. With the goal of testing our choice of

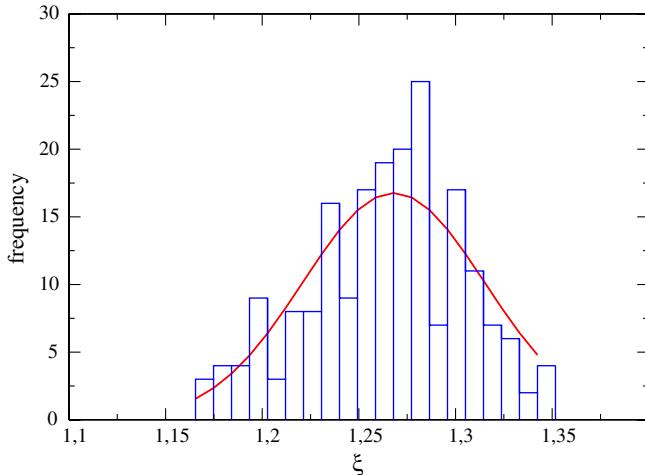


FIG. 7 (color online). Histogram of the distribution of values of  $\xi$  obtained from the 200 bootstrap samples. The curve is a Gaussian distribution corresponding to  $\xi = 1.268 \pm 0.047$  (the NNLO result with augmented errors as explained in the text).

functional form and the error associated with the truncation of the chiral series, we perform fits with only two of the three NNLO terms, omitting each one in turn. All fits give good values of  $\chi^2/\text{d.o.f.}$  and p value. The values of  $\xi$  obtained are scattered around the distribution in Fig. 7 but always within 1.5 statistical  $\sigma$  of the central value. This consistency, together with the fact that the NNLO LECs are not well determined by the fit, indicates that the statistical error already accounts for the possibility of having one of the unknown constants equal or close to zero. If we inflate and symmetrize our statistical errors to  $\pm 0.047$ , we cover the spread of results from the different fitting functions tried (including the NLO one). We take this value as our estimate of both statistical and chiral systematic uncertainties.

An alternative way of estimating the uncertainty in the truncation of the chiral series and the fitting function would be taking the difference between the NLO and the NNLO fits results. If we add this difference with the statistical errors in Table V in quadrature, the uncertainty would be slightly smaller than the  $\pm 0.047$  we are taking as our estimate of these two sources of error.

In the rest of this section, we list the errors associated with the uncertainty of several input parameters used in the continuum and chiral fits, that typically can be estimated by varying the inputs and redoing the fits.

### 1. Light-quark mass uncertainty

The physical values of the light-quark masses used for the extrapolations and interpolations for  $\xi$  are determined by the MILC collaboration [87,100]. They are obtained by making the charged pions and kaons take on their physical values after removal of electromagnetic effects and are listed in Table VI.

TABLE VI. Input for the physical light-quark masses used in the chiral extrapolations. These values were determined by the MILC collaboration [87,100]. Physical values are found from chiral fits that have been extrapolated to the continuum, but masses are still in units of the  $a \approx 0.09$  fm lattice spacing.

Quantity	Physical
$am_s \times 10^2$	2.72 (8)
$a \frac{(m_u+m_d)}{2} \times 10^3$	0.997 (35)
$am_d \times 10^3$	1.40(6)

The error on  $\xi$  due to the light-quark mass uncertainties is obtained by individually varying each quark mass within this uncertainty and repeating the preferred chiral fit and extrapolation. The central values arrived at using each mass variation are compared to the results of the fit which used the central values for the masses, and their differences are added in quadrature. This gives a total systematic error due to the light-quark mass uncertainties of 0.5% for  $\xi$ .

### 2. Uncertainty in the scale $r_1$

The value of  $r_1$  used in this analysis to convert from lattice to physical units, as described in Sec. VA, is  $r_1 = 0.3117(22)$  fm. The results discussed in previous sections are obtained by fixing  $r_1$  to its central value. In order to estimate the uncertainty due to the error in  $r_1$  we change  $r_1$  by  $\pm 0.0022$  fm and all parameters that depend on the physical  $r_1$  are appropriately adjusted. The uncertainty in scale gives a systematic error of 0.2%, which is very small due to the fact that  $\xi$  is a dimensionless quantity and the scale only enters in the normalization to lattice units of the chiral corrections  $(1/(f_\pi r_1)^2)$  and indirectly via the tuning of the quark masses.

### E. Heavy-quark discretization effects

The discretization errors associated with our choice of heavy-quark action to simulate the bottom quarks can be described in terms of the difference in the lattice and continuum Wilson coefficients of higher dimension operators in the HQET expansion. Those come from two sources: the mismatch between continuum and lattice in the Lagrangian and the mismatch in the four-fermion operators whose matrix elements yield  $f_B \sqrt{B_B}$  and, thus,  $\xi$ . For a particular operator  $Q_i$  the error can be written in terms of the usual power counting magnitudes times functions that reflect the particular  $m_0 a$  dependence of the action [56,101]

$$\text{error}_i = z_i f_i(m_0 a) (a \Lambda_{\text{QCD}})^{s_i}, \quad (6.2)$$

where  $s_i = \dim Q_i - 4$  for Lagrangian operators  $Q_i$  of dimension 4 and 5, and  $s_i = \dim Q_i - 6$  for four-fermion operators  $Q_i$  of dimension 7 and 8, and the  $z_i$  are constants. The functions  $f_i(m_0 a)$  can be deduced from Refs. [40,102] and were discussed in detail in Ref. [56] for the form

factors parametrizing  $B \rightarrow \pi l \nu$  decays and in Ref. [59] for heavy-light decay constants. A detailed study of these corrections for the matrix elements of all the operators contributing to neutral  $B$  mixing in the SM and beyond will be presented elsewhere [77]. Here we only summarize the sources of the different corrections for  $\langle \mathcal{O}_1 \rangle$  and  $\xi$ . The explicit form of the different functions  $f_i(m_0 a)$  can be found in Appendix B.

From the Lagrangian, there are  $O(a^2)$  errors and  $O(\alpha_s a)$  errors which are identical to those in Eqs. (A12) and (A19) in Ref. [59]. They are proportional to the functions  $f_E(m_0 a)$  in Eq. (B1) and  $f_B(m_0 a)$  in Eq. (B2) of Appendix B, respectively.

From the four-fermion operators, we have  $O(a^2)$  errors coming from higher order corrections to the rotation relation in Eq. (2.29). They are generated by the mismatch between lattice and continuum coefficients of the operators  $\bar{q}\Gamma\mathbf{D}^2 b$ ,  $\bar{q}\Gamma i \cdot \mathbf{B} b$  and  $\bar{q}\Gamma \boldsymbol{\alpha} \cdot \mathbf{E} b$  in the same way as in Eqs. (A13) and (A14) of Ref. [59], but with an extra overall factor of two due to the fact that we have two heavy fields in our leading-order operator, not one. These corrections are proportional to the functions  $f_X(m_0 a)$  and  $f_Y(m_0 a)$  in Eq. (B3) of Appendix B, respectively.

The last contribution, and the least straightforward, comes from the  $O(\alpha_s a)$  corrections to the four-fermion operators. In principle, to subleading order, there are a basis of twelve new local operators in the effective Hamiltonian. However, using symmetry constraints, Fierz transformations, and rewriting some combinations as total derivatives, only five independent operators remain [37,103]. Because we separate the temporal and spatial parts of the operators through this analysis, the total temporal and spatial components of those five operators should be compared with the temporal and spatial parts of the leading operators. As explained in Ref. [77], this produces a difference of  $3f_3(m_0 a)$ , where  $f_3$  is given in Eq. (B4).

In Table VII, we list all the contributions together with the functions  $f_i$ , and the proportionality constant  $z_i$  in Eq. (6.2). We also list the numerical values of the different contributions to the heavy-quark discretization error in percentage for  $f_B\sqrt{B_B}$ . In order to get the numerical results, we use  $\Lambda_{\text{QCD}} = 700$  MeV and  $\alpha_s = \alpha_V(2/a)$  listed in Table II. The final heavy-quark discretization error for

TABLE VII. Heavy-quark discretization errors given in percentages.

Contribution	$f_i$	$z_i$	Error $f_B\sqrt{B_B}$ (%) (coarse,fine)	Error $\xi$ (%) (coarse,fine)
$\mathcal{O}(a^2)$ Lagrangian	$f_E$	2	(0.28,0.16)	
$\mathcal{O}(a_s a)$ Lagrangian	$f_B$	2	(0.96,0.58)	
$\mathcal{O}(a^2)$ Operator	$f_X$	4	(1.29,0.74)	
	$f_Y$	2	(0.23,0.18)	
$\mathcal{O}(a_s a)$ Operator	$f_3$	3	(1.32,0.75)	
Total error			(2.1,1.3)	(0.2,0.1)

$f_B\sqrt{B_B}$  is 1.3% for the  $a \approx 0.09$  fm ensembles and 2.1% for the  $0.12$  fm ones.

These errors largely cancel in  $\xi$ . The effect of the cancellation on the error can be estimated by multiplying the errors in  $f_B\sqrt{B_B}$  by a factor of  $(m_s - m_d)/\Lambda_{\text{QCD}}$  which gives a final heavy-quark discretization error for  $\xi$  of 0.2% for the coarse lattice and 0.1% for the fine lattice. This agrees well with the estimate of this type of error for the ratio  $f_{B_s}/f_B$  [59],  $\sim 0.3\%$ , using a very similar set of data and statistics. The strategy followed in Ref. [59] differs from the one described here. In that paper, terms of the form in Eq. (6.2) were directly added to the chiral and continuum extrapolation fitting functions with a coefficient of order one to be determined by the fit. Ultimately, we would like to employ that strategy also for  $B^0-\bar{B}^0$  mixing studies. For this work, however, we simply take the larger estimate from the ratio  $f_{B_s}/f_B$  as our estimate of the uncertainty in  $\xi$  due to heavy-quark discretization errors.

## F. Finite volume corrections

In order to evaluate the finite volume corrections in our calculation, we follow the prescription in Refs. [65,104]. The MILC lattices are large enough in the time direction that it can be treated as infinite to a very good approximation, so we are interested in corrections due to finite spatial volume only. They are estimated by replacing infinite-volume integrals in the chiral expression with finite sums over the spatial momentum.

Including finite volume corrections in the chiral expressions and redoing the fits reveals negligible errors,  $<0.1\%$ .

## G. Tuning of the tadpole parameter $u_0$

The tadpole improvement factor  $u_0$  is a parameter of the gauge and asqtad staggered (sea) quark action and is determined from the fourth root of the average plaquette. The tadpole improvement factor also enters into the valence light and heavy quark actions. On the  $a \approx 0.09$  fm ensembles, the valence quarks are generated with the same values of  $u_0$  as the sea. However, on the  $a \approx 0.12$  fm ensembles, the valence quark actions use values of  $u_0$  obtained from the average link in Landau gauge instead. The differences between the values of  $u_0$  obtained with the two methods is around 3–4%.

The effect on  $f_{B_s}/f_B$  of the mismatch between  $u_0$  values in the valence and the sea sectors of the  $a \approx 0.12$  fm ensembles was estimated to be  $<0.1\%$  in Ref [59]. Since this is much smaller than the errors due to statistics, chiral fits, and continuum and chiral extrapolation, we take this estimate as our error on  $\xi$ .

## VII. DISCUSSION OF RESULTS AND FUTURE IMPROVEMENTS

The error budget for the  $SU(3)$ -breaking mixing parameter  $\xi$  described in the previous sections is summarized

TABLE VIII. Complete error budget and total error for the  $B^0$  mixing parameter  $\xi$ . All errors are given in percentages.

Source of uncertainty	Error (%)
Statistics $\oplus$ light-quark disc. $\oplus$ chiral extrapolation	3.7
Mixing with wrong-spin operators	3.2
Heavy-quark discretization	0.3
Scale uncertainty ( $r_1$ )	0.2
Light-quark masses	0.5
One-loop matching	0.5
Tuning $\kappa_b$	0.4
Finite volume	0.1
Mistuned coarse $u_0$	0.1
Total error	5.0

in Table VIII. For the first error in the table we prefer not to attempt to disentangle the statistical, light-quark discretization, and chiral extrapolation errors since, as explained in Sec. VID, the lack of knowledge about the LECs at NNLO makes a reliable separation impossible. Our final result is

$$\xi = 1.268 \pm 0.063. \quad (7.1)$$

The total uncertainty is dominated by the combined statistical, light-quark discretization, and chiral extrapolation error, and the uncertainty associated with the wrong spin operators in the chiral-continuum extrapolation.

Combining our result in Eq. (7.1) with the averages of the experimentally measured values of the mass differences  $\Delta M_d = (0.507 \pm 0.004) \text{ ps}^{-1}$  [16] and  $\Delta M_s = (17.69 \pm 0.08) \text{ ps}^{-1}$  [105], and the meson masses  $M_{B_s^0} = (5366.0 \pm 0.9) \text{ MeV}$  and  $M_{B_d^0} = (5279.5 \pm 0.5) \text{ MeV}$  [16], we quote a value for the ratio of the CKM matrix elements

$$\left| \frac{V_{td}}{V_{ts}} \right| = 0.216 \pm 0.011, \quad (7.2)$$

assuming no new physics in  $B_{(s)}^0$ - $\bar{B}_{(s)}^0$  mixing. The error includes the uncertainties in the  $B$ -meson masses and mass differences but it is strongly dominated by the error in  $\xi$ .

We can also take our result for  $\xi$  and combine it with the value of the decay constant ratio  $f_{B_s}/f_{B_d} = 1.229 \pm 0.026$  calculated by our collaboration [59] to determine the ratio of bag parameters

$$\frac{B_{B_s}}{B_{B_d}} = \xi^2 \left( \frac{f_{B_d}}{f_{B_s}} \right)^2 = 1.06 \pm 0.11. \quad (7.3)$$

The two results for  $\xi$  and  $f_{B_s}/f_{B_d}$  are correlated, but the statistical analyses were done independently so we cannot include the correlations in calculating the uncertainty in the ratio of bag parameters. Therefore, the error shown in the result of Eq. (7.3) is overestimated. However, as part of the future work, we plan to perform a common analysis of matrix elements and decay constants, from which we will

be able to account for correlations in extracting the value of the bag parameters and thus greatly reduce the error in Eq. (7.3). We will do the same for the individual bag parameters corresponding to all the operators in the basis in Eq. (1.3).

Our result for  $\xi$  in Eq. (7.1) is in good agreement with the HPQCD value obtained in Ref. [32],  $\xi = 1.258(33)$ . Note, however, that HPQCD did not estimate the effects of the wrong spin operators that appear in the complete NLO chiral expression, so the full error in their result may be somewhat bigger than what was quoted. The agreement of these two determinations of  $\xi$  provides an excellent check of the methodology and systematic error study in both analyses. In addition, it helps to increase the confidence in the robustness of lattice results for a parameter of great importance in phenomenological studies. In this article, we have established and tested the methodology to apply to broader studies of  $B^0$  mixing with the same lattice formulations for light and bottom quarks as used here.

Statistical errors could be reduced significantly by expanding the analysis to include the full set of available configurations (approximately 2000) at each of the  $a \approx 0.12 \text{ fm}$  and  $a \approx 0.09 \text{ fm}$  ensembles. The current runs of our collaboration on the extended ensembles are also implementing sources located at a random spatial and time location to reduce further the statistical errors. We expect a reduction of the statistical errors by about a factor of two.

The other dominant error of our calculation, the omission in the rHMS $\chi$ PT analysis of terms generated by wrong spin operators, will be eliminated when a complete analysis is done with the full rHMS $\chi$ PT expressions [86]. A result for  $\xi$  that properly includes the wrong spin terms requires the calculation of the continuum matrix elements not only of the operator  $\mathcal{O}_1$  as we have done in this work, but also of  $\mathcal{O}_2$  and  $\mathcal{O}_3$ , and simultaneous chiral and continuum extrapolations of all three matrix elements.

The discretization errors, related to both heavy and light quarks, will be reduced in a straightforward way by simulations at smaller lattice spacing, i.e., on the  $a \approx 0.06 \text{ fm}$  and  $a \approx 0.045 \text{ fm}$  MILC lattices. The reduction of both statistical and discretization errors will also yield cleaner and more accurate continuum and chiral extrapolations. Including data at smaller lattice spacings will also reduce the uncertainty associated with the perturbative matching from the reduction of  $\alpha_s = \alpha_V(2/a)$ . Although not relevant for the reduction of the total error in  $\xi$ , this will be important in the determination of the matrix elements  $\langle \mathcal{O}_i \rangle$  themselves.

Similarly, although the uncertainty associated with heavy-quark discretization effects is a subdominant source of error in the determination of  $\xi$ , it is one of the main errors in the determination of  $\langle \mathcal{O}_i \rangle$  [37]. In order to have a more reliable, data-driven estimation of these effects, in our on-going analyses we plan to employ the strategy used in Ref. [59], in which terms like the ones in Eq. (6.2) are

included in the chiral-continuum extrapolation fitting form with free parameters to be determined from the fit.

Our new analysis, which incorporates the improvements mentioned above, includes the study of the matrix elements of all five operators that contribute to  $H_{\text{eff}}^{\Delta B=2}$  [29]. This will allow not only the precise SM determination of  $\Delta M_{s,d}$ ,  $\Delta \Gamma_{s,d}$ , and  $\xi$ , but will also provide the nonperturbative inputs needed to put constraints on BSM models using experimental data on  $B^0$  mixing and related observables.

## ACKNOWLEDGMENTS

Computations for this work were carried out with resources provided by the USQCD Collaboration, the Argonne Leadership Computing Facility, the National Energy Research Scientific Computing Center, and the Los Alamos National Laboratory, which are funded by the Office of Science of the U.S. Department of Energy; and with resources provided by the National Institute for Computational Science, the Pittsburgh Supercomputer Center, the San Diego Supercomputer Center, and the Texas Advanced Computing Center, which are funded through the National Science Foundation's Teragrid/XSEDE Program. This work was supported in part by the U.S. Department of Energy under Grants No. DE-FC02-06ER41446 (C.D., L.L., M.B.O.), No. DE-FG02-91ER40661 (S.G.), No. DE-FG02-91ER40677 (C.M.B., R.T.E., E.D.F., E.G., R.J., A.X.K.), No. DE-FG02-91ER40628 (C.B.), No. DE-FG02-04ER-41298 (D.T.); by the National Science Foundation under Grants No. PHY-0555243, No. PHY-0757333, No. PHY-0703296, No. PHY10-67881 (C.D., L.L., M.B.O.), No. PHY-0757035 (R.S.), and No. PHY-0704171 (J.E.H.); by the MICINN, Spain, under Grant No. FPA2010-16696 and

*Ramón y Cajal* program (E.G.); by Junta de Andalucía, Spain, under Grants No. FQM-101, No. FQM-330, No. FQM-03048, and No. FQM-6552 (E.G.); by the URA Visiting Scholars' program (C.M.B., R.T.E., E.G., M.B.O.); by the Fermilab Fellowship in Theoretical Physics (C.M.B.); and by the Science and Technology Facilities Council and the Scottish Universities Physics Alliance (J.L.). This manuscript has been co-authored by employees of Brookhaven Science Associates, LLC, under Contract No. DE-AC02-98CH10886 with the U.S. Department of Energy. Fermilab is operated by Fermi Research Alliance, LLC, under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

## APPENDIX A: STAGGERED CHIRAL PERTURBATION THEORY FOR $B^0$ - $\bar{B}^0$ MIXING

In this appendix we describe the functional form we use in the chiral and continuum extrapolation of the matrix elements  $\langle \bar{B}_q^0 | \mathcal{O}_1^q | B_q^0 \rangle$ . Further discussion, as well as complete NLO rHMS $\chi$ PT expressions for  $\langle \bar{B}_q^0 | \mathcal{O}_1^q | B_q^0 \rangle$  with  $i = 1, \dots, 5$  and the corresponding bag parameters can be found in Ref. [86].

At NLO in rHMS $\chi$ PT and at first order in the heavy-quark expansion we use

$$\begin{aligned} \langle \bar{B}_q^0 | \mathcal{O}_1^q | B_q^0 \rangle = & \alpha \left( 1 + \frac{\mathcal{W}_{q\bar{b}} + \mathcal{W}_{b\bar{q}}}{2} + \mathcal{T}_q + \mathcal{Q}_q \right) \\ & + L_v m_q + L_s (2m_l + m_h) + L_a a^2. \end{aligned} \quad (\text{A1})$$

$\alpha$ ,  $L_v$ ,  $L_s$ , and  $L_a$  are constants to be determined from the fits to lattice data. The quantities in script for the partially quenched  $2+1$  ( $m_u = m_d \neq m_s$ ) case are

$$\begin{aligned} \mathcal{W}_{q\bar{b}} = \mathcal{W}_{b\bar{q}} &= \frac{ig_{B^*B\pi}^2}{f_\pi^2} \left\{ \frac{1}{16} \sum_{S,\rho} N_\rho \mathcal{H}_{S,q,\rho}^{\Delta^*+\delta_{S,q}} + \frac{1}{3} \left[ R_{X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \frac{\partial \mathcal{H}_{X,I}^{\Delta^*}}{\partial m_{X_I}^2} - \sum_{j \in \{M_I^{(5)}\}} D_{j,X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \mathcal{H}_{j,I}^{\Delta^*} \right] \right. \\ &\quad \left. + a^2 \delta_V' \left[ R_{X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) \frac{\partial \mathcal{H}_{X,V}^{\Delta^*}}{\partial m_{X_V}^2} - \sum_{j \in \{M_V^{(7)}\}} D_{j,X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) \mathcal{H}_{j,V}^{\Delta^*} \right] + (V \rightarrow A) \right\}, \end{aligned} \quad (\text{A2})$$

$$\begin{aligned} \mathcal{T}_q &= \frac{-i}{f_\pi^2} \left\{ \frac{1}{16} \sum_{S,\rho} N_\rho I_{qS,\rho} + \frac{1}{16} \sum_\rho N_\rho I_{X,\rho} + \frac{2}{3} \left[ R_{X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \left( \frac{\partial I_{X_I}}{\partial m_{X_I}^2} \right) - \sum_{j \in \{M_I^{(5)}\}} D_{j,X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) I_j \right] \right. \\ &\quad \left. + a^2 \delta_V' \left[ R_{X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) \left( \frac{\partial I_{X_V}}{\partial m_{X_V}^2} \right) - \sum_{j \in \{M_V^{(7)}\}} D_{j,X_V}^{[3,2]}(\{M_{X_V}^{(7)}\}; \{\mu_V\}) I_j \right] + (V \rightarrow A) \right\}, \end{aligned} \quad (\text{A3})$$

$$\mathcal{Q}_q = \frac{-ig_{B^*B\pi}^2}{f_\pi^2} \left\{ \frac{1}{16} \sum_\rho N_\rho \mathcal{H}_{X,\rho}^{\Delta^*} + \frac{1}{3} \left[ R_{X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \left( \frac{\partial \mathcal{H}_{X,I}^\Delta}{\partial m_{X_I}^2} \right) - \sum_{j \in \{M_I^{(5)}\}} D_{j,X_I}^{[2,2]}(\{M_{X_I}^{(5)}\}; \{\mu_I\}) \mathcal{H}_{j,I}^\Delta \right] \right\}. \quad (\text{A4})$$

In the equations above, the index  $\rho$  runs over the taste representation ( $P, A, T, V, I$ ) with degeneracies  $N_\rho$  ( $N_\rho = 1, 4, 6, 4, 1$ , respectively), and  $S$  runs over the sea flavors  $u, d, s$ . The meson  $X$  is made of two light valence quarks  $q$ , and  $m_X$  is its mass. The functions  $\mathcal{H}$  and  $I$  are the integrals defined in Appendix A of Ref. [85]. The subscripts on those functions label the flavor and taste of the meson masses at which they are evaluated.

The superscript in  $\mathcal{H}$  is the second argument for that function as defined in Ref. [85]. In addition to the hyperfine splitting  $\Delta^* = M_{B^*} - M_B$ , it includes a light flavor splitting  $\delta_{S,q}$  whenever the light flavor of the vector meson in the loop is different from the external flavor.

The splitting is  $\delta_{S,q} \equiv M_{B^0S} - M_{B_q^0} = 2\lambda_1\mu(m_S - m_q)$ , where  $\lambda_1$  and  $\mu$  are low-energy constants. The constant  $\lambda_1$  comes from heavy quark effective theory, and  $\mu$  is defined in Eq. (5.2).

The residue functions  $R_j^{[n,k]}$  and  $D_{j,l}^{[n,k]}$  in the expressions above are defined by Ref. [63]

$$\begin{aligned} R_j^{[n,k]}(\{m\}, \{\mu\}) &\equiv \frac{\prod_{a=1}^k (\mu_a^2 - m_j^2)}{\prod_{i \neq j} (m_i^2 - m_j^2)}, \\ D_{j,l}^{[n,k]}(\{m\}, \{\mu\}) &\equiv -\frac{d}{dm_l^2} R_j^{[n,k]}(\{m\}, \{\mu\}). \end{aligned} \quad (\text{A5})$$

The mass combinations appearing as arguments of these functions in the  $2 + 1$  partially quenched theory are

$$\begin{aligned} \{M_X^{(5)}\} &\equiv \{m_\eta, m_X\}, \\ \{M_X^{(7)}\} &\equiv \{m_\eta, m_{\eta'}, m_X\}, \\ \{\mu\} &\equiv \{m_L, m_H\}, \end{aligned} \quad (\text{A6})$$

where  $m_L$  is the meson mass made from  $l\bar{l}$  sea quarks, and  $m_H$  is the meson mass made from  $h\bar{h}$  sea quarks. The tastes of these mesons are indicated explicitly in the equations above.

Since we are not including the effects of the hyperfine splitting  $\Delta^*$  or the light flavor splittings  $\delta_k$  in this work, the functions  $\mathcal{H}$  and  $I$  appearing in the wave function, tadpole, and sunset contributions simplify to

$$i\mathcal{H}_{k,\Xi}^0 = -3iI_{k,\Xi} = -\frac{3}{16\pi^2} M_{k,\Xi}^2 \ln\left(\frac{M_{k,\Xi}^2}{\Lambda_\chi^2}\right). \quad (\text{A7})$$

## APPENDIX B: FUNCTIONS PARAMETRIZING HEAVY-QUARK DISCRETIZATION ERRORS

In this appendix we collect the functions  $f_i$  needed in Eq. (6.2) to estimate the heavy-quark discretization errors affecting our calculation. For details on the origin of these functions and the effects of higher-dimension operators in the Lagrangian, see Ref. [102]. For further details on the application to the estimation of heavy-quark discretization errors in  $B^0$ - $\bar{B}^0$  mixing, see Ref. [77].

(i)  $O(a^2)$  errors from the Lagrangian:

$$f_E(m_0a) = \frac{1}{2} \left[ \frac{(1 + m_0a) - 1}{m_0a(2 + m_0a)(1 + m_0a)} \right. \\ \left. - \frac{1}{4(1 + m_0a)^2} \right]. \quad (\text{B1})$$

(ii)  $O(\alpha_s a)$  errors from the Lagrangian:

$$f_B(m_0a) = \frac{\alpha_s}{2(1 + m_0a)}. \quad (\text{B2})$$

(iii)  $O(a^2)$  errors from the four-fermion operator:

$$\begin{aligned} f_X(m_0a) &= \frac{1}{2} \left[ \frac{1}{2(1 + m_0a)} \right. \\ &\quad \left. - \left( \frac{m_0a}{2(2 + m_0a)(1 + m_0a)} \right)^2 \right] \\ f_Y(m_0a) &= \frac{2 + 4m_0a + (m_0a)^2}{4(1 + m_0a)^2(2 + m_0a)^2}. \end{aligned} \quad (\text{B3})$$

(iv)  $O(\alpha_s a)$  errors from the four-fermion operator:

$$f_3(m_0a) = \frac{\alpha_s}{2(2 + m_0a)}. \quad (\text{B4})$$

## APPENDIX C: PRIOR CENTRAL VALUES AND WIDTHS FOR THE CORRELATOR FITS

In Table IX we collect the prior central values and widths used in the correlator fits described in Sec. IV. The amplitude parameters are defined in Eqs. (2.21) and (2.22), and the energy differences are defined as  $\Delta E_{i+1,i} \equiv a(E_{i+1} - E_i)$ .

TABLE IX. The priors with index 0 refer to the ground state. Superscripts  $d$  and  $1S$  refer to the local and  $1S$  smeared sources, respectively. Higher-energy state priors have indices  $i$  and  $j$ . The prime in  $E'_i$  refers to an opposite parity (oscillating) state.

	Prior central value	Prior width
$Z_0^{1S}$	2.2	0.5
$Z_i^{1S}$	0.01	0.5
$Z_0^d$	0.45	0.45
$Z_i^d$	0.01	1
$O_{00}$	0.01	0.02
$O_{ij}$	0.01	0.1
$E_0$ (0.12 fm)	1.95	0.15
$E'_0$ (0.12 fm)	2.25	0.15
$E_0$ (0.09 fm)	1.65	0.15
$E'_0$ (0.09 fm)	1.85	0.15
$\log \Delta E_{i+1,i}$	-1.5	0.5
$\log \Delta E'_{i+1,i}$	-1.5	0.5

- [1] E. Lunghi and A. Soni, *Phys. Lett. B* **697**, 323 (2011).
- [2] V.M. Abazov *et al.* (D0 Collaboration), *Phys. Rev. Lett.* **105**, 081801 (2010).
- [3] J. Laiho, E. Lunghi, and R. Van De Water, *Proc. Sci., FPCP2010* (2010) 040.
- [4] A. Lenz *et al.*, [arXiv:1203.0238](#).
- [5] E. Lunghi and A. Soni, *Phys. Rev. Lett.* **104**, 251802 (2010).
- [6] A. Lenz *et al.*, *Phys. Rev. D* **83**, 036004 (2011).
- [7] A.J. Bevan *et al.* (UTfit Collaboration), *Proc. Sci., ICHEP2010* (2010) 270.
- [8] A. Lenz and U. Nierste, *J. High Energy Phys.* **06** (2007) 072.
- [9] M. Bona *et al.* (UTfit Collaboration), *PMC Phys. A* **3**, 6 (2009).
- [10] T. Aaltonen *et al.* (CDF Collaboration), *Phys. Rev. D* **85**, 072002 (2012).
- [11] V.M. Abazov *et al.* (D0 Collaboration), *Phys. Rev. D* **85**, 032006 (2012).
- [12] K. Anikeev *et al.*, [arXiv:hep-ph/0201071](#).
- [13] A. Abulencia *et al.* (CDF Collaboration), *Phys. Rev. Lett.* **97**, 242003 (2006).
- [14] V.M. Abazov *et al.* (D0 Collaboration), *Phys. Rev. Lett.* **97**, 021802 (2006).
- [15] R. Aaij *et al.* (LHCb Collaboration), *Phys. Lett. B* **709**, 177 (2012).
- [16] K. Nakamura *et al.* (Particle Data Group), *J. Phys. G* **37**, 075021 (2010).
- [17] M. Bona *et al.* (UTfit Collaboration), *J. High Energy Phys.* **03** (2008) 049.
- [18] B.A. Dobrescu and G.Z. Krnjaic, *Phys. Rev. D* **85**, 075020 (2012).
- [19] W. Altmannshofer and M. Carena, *Phys. Rev. D* **85**, 075006 (2012).
- [20] A.J. Buras and J. Girrbach, *J. High Energy Phys.* **03** (2012) 052.
- [21] M. Blanke, A.J. Buras, K. Gemmeler, and T. Heidsieck, *J. High Energy Phys.* **03** (2012) 024.
- [22] A.J. Buras, M. Nagai, and P. Paradisi, *J. High Energy Phys.* **05** (2011) 005.
- [23] A.J. Buras, K. Gemmeler, and G. Isidori, *Nucl. Phys.* **B843**, 107 (2011).
- [24] A.J. Buras, B. Duling, T. Feldmann, T. Heidsieck, C. Promberger, and S. Recksiegel, *J. High Energy Phys.* **09** (2010) 106.
- [25] L. Wolfenstein, *Phys. Rev. Lett.* **51**, 1945 (1983).
- [26] A.J. Buras, M.E. Lautenbacher, and G. Ostermaier, *Phys. Rev. D* **50**, 3433 (1994).
- [27] F. Gabbiani, E. Gabrielli, A. Masiero, and L. Silvestrini, *Nucl. Phys.* **B477**, 321 (1996).
- [28] D. Becirevic, V. Gimenez, G. Martinelli, M. Papinutto, and J. Reyes, *J. High Energy Phys.* **04** (2002) 025.
- [29] C.M. Bouchard *et al.* (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LATTICE2011* (2011) 274.
- [30] T. Inami and C.S. Lim, *Prog. Theor. Phys.* **65**, 297 (1981); **65**, 1772(E) (1981).
- [31] A.J. Buras, M. Jamin, and P.H. Weisz, *Nucl. Phys.* **B347**, 491 (1990).
- [32] E. Gámiz, C.T.H. Davies, G.P. Lepage, J. Shigemitsu, and M. Wingate (HPQCD Collaboration), *Phys. Rev. D* **80**, 014503 (2009).
- [33] C. Albertus *et al.* (RBC and UKQCD Collaborations), *Phys. Rev. D* **82**, 014505 (2010).
- [34] R.T. Evans, A.X. El-Khadra, and M. Di Pierro (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LAT2006* (2006) 081.
- [35] R. Evans, E. Gámiz, A.X. El-Khadra, and M. Di Pierro (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LAT2007* (2007) 354.
- [36] R.T. Evans *et al.* (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LAT2008* (2008) 052.
- [37] R.T. Evans *et al.* (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LAT2009* (2009) 245.
- [38] G.P. Lepage, L. Magnea, C. Nakhleh, U. Magnea, and K. Hornbostel, *Phys. Rev. D* **46**, 4052 (1992).
- [39] B. Sheikholeslami and R. Wohlert, *Nucl. Phys.* **B259**, 572 (1985).
- [40] A.X. El-Khadra, A.S. Kronfeld, and P.B. Mackenzie, *Phys. Rev. D* **55**, 3933 (1997).
- [41] E. Golowich, J. Hewett, S. Pakvasa, and A.A. Petrov, *Phys. Rev. D* **76**, 095009 (2007).
- [42] C.W. Bernard, T. Burch, K. Orginos, D. Toussaint, T.A. DeGrand, C. DeTar, S. Datta, S. Gottlieb, U.M. Heller, and R. Sugar (MILC Collaboration), *Phys. Rev. D* **64**, 054506 (2001).
- [43] M.G. Alford, W. Dimm, G.P. Lepage, G. Hockney, and P.B. Mackenzie, *Phys. Lett. B* **361**, 87 (1995).
- [44] M. Luscher and P. Weisz, *Phys. Lett.* **158B**, 250 (1985).
- [45] S. Prelovsek, *Phys. Rev. D* **73**, 014506 (2006).
- [46] C. Bernard, M. Golterman, and Y. Shamir, *Phys. Rev. D* **73**, 114511 (2006).
- [47] C. Bernard, *Phys. Rev. D* **73**, 114503 (2006).
- [48] Y. Shamir, *Phys. Rev. D* **71**, 034509 (2005).
- [49] Y. Shamir, *Phys. Rev. D* **75**, 054503 (2007).
- [50] C. Bernard, M. Golterman, and Y. Shamir, *Phys. Rev. D* **77**, 074505 (2008).
- [51] S.R. Sharpe, *Proc. Sci., LAT2006* (2006) 022.
- [52] A.S. Kronfeld, *Proc. Sci., LAT2007* (2007) 016.
- [53] M. Golterman, *Proc. Sci., CONFINEMENT8* (2008) 014.
- [54] G.C. Donald, C.T.H. Davies, E. Follana, and A.S. Kronfeld, *Phys. Rev. D* **84**, 054504 (2011).
- [55] C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **83**, 034503 (2011).
- [56] J.A. Bailey *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **79**, 054507 (2009).
- [57] C. Bernard *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **79**, 014506 (2009).
- [58] J.A. Bailey *et al.* (Fermilab Lattice and MILC Collaborations), *Proc. Sci., LATTICE2010* (2010) 311.
- [59] A. Bazavov *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **85**, 114506 (2012).
- [60] J.A. Bailey *et al.* (Fermilab Lattice and MILC Collaborations), *Phys. Rev. D* **85**, 114502 (2012).
- [61] W. Lee and S. Sharpe, *Phys. Rev. D* **60**, 114503 (1999).
- [62] C. Aubin and C. Bernard, *Phys. Rev. D* **68**, 034014 (2003).
- [63] C. Aubin and C. Bernard, *Phys. Rev. D* **68**, 074011 (2003).
- [64] S.R. Sharpe and R.S. Van de Water, *Phys. Rev. D* **71**, 114505 (2005).
- [65] C. Aubin and C. Bernard, *Phys. Rev. D* **73**, 014515 (2006).
- [66] A. Bazavov *et al.* (MILC Collaborations), *Rev. Mod. Phys.* **82**, 1349 (2010).
- [67] G.P. Lepage, *Phys. Rev. D* **59**, 074502 (1999).

- [68] K. Symanzik, *Nucl. Phys.* **B226**, 205 (1983).
- [69] G. P. Lepage and P. B. Mackenzie, *Phys. Rev. D* **48**, 2250 (1993).
- [70] Z. Hao, G. M. von Hippel, R. R. Horgan, Q. J. Mason, and H. D. Trottier, *Phys. Rev. D* **76**, 034507 (2007).
- [71] M. Wingate, J. Shigemitsu, C. T. H. Davies, G. P. Lepage, and H. D. Trottier, *Phys. Rev. D* **67**, 054505 (2003).
- [72] C. R. Allton, arXiv:hep-lat/9610016v1.
- [73] L. Susskind, *Phys. Rev. D* **16**, 3031 (1977); H. S. Sharatchandra, H. J. Thun, and P. Weisz, *Nucl. Phys.* **B192**, 205 (1981).
- [74] A. S. Kronfeld, *Phys. Rev. D* **62**, 014505 (2000).
- [75] G. P. Lepage and P. B. Mackenzie, *Phys. Rev. D* **48**, 2250 (1993).
- [76] E. Gámiz, J. Shigemitsu, and H. Trottier, *Phys. Rev. D* **77**, 114505 (2008).
- [77] E. Gámiz, A. El-Khadra, and A. Kronfeld (unpublished).
- [78] Q. Mason, H. D. Trottier, C. T. H. Davies, K. Foley, A. Gray, G. P. Lepage, M. Nobes, and J. Shigemitsu (HPQCD and UKQCD Collaborations), *Phys. Rev. Lett.* **95**, 052002 (2005).
- [79] S. J. Brodsky, G. P. Lepage, and P. B. Mackenzie, *Phys. Rev. D* **28**, 228 (1983).
- [80] K. Hornbostel, G. P. Lepage, and C. Morningstar, *Phys. Rev. D* **67**, 034023 (2003).
- [81] J. L. Richardson, *Phys. Lett.* **82B**, 272 (1979).
- [82] M. Di Pierro, A. X. El-Khadraa, S. Gottlieb, A. S. Kronfeld, P. B. Mackenzie, D. P. Menscher, M. B. Oktay, and J. N. Simone, *Nucl. Phys. B, Proc. Suppl.* **119**, 586 (2003).
- [83] G. P. Lepage, B. Clark, C. T. H. Davies, K. Hornbostel, P. B. Mackenzie, C. Morningstar, H. Trottier, *Nucl. Phys. B, Proc. Suppl.* **106**, 12 (2002).
- [84] C. Morningstar, *Nucl. Phys. B, Proc. Suppl.* **109**, 185 (2002).
- [85] W. Detmold and C. J. D. Lin, *Phys. Rev. D* **76**, 014501 (2007).
- [86] C. Bernard (unpublished).
- [87] C. Aubin, C. Bernard, C. DeTar, J. Osborn, S. Gottlieb, E. B. Gregory, D. Toussaint, U. M. Heller, J. E. Hetrick, and R. Sugar (MILC Collaboration), *Phys. Rev. D* **70**, 114501 (2004).
- [88] C. W. Bernard, *Phys. Rev. D* **62**, 034503 (2000).
- [89] R. Sommer, *Nucl. Phys.* **B411**, 839 (1994).
- [90] A. Bazavov *et al.* (MILC Collaboration), *Proc. Sci., CD09* (2009) 007.
- [91] C. T. H. Davies, E. Follana, I. D. Kendall, G. P. Lepage, and C. McNeile (HPQCD Collaboration), *Phys. Rev. D* **81**, 034506 (2010).
- [92] I. W. Stewart, *Nucl. Phys.* **B529**, 62 (1998).
- [93] R. Casalbuoni, A. Deandrea, N. Di Bartolomeo, R. Gatto, F. Feruglio, and G. Nardulli, *Phys. Rep.* **281**, 145 (1997).
- [94] A. Anastassov *et al.* (CLEO Collaboration), *Phys. Rev. D* **65**, 032003 (2002).
- [95] A. Abada, D. Becirevic, P. Boucaud, G. Herdoiza, J. P. Leroy, A. Le Yaouanc, O. Pene, and J. Rodriguez-Quintero, *Nucl. Phys. B, Proc. Suppl.* **119**, 641 (2003).
- [96] M. C. Arnesen, B. Grinstein, I. Z. Rothstein, and I. W. Stewart, *Phys. Rev. Lett.* **95**, 071802 (2005).
- [97] H. Ohki, H. Matsufuru, and T. Onogi, *Phys. Rev. D* **77**, 094509 (2008).
- [98] J. Bulava *et al.* (ALPHA Collaboration), *Proc. Sci., LATTICE2010* (2010) 303.
- [99] W. Detmold, C.-J. D. Lin, and S. Meinel, *Phys. Rev. D* **85**, 114508 (2012).
- [100] C. Bernard, *Proc. Sci., LAT2007* (2007) 090.
- [101] A. S. Kronfeld, *Nucl. Phys. B, Proc. Suppl.* **129**, 46 (2004).
- [102] M. B. Oktay and A. S. Kronfeld, *Phys. Rev. D* **78**, 014504 (2008).
- [103] W. Kilian and T. Mannel, *Phys. Lett. B* **301**, 382 (1993).
- [104] C. Bernard (MILC Collaboration), *Phys. Rev. D* **65**, 054031 (2002).
- [105] Average of the CDF and LHCb results by the Heavy Flavor Averaging Group <http://www.slac.stanford.edu/xorg/hfag/>.

# **Computer Science & Information Technology**

David C. Wyld  
Natarajan Meghanathan  
Dhinaharan Nagamalai

## **Third International Conference on Computer Science, Engineering & Applications (ICCSEA 2013)**

**Delhi, India. May 2013.**

**Proceedings**

Computer Science Conference Proceedings  
**AIRCC**

ructing a Normalisation Dictionary Methods in Natural Language  
es 421–432, Jeju, Korea.  
Building re-usable dictionary

and Katherine J. Miller, "In-  
ormance Data Compression",  
inimum-Redundancy Codes",  
igital article.  
-486-20097-3.

atic construction of domain-  
38 Issue 10, September 2011,

omatic identification of hyper-  
orghi12.531  
ment Analysis from Massive  
ence on Empirical Methods in  
e Learning EMNLP-CoNLL

nation sciences Institute. Uni-

Messages: Makn Sens a #twit-

# OpenCL programming using Python syntax

Massimo Di Pierro

School of Computing, DePaul University, Chicago IL 60604, USA

## ABSTRACT

We describe *ocl*, a Python library built on top of *pyOpenCL* and *numpy*. It allows programming GPU devices using Python. Python functions which are marked up using the provided decorator, are converted into C99/OpenCL and compiled using the JIT at runtime. This approach lowers the barrier to entry to programming GPU devices since it requires only Python syntax and no external compilation or linking steps. The resulting Python program runs even if a GPU is not available. As an example of application, we solve the problem of computing the covariance matrix for historical stock prices and determining the optimal portfolio according to Modern Portfolio Theory.

## 1. INTRODUCTION

Complying with Moore's Law, the number of transistors on integrated circuits continues to double every 18 months. This has resulted in modern CPUs having multiple computing cores. To date, a state of the art AMD Opteron 8380 hosts 32 computing cores. A Nvidia Tesla GPU K20X hosts 2688 CUDA cores. A modern desktop computer can host multiple CPUs and GPUs. Programming such heterogeneous architectures is a challenge. Traditional paradigms for programming parallel machines are not adequate to handle large number of cores.

Modern programming languages have proposed various solutions. For example Clojure uses transactional memory to simplify multi-threaded programming. Erlang and Go uses lightweight threads that communicate via message passing. Haskell uses a library of distributed data structures (REPL) optimized for different types of architectures. Yet they are not as fast as C code (except possibly for Go), scale but only up to a point, and they are not designed to work on GPUs. Nvidia has proposed the CUDA framework for programming GPUs and it remains the best solution for programming Nvidia devices. Unfortunately it does not work on regular CPUs.

The Kronos Consortium supported by Nvidia, AMD, Intel, and ARM, has developed the Open Common Language framework (OpenCL) which has many similarities with CUDA, but promises more portability and support for different architectures including Intel/AMD CPUs, Nvidia/ATI GPU, and ARM chips such as those used on mobile phones.

Both CUDA and OpenCL programs are divided into two parts. A host program usually written in C or C++ and one or more kernels written in a C-like language. The host program compiles the kernels and runs them on the available devices (the GPUs in the case of CUDA and GPUs or CPUs in the case of OpenCL). The host program also deals with loading and saving operations between the main memory and the computing devices. The OpenCL language is very close to David C. Wyld (Eds) : ICCSEA, SPPR, CSIA, WimoA - 2013  
pp. 57–66, 2013. © CS & IT-CSCP 2013

DOI : 10.5121/csit.2013.3506

C99 while the CUDA dialect is more powerful and, for example, supports templates.

A major complexity in writing CUDA and OpenCL code consists in having to write code inside code (the kernel managed by the host). This process can be somewhat simplified using a different language for the host. For example, the pyCUDA and pyOpenCL libraries created by Andreas Klöckner[3] allow compiling, queuing, and running kernels (in CUDA and OpenCL respectively) from inside a Python program. They are integrated with numpy[4], the Python numeric library.

In this paper we discuss a library called ocl which is built on top of pyOpenCL. It allows writing both the host program and the kernels using the Python languages without loss of performance compared to native OpenCL. ocl consists of two parts:

- A thin layer on top of pyOpenCL which encapsulates the device state (consisting of a context and a task queue).
- A function decorator which, at runtime, converts decorated Python functions into C99 code and uses the OpenCL JIT to compile them.

Python + numpy + pyOpenCL + ocl allow development of parallel programs which run everywhere (CPUs and GPUs), are written in a single language, and do not require any outside compilation, linking, or other building step.

ocl is similar to Cython[8] and Clyther[9] in the sense that it works by parsing the Python code into a Python Abstract Syntax Tree (AST) and then serializing the AST into C99 code. There are differences between Clyther and ocl: The implementation is different. The semantic used to build the kernel body is different and closer to C in the ocl case. Moreover, the ocl implementation is extensible and, in fact, it can be used to generate and compile C code (without the need for OpenCL, like Cython) but it can also be used to generate JavaScript code (to run, for example, in a browser). Javascript code generation is beyond the scope of this paper since its application is in web development, not high performance computing.

In order to explain the syntax of ocl we will consider first the simple example of adding two vectors and then a more complex financial example of computing the correlation matrix for arithmetic return of multiple time series and determine the optimal portfolio according to Modern Portfolio Theory.

## 2. SIMPLE EXAMPLE

In our first example we consider the following Python code which defines two arrays, fills them with random Gaussian floating numbers, and adds them using the available devices.

```
from ocl import Device
import numpy
import random

n = 1000000
a = numpy.zeros(n, dtype=numpy.float32)
b = numpy.zeros(n, dtype=numpy.float32)

for k in xrange(n):
    a[k] = random.gauss(0,1)
```

b[k] = r2

device = De  
a\_buffer = d  
b\_buffer = d

@device.cor

def add(a, b)  
 k = new\_  
 b[k] = a[

program = de  
program.addc  
b = device.re

In this exa

- Lines
- Lines
- them v
- Lines 9
- standard
- Line 11
- Lines 12
- a\_buffe
- Lines 13
- Line 14
- into C9
- Lines 15
- Line 20
- value fu
- Line 23
- Line 24
- function
- Line 25

The progra

python mypro

Notice how

@device.comp

This decorato  
into an Abstra  
write OpenCL  
Supported c  
and continue.

rite code inside  
using a different  
ted by Andreas  
CL respectively)  
meric library.

t allows writing  
of performance

isting of a  
is into C99

ams which run  
uire any outside

the Python code  
' code. There are  
tic used to build  
nplementation is  
ut the need for  
, for example, in  
application is in

e of adding two  
lation matrix for  
ording to Modern

arrays, fills them  
es.

```
device = Device()
a_buffer = device.buffer(source=a, mode=device.flags.READ_ONLY)
b_buffer = device.buffer(source=b)

@device.compiler.define_kernel(a='global:const:ptr_float',
                               b='global:ptr_float')
def add(a, b):
    k = new_int(get_global_id(0))
    b[k] = a[k]+b[k]

program = device.compile()
program.add(device.queue, [n], None, a_buffer, b_buffer)
b = device.retrieve(b_buffer, shape=(n,))
```

In this example:

- Lines 1-3 import the required modules.
- Lines 6-7 define two numpy arrays (a and b) of size n in the main RAM memory and fill them with zeros.
- Lines 9-11 populate the arrays a and b with random Gaussian numbers with mean 0 and standard deviation 1.
- Line 13 determines the available device and incorporates its state.
- Lines 14-15 copies the arrays a and b into the memory of the device represented by a\_buffer and b\_buffer respectively.
- Lines 17-21 define a new kernel called "add".
- Line 17-18 decorate the "add" function to specify it is a kernel and needs to be converted into C99 code and declare the types of the function arguments.
- Lines 19-21 define the body of the kernel.
- Line 20 runs on the device (one instance per thread) and on each thread returns a different value for k.
- Line 23 converts and compiles the kernel, and loads it into the device.
- Line 24 calls the function "add". More precisely it queues [n] threads each calling the function "add" and determines that a\_buffer and b\_buffer are to be passed as arguments.
- Line 25 retrieves the output array b\_buffer from the device and copies it into b.

The program above is run with a single Bash shell command:

```
python myprogram.py
```

Notice how the kernel function "add" is preceded by the decorator

```
@device.compiler.define_kernel(...)
```

This decorator performs introspection of the function at runtime, parses the body of the function into an Abstract Syntax Tree (AST) for later conversion to C99/OpenCL code. This allows us to write OpenCL code using Python syntax.

Supported control structures and commands include def, if..elif else, for ... range, while, break, and continue.

The decorated code is converted at runtime into the following Python AST

```
FunctionDef(name='add',
    args=arguments(args=[Name(id='a'), Name(id='b', ctx=Param())]),
    body=[Assign(targets=[Name(id='k')], value=Call(func=Name(id='new_int'),
        args=[Call(func=Name(id='get_global_id'),
            args=[Num(n=0)])]), Assign(targets=[Subscript(value=Name(id='b'),
                slice=Index(value=Name(id='k')))], value=BinOp(left=Subscript(value=Name(id='a'),
                    slice=Index(value=Name(id='k'))), op=Add(),
                    right=Subscript(value=Name(id='b'),
                        slice=Index(value=Name(id='k'))))), Return(value=Name(id='None'))])
```

(we simplified the AST by omitting irrelevant parameters).

The call to device.compile(...) converts the AST into the following code:

```
__kernel void add(__global const float* a, __global float* b) {
    int k;
    k = get_global_id(0);
    b[k] = (a[k] + b[k]);
}
```

`__kernel` is an OpenCL modifier which declares the function to be a kernel. `__global` and `__local` are used to declare the type of memory storage: global of the device or local of the core, respectively.

One complication is that C99/OpenCL is a statically typed language while Python is only strongly typed. Therefore one needs a way to declare the type of variables using Python syntax. For the function arguments this is done in the decorator arguments. For example:

`a = "global:const:ptr_float"`

is converted into

`__global const float *a`

The type of local variables is defined using the *pseudo-casting* operators `new_int`, `new_float`, etc. For example:

`k = new_int(get_global_id(0))`

is converted into

```
int k;
k = get_global_id(0);
```

Variable types must be declared when first assigned. The converter checks that this is the case.<sup>10</sup>

prevent further and more obscure compile time errors.

In this example, `get_global_id(0)` is an OpenCL function user to identify the rank of the current running instance of the kernel. If one queues  $[n]$  kernel tasks, it will return all values from 0 to  $n-1$ , a different value for each running task. The order in which queued tasks are executed is not guaranteed.

The return type of a function is determined automatically but one must return a variable (or `None`), not an expression.

Other variable types and pointers can be defined and manipulated using the syntax shown in the following table:

ocl	C99/OpenCL
<code>x = new_type(...)</code>	<code>type x = ...;</code>
<code>x = new_ptr_type(...)</code>	<code>type *x = ...;</code>
<code>x = new_ptr_ptr_type(...)</code>	<code>type **x = ...;</code>
<code>ADDR(x)</code>	<code>&amp;x</code>
<code>REFD(x)</code>	<code>*x</code>
<code>CAST(prt_type,x)</code>	<code>(type*)x</code>

OpenCL provides special types of variables to take advantages of vector registries. For example `float4` can store four floating point numbers called `x`, `y`, `z`, and `w`. These special vector types can be used from ocl as follows:

```
q = new_float4((0.0,0.0,0.0,0.0))
q.x = 1.0
```

Notice that the goal of ocl is not to translate arbitrary Python code into C99/OpenCL but to allow the use of Python syntax to write OpenCL kernels and functions. Only C99/OpenCL variable types are allowed, not Python types such as lists and dictionaries. This may be supported in future versions of ocl. Moreover all functions called in the Python code which constitutes the body of the add function are intended to be OpenCL functions not Python functions. While Python code is normally garbage collected, the decorated code runs on the device as C code and it is not garbage collected therefore explicit memory management may be necessary:

```
@device.compiler.define_kernel...
def some_kernel(...):
    d = new_ptr_float(malloc(size))
    ...
    free(d)
```

which would be converted into:

```
__kernel void some_kernel(...):
    float* d;
    d = (float*)malloc(size);
    ...
```

at this is the case to

```

    free(d);
}

```

One can define multiple kernels within the same program and call them when needed. One can use the `@device.compiler.define(...)` decorator to define other functions to be executed on the device. They will be visible and callable by the kernels, but they will not be callable from the host program.

One can pass constants from the Python context to the OpenCL context:

```
device.compile(..., constants = dict(n = m))
```

where n is a constant in the kernel and m is a variable in the host program.

Additional C files can be included using the syntax

```
device.compile(..., includes = ['#include <math.h>'])
```

where includes is a list of #include statements.

### 3. FINANCIAL APPLICATION EXAMPLE

In the following example, we solve the typical financial problem of analyzing multiple time series such as stock prices,  $p[k,t]$  (simulated stock prices). For each time series k we compute the arithmetic daily returns,  $r[k,t]$ , and the average returns,  $\mu[k]$ . We then compute the covariance matrix,  $\Sigma[i,j]$ , and the correlation matrix,  $\text{cor}[i,j]$ . We use different kernels for each part of the computation.

Finally, to make the application more practical, we use Modern Portfolio Theory [10] to compute a tangency portfolio which maximizes the Sharpe ratio, *i.e.* return/risk under the assumption of normal returns:

Here  $\mu$  is the vector of average returns ( $\mu$ ),  $\Sigma$  is the covariance matrix ( $\Sigma$ ), and  $\rho$  is the input risk free interest rate. The tangency portfolio is identified by the vector  $x$  (aka array  $x$  in the code) whose terms indicate the amount to be invested in each stock (must add up to one dollar). We perform this maximization on the CPU to demonstrate integration with the numpy Linear Algebra package.

We use the symbols i and j to identify the stock time series, and the symbol t for time (for daily data t is a day). n is the number of stocks and m is the number of trading days.

We use the canvas[6] library, based on the Python matplotlib library, to display one of the stock price series and the resulting correlation matrix. Below is the complete code. The output from the code can be seen in fig. 1.

```

from ocl import Device
from canvas import Canvas
import random

```

```

import numpy
from math import exp

1. One can
ted on the
'm the host

n = 1000 # number of time series
m = 250 # number of trading days for time series
p = numpy.zeros((n,m), dtype=numpy.float32)
r = numpy.zeros((n,m), dtype=numpy.float32)
mu = numpy.zeros(n, dtype=numpy.float32)
cov = numpy.zeros((n,n), dtype=numpy.float32)
cor = numpy.zeros((n,n), dtype=numpy.float32)

for k in xrange(n):
    p[k,0] = 100.0
    for t in xrange(1,m):
        c = 1.0 if k==0 else (p[k-1,t]/p[k-1,t-1])
        p[k,t] = p[k,t-1]*exp(random.gauss(0.0001,0.10))*c

device = Device()
p_buffer = device.buffer(source=p, mode=device.flags.READ_ONLY)
r_buffer = device.buffer(source=r)
mu_buffer = device.buffer(source=mu)
cov_buffer = device.buffer(source=cov)
cor_buffer = device.buffer(source=cor)

@device.compiler.define_kernel(p='global:const:ptr_float',
                               r='global:ptr_float')
def compute_r(p, r):
    i = new_int(get_global_id(0))
    for t in range(0,m-1):
        r[i*m+t] = p[i*m+t+1]/p[i*m+t] - 1.0

@device.compiler.define_kernel(r='global:ptr_float',
                               mu='global:ptr_float')
def compute_mu(r, mu):
    i = new_int(get_global_id(0))
    sum = new_float(0.0)
    for t in range(0,m-1):
        sum = sum + r[i*m+t]
    mu[i] = sum/(m-1)

@device.compiler.define_kernel(r='global:ptr_float',
                               mu='global:ptr_float', cov='global:ptr_float')
def compute_cov(r, mu, cov):
    i = new_int(get_global_id(0))
    j = new_int(get_global_id(1))
    sum = new_float(0.0)
    for t in range(0,m-1):
        sum = sum + r[i*m+t]*r[j*m+t]
    cov[i*n+j] = sum/(m-1)-mu[i]*mu[j]

```

time series  
compute the  
covariance  
part of the

to compute  
umption of

is the input  
n the code)  
dollar). We  
ear Algebra

e (for daily

of the stock  
ut from the

```

@device.compiler.define_kernel(cov='global:ptr_float',
                               cor='global:ptr_float')
def compute_cor(cov, cor):
    i = new_int(get_global_id(0))
    j = new_int(get_global_id(1))
    cor[i*n+j] = cov[i*n+j] / sqrt(cov[i*n+i]*cov[j*n+j])

program = device.compile(constants=dict(n=n,m=m))

q = device.queue
program.compute_r(q, [n], None, p_buffer, r_buffer)
program.compute_mu(q, [n], None, r_buffer, mu_buffer)
program.compute_cov(q, [n,n], None, r_buffer, mu_buffer, cov_buffer)
program.compute_cor(q, [n,n], None, cov_buffer, cor_buffer)

r = device.retrieve(r_buffer,shape=(n,m))
mu = device.retrieve(mu_buffer,shape=(n,))
cov = device.retrieve(cov_buffer,shape=(n,n))
cor = device.retrieve(cor_buffer,shape=(n,n))

points = [(x,y) for (x,y) in enumerate(p[0])]
Canvas(title='Price').plot(points).save(filename='price.png')
Canvas(title='Correlations').imshow(cor).save(filename='cor.png')

rf = 0.05/m # input daily risk free interest rate
x = numpy.linalg.solve(cov,mu-rf) # cov*x = (mu-rf)
x *= 1.00/sum(x) # assumes 1.00 dollars in total investment
open('optimal_portfolio','w').write(repr(x))

```

Notice how the memory buffers are always 1-dimensional therefore the i,j indexes have to be mapped into a 1D index  $i*n+j$ . Also notice that while kernels `compute_r` and `compute_mu` are called  $[n]$  times (once per stock k), kernels `compute_cov` and `compute_cor` are called  $[n,n]$  times, once per each couple of stocks i,j. The values of i,j are retrieved by `get_global_id(0)` and `(1)` respectively.

In this program we have defined multiple kernels and complied them at once. We call one kernel at the time to make sure that the call to the previous kernel is completed before running the next one.

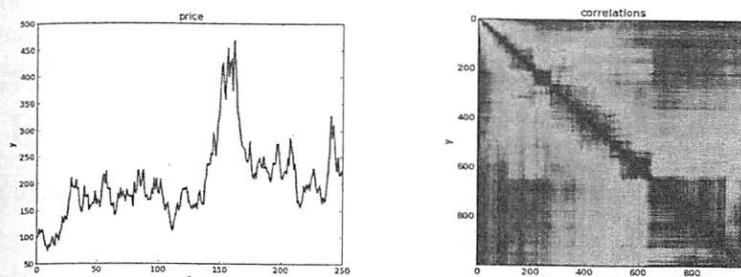


Figure 1: The image on the left shows one of the randomly generated stock price histories. The image on the right represents the computed correlation matrix. Rows and columns corresponds to stock returns and the color at the intersection is their correlation (red for high correlation and blue for no correlation). The resulting shape is an artifact of the algorithm used to generate random data.

#### 4. OCL WITHOUT OpenCL

ocl can be used to generate C99 code, compile it, and run it without OpenCL. In this case the code always runs on the CPU and not on the GPU. Here is a simple example:

```
from ocl import Compiler
c99 = Compiler()

@c99.define(n='int')
def factorial(n):
    output = 1
    for k in range(1, n + 1):
        output = output * k
    return output

xes have to be
> compute_mu are
led [n,n] times,
l_id(0) and (1)

call one kernel
running the next
```

```

compiled = c99.compile()
print(compiled.factorial(10))
assert compiled.factorial(10) == factorial(10)
```

In this example, we have replaced device.compiler with c99 = Compiler() since “device” is an OpenCL specific concept. We have been careful to engineer the function “factorial” so that it can run both in Python (factorial) and compiled in C99 (compiled.factorial). This is not always possible but it possible for functions that only call mathematical libraries, such as our factorial function. Notice that c99.compile() requires the presence of gcc installed instead of the OpenCL JIT.

#### 5. CONCLUSIONS

In this paper we provide an introduction to ocl, a library that allows writing OpenCL programs and kernels using exclusively Python syntax. ocl is built on top of pyOpenCL[2], meta[7], and numpy[4] (the Python Scientific Libraries). The Python code is converted to C99/OpenCL and compiled at run-time. It can run on any OpenCL compatible device including ordinary Intel/AMD CPUs, Nvidia/ATI GPUs, and ARM CPUs with the same performance as native OpenCL code.

Our approach does not necessarily simplifies the algorithms but it makes them more readable, portable, and eliminates external compilation and linking steps thus lowering the barrier of entry to GPU programming. It does not eliminate the need to understand OpenCL semantic but allows the use of a simpler syntax.

As an example of application we considered the computation of the covariance and correlation matrices from financial data series, in order determine the optimal portfolio according to Model Portfolio Theory [10].

We plan to extend ocl to support a richer syntax, generate CUDA as well as OpenCL code, and provide better debugging information. Moreover, since Python code can easily be serialized and deserialized at runtime, we plan to improve the library to allow distributed computations where the code is written once (in Python), distributed, then compiled and ran on different worker nodes using heterogenous devices.

#### ACKNOWLEDGEMENTS

I thank Andreas Klöckner for pyOpenCL and the excellent documentation.

#### REFERENCES

- [1] Lindholm, Erik, et al. "NVIDIA Tesla: A unified graphics and computing architecture." *Micro*, IEEE 28.2 (2008): 39-55.
- [2] Munshi, Aaftab. "OpenCL: Parallel Computing on the GPU and CPU." SIGGRAPH, Tutorial (2008).
- [3] Klöckner, Andreas, et al. "Pycuda and pyopencl: A scripting-based approach to gpu run-time code generation." *Parallel Computing* 38.3 (2012): 157-174.
- [4] Oliphant, Travis E. *A Guide to NumPy*. Vol. 1. USA: Trelgol Publishing, 2006.
- [5] <https://github.com/mdipierro/ocl>
- [6] <https://github.com/mdipierro/canvas>
- [7] <http://rossross.github.com/Meta/html/index.html>
- [8] Behnel, Stefan, et al. "Cython: The best of both worlds." *Computing in Science & Engineering* 13.2 (2011): 31-39.
- [9] <http://rossross.github.com/Clyther/>
- [10] Markowitz, Harry M. "Foundations of portfolio theory." *The Journal of Finance* 46.2 (2012): 469-477.

## QCL: OpenCL meta programming for lattice QCD

---

**Massimo Di Pierro**

*School of Computing - DePaul University - Chicago*  
*E-mail:* [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu)

QCL is an experimental Python library that generates and executes real time OpenCL code for lattice Quantum Field Theory computations. QCL allows the user to describe arbitrary gauge and fermionic actions in terms of path shapes (links, plaquettes, staples, etc.) and generates optimized code to perform matrix multiplications along the paths. The generated code is then utilized by the algorithms, including the heatbath and the  $D$ . It can generate code for any improved action that can be described as a sum of paths. QCL works for arbitrary  $SU(N_c)$  gauge groups and number of dimensions. It includes the BLAS/LaPack libraries and plotting capabilities.

POS(LATTICE 2013)043

*31st International Symposium on Lattice Field Theory - LATTICE 2013*  
*July 29 - August 3, 2013*  
*Mainz, Germany*

## 1. Introduction

In the last 10 years GPU programming has become increasingly important for High Performance Computing and Lattice QCD in particular, yet most of the focus has been on implementations of SU(3) gauge field theories in 4 dimensions and highly optimized CUDA kernels for specific QCD algorithms [1, 2, 3, 4, 5, 6]. Cardoso *et al.* [7] have tackled the problem for arbitrary  $SU(N_n)$  but only in 4D and only for the Wilson Gauge Action. In this paper we present a different and more general approach.

QCL is a Python library that uses meta-programming techniques to generate and run OpenCL [8] code. QCL allows the developer to program using the Python language and exposes an Application Program Interface (API) to describe actions and algorithms. The algorithms, however, are not yet implemented but instead generated in real time based on the description of the action. Actions are described by a gauge group, a set of path shapes (closed paths for gauge terms, open paths for smearing operators and fermion terms), and their symmetries. The OpenCL code for the algorithms is then generated as needed in real time. QCL can generate code for improved actions as well, as long as they can be described as sums of paths. Some of the supported algorithms include the heathbath, the  $\mathcal{D}$ , the Minimum Residual (MinRes), and the Stabilized Bi-Conjugate Inverter (BiCGStab). The heathbath supports arbitrary  $SU(N_c)$  gauge groups and number of dimensions.  $\mathcal{D}$  also supports arbitrary  $SU(N_c)$  gauge groups, Wilson and Staggered types of fermions.

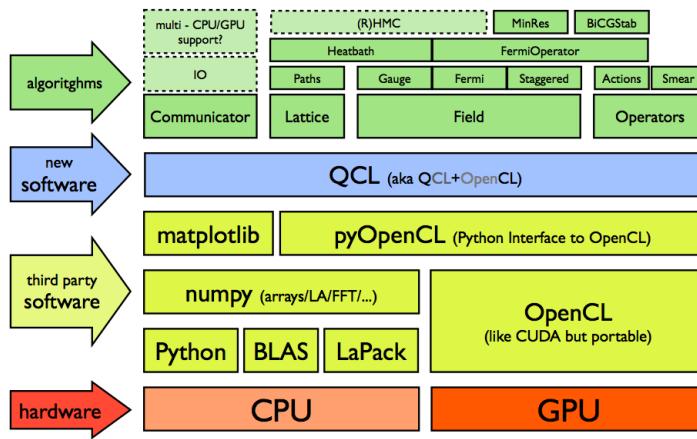
Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms including CPUs, GPUs, DSPs, and virtual machines (LLVM). Like CUDA, OpenCL implements a form of task-based parallelism and includes a programming language derived from C99 for writing computing kernels. One kernel and a unit of data constitutes a task. OpenCL programs consist of two parts: a host program and a collection of kernels. The host program loads data from RAM, assigns data to kernels, and queues the tasks for execution on the available devices. OpenCL includes a just-in-time compiler which allows the host program to generate and compile kernels at run-time. Unlike CUDA, which is mostly an initiative of a single vendor, OpenCL is an open standard maintained by the non-profit technology consortium Kronos Group, adopted and supported by Nvidia, AMD, Intel, ARM, Apple, Qualcomm, and others. Since kernels are written in OpenCL, QCL code presents no overhead compared to native OpenCL code.

In QCL the host program is written in Python using the pyOpenCL library [9], the numpy library (which provides efficient arrays and BLAS/LaPack algorithms), and the matplotlib library (for plotting).

## 2. Architecture Overview

QCL comprises of the following core classes (Fig 1):

- `Lattice`, which stores information about lattice size and topology.
- `Field`, which can store arbitrary data at the lattice sites.
- `ComplexScalarField`, which stores Complex numbers at lattice sites.
- `GaugeField`, a  $SU(N_c)$  gauge field in  $d$ -dimensions.



**Figure 1:** Architectural overview of QCL.

- WilsonField, which stores  $N_c$  spinors at each lattice site.
- StaggeredField, which stores  $N_c$  Complex numbers at each lattice site.
- GaugeAction, a collection of closed path shapes and their relative coefficients.
- GaugeSmearOperator, a collection of open path shapes, and their relative smearing coefficients.
- FermiOperator, a collection of open path shapes, and their relative coefficients and spin structure. A  $\not{D}$  operator is a FermiOperator.

All QCL programs start by importing from the `qcl` module and defining a lattice instance. Note that one can define multiple lattices in the same program. Here is an example of a  $6^4$  lattice:

```
1 from qcl import *
2 lattice = Q.Lattice([6,6,6,6])
```

Given a lattice one can define multiple fields on each lattice. Here is a  $SU(2)$  gauge field:

```
1 U = lattice.GaugeField(2)
```

Fields live in RAM until an algorithm copies them onto a device (for example, a GPU). Gauge fields can be initialized using `set_cold`, `set_hot` (implemented in OpenCL) and can be accessed element by element:

```
1 U.set_cold()
2 print U[(0,0,0,0),0]
3 print U[(0,0,0,0),0,0,0]
```

While in RAM, fields are stored as `numpy` arrays.

In QCL we distinguish between a path (which has a shape and an origin) and a path shape (which lacks an origin and may be translated and symmetrized). Path shapes on the lattice are represented by Python tuples. Sets of paths are represented as lists of tuples. The `bc_symmetrize`

*qcl*

method can perform the hyper-octahedral symmetrization ( $BC_n$ ) of any set of paths. The `derive_paths` method removes a link from a set of path shapes thus obtaining the corresponding staples:

```
1 plaquette = (1,2,-1,-2)
2 print bc_symmetrize(plaquette)
3 print remove_duplicates(derive_paths(bc_symmetrize(plaquette),2))
```

In the example below we define a scalar field  $\rho$  and set it equal to the trace of the link product along a given path, for each lattice site:

```
1 rho = lattice.Field([1])
2 rho.set_link_product(U, [(1,2,3,4,-1,-2,-3,-4)])
3 print rho*rho
```

A gauge action can be constructed by adding terms (in the example below a single plaquette), which are automatically symmetrized. The `heatbath` of the `action` object generates OpenCL code. The `run` method runs the code. Here is an example for the simple Wilson gauge action:

```
1 action = GaugeAction(U).add_term((1,2,-1,-2))
2 code = action.heatbath(beta=4.0,n_iter=100)
3 code.run()
4 print U.average_plaquette()
```

The `add_term` method can be chained to construct more complex gauge actions or, similarly, `GaugeSmearOperators`. QCL also understands named smearing algorithms including *fat3*, *fat5*, *fat7*, *lepage*, and *hisq*:

```
1 V = clone(U)
2 S = GaugeSmearOperator(U).add_term([1],1.0).add_term([2,1,-2],0.1)
3 V.set_smeread(S,reunitarize=4)
4 V.set_fat(U,'fat7+lepage')
5 V.set_hisq(U)
```

The simplest type of fermionic field is the `FermiField` which stores  $N_{spin} \times N_c$  complex numbers per lattice site. In the example below we define one with 4 spin components, clone it, and set one of the components to 1:

```
1 phi = lattice.FermiField(4,U.nc)
2 psi = clone(phi)
3 phi[(0,0,3,3),0,0] = 1.0
```

Fermionic operators are built similarly to gauge actions using the `add_term` method but with an additional parameter, the Gamma structure of the term. QCL also understands named terms such as `add_clover4d_terms`:

```
1 kappa = 0.112
2 D = FermiOperator(U).add_diagonal_term(1.0)
3 for mu in (1,2,3,4):
4     D.add_term(kappa*(I-G[mu]), [(mu,)])
5     D.add_term(kappa*(I+G[mu]), [(-mu,)])
6 D.add_clover4d_terms(c_SW = 0.1)
```

Notice that `G[mu]` is a Gamma matrix and `kappa*(I-G[mu])` is computed only once at the Python level. `D.add_term(...)` then generates the OpenCL code and only considers non-zero terms. It does not generate code for terms that have a known zero coefficient. This is pictorially represented in Fig 2.

Here are examples of multiplication by  $\mathcal{D}$  and by its inverse:

$$\psi = \mathcal{D}\phi \quad \text{and} \quad \phi = \mathcal{D}^{-1}\psi \quad (2.1)$$

*qcl*

---

```
1 psi.set(D,phi)
2 phi.set(invert_minimum_residue,D,psi)
```

In the example below we make a meson propagator by looping over the 4 spin components and the `U.nc` color components for the source. For each source we use the `invert_minimum_residue` method to compute the sink and the `make_meson` operator to contract them. Notice that the first and last arguments of `make_meson` are fermions, while the second argument is an arbitrary spin structure:

```
1 prop = lattice.ComplexScalarField()
2 for spin in range(4):
3     for color in range(U.nc):
4         psi = lattice.FermiField(4,U.nc)
5         psi[(0,0,0,0),spin,color] = 1.0
6         phi.set(invert_minimum_residue,D,psi)
7         prop += make_meson(phi,I,phi)
8 prop_fft = prop.fft()
9 prop_t = [(t,math.log(prop_fft[t,0,0,0].real)) for t in range(lattice.shape[0])]
10 Canvas().plot(prop_t).save('meson.prop.png')
```

QCL includes operators to contract arbitrary mesons and baryons. In the above example we also used the `numpy fft` function to compute the propagator in momentum space, projected out the zero momentum component, and made a plot using the `Canvas` library, which is built into QCL.

Below we use `Canvas` to make a 2D slice of the (0,0) component of a `psi` field:

```
1 chi = psi.slice((0,0),(0,0))
2 Canvas().imshow(chi).save('fermi.propagator.png')
```

QCL also supports Staggered fields:

```
1 phi = lattice.StaggeredField(U.nc)
2 psi = clone(phi)
3 phi[(0,0,3,3),0] = 1.0
```

and they can be multiplied by `FermiOperators` using the same syntax as the Wilson-like fields:

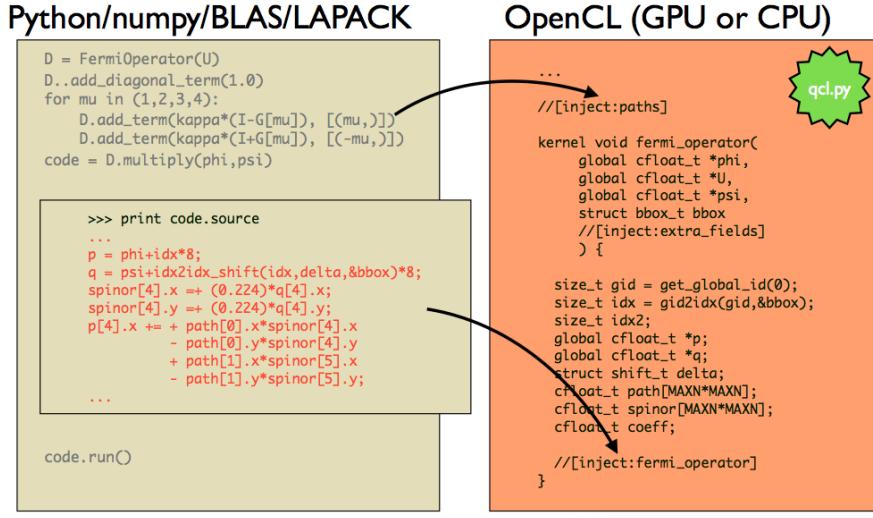
```
1 kappa = 0.112
2 D = FermiOperator(U).add_staggered_action(kappa = 0.112)
3 psi.set(D,phi)
4 psi.set(invert_minimum_residue,D,phi)
```

### 3. Limitations

At this time QCL has the following limitations, which we are working to overcome.

First, only one node and one device can be supported at a time. This means QCL cannot take advantage of multiple GPUs because it cannot split a lattice over multiple devices. The generated OpenCL is flexible enough to be able to access and loop over a subset of the lattice, but QCL lacks the code to synchronize memory buffers across devices. We are working to integrate QCL with MPI to overcome this limitation.

Second, QCL currently performs many unnecessary transfers between the host and the device. Most linear time operations, including scalar products, are performed on the host using `numpy`, while operations involving products of color matrices are performed on the device. This means that each application of  $\mathcal{D}$  involves transfers of data. This is unnecessary and can be eliminated by implementing all scalar products in OpenCL.



**Figure 2:** Example of code generation in QCL. The left pane shows Python code. The right pane shows an OpenCL template. The comments are replaced by generated code at runtime. The code is then compiled on demand.

QCL also lacks converters to and from existing file formats. In a future version of QCL we plan to support the FermiQCD, ILDG, and MILC formats.

QCL's fourth limitation is in its optimizations. Currently QCL performs the following optimizations when generating OpenCL code: 1) It groups together all paths that have the same spin structure. 2) It only generates code for those color/spin components that have a non-zero coefficient. 3) If two paths start with the same product of links, the product of links is cached and reused. Despite these optimizations, QCL is not smart enough to realize that, for example, a 5 staple includes a 3 staple and should therefore store the result for all 3 staples before computing 5 staples. The complexity here is twofold: determining all possible overlaps of paths is a problem of exponential complexity, and reusing sub-paths starting at a different origin requires additional memory and therefore a tradeoff of CPU usage vs memory. It is well known that GPUs are memory bound, so it is usually preferable to duplicate computation rather than cache more vectors. We plan to study this further.

Finally, QCL supports single precision only. We plan to support double and mixed precions in the future.

#### 4. Conclusions and Outlook

In this paper we present an experimental approach to OpenCL metaprogramming for QFT and QCD. In our library QCL, we describe actions in terms of path shapes (links, plaquettes, staples, long staples, etc.) for arbitrary  $SU(N_x)$  gauge groups in D-dimensions. QCL then writes the corresponding OpenCL code. Thus far our main goal was correctness. We have tested our code against FermiQCD, and we believe it is working correctly. Some optimizations are implemented,

but several bottlenecks remain in the system, specifically for the fermionic sector. At this time we are working to eliminate them.

The code is hosted in GitHub [10].

## References

- [1] Nvidia, C. U. D. A. “Compute unified device architecture programming guide.” (2007).
- [2] Babich, Ronald, Michael A. Clark, and Balint Jo. “Parallelizing the QUDA library for multi-GPU calculations in lattice quantum chromodynamics.” High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for. IEEE, 2010.
- [3] Bach, Matthias, et al. “Lattice QCD based on OpenCL.” Computer Physics Communications (2013).
- [4] Philipsen, Owe, et al. “Lattice QCD using OpenCL.” XXIX International Symposium on Lattice Field Theory. 2011.
- [5] Demchik, Vadim, and Natalia Kolomoyets. “QCDGPU: open-source package for Monte Carlo lattice simulations on OpenCL-compatible multi-GPU systems.” arXiv preprint arXiv:1310.7087 (2013).
- [6] Bonati, Claudio, et al. “QCD simulations with staggered fermions on GPUs.” Computer Physics Communications 183.4 (2012): 853-863.
- [7] Cardoso, Nuno, and Pedro Bicudo. “Generating SU ( $N_c$ ) pure gauge lattice QCD configurations on GPUs with CUDA.” Computer Physics Communications (2012).
- [8] Munshi, Aaftab. “The opencl specification.” Khronos OpenCL Working Group 1 (2009): 11-15.
- [9] Klöckner, Andreas, et al. “PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation.” Parallel Computing 38.3 (2012): 157-174.
- [10] <https://github.com/mdipierro/qcl/>

## The new “Gauge Connection” at NERSC

PoS (LATTICE 2013) 427

**Massimo Di Pierro\***

*School of Computing - DePaul University - Chicago*  
*E-mail:* [mdipierro@cs.depaul.edu](mailto:mdipierro@cs.depaul.edu)

**James Hetrick**

*University of the Pacific*  
*E-mail:* [jhetrick@pacific.edu](mailto:jhetrick@pacific.edu)

**Shreyas Cholia**

*Lawrence Berkely Laboratory*  
*E-mail:* [scholia@lbl.gov](mailto:scholia@lbl.gov)

**James Simone**

*Fermi National Accelerator Laboratory*  
*E-mail:* [simone@fnal.gov](mailto:simone@fnal.gov)

**Carleton DeTar**

*University of Utah*  
*E-mail:* [detar@physics.utah.edu](mailto:detar@physics.utah.edu)

In this paper we present a new web interface (NERSC3) for the Gauge Connection, one of the largest repositories of lattice gauge configurations in the US. The Gauge Connection is hosted at the National Energy Research Scientific Computing Center (NERSC), and stores more than 16TB of data using the High Performance Storage System (HPSS). The new interface allows users to search, download, and view statistics on lattice QCD ensembles hosted at NERSC. It also aggregates metadata from other major lattice archives (ILDG), and allows one to search the metadata in a single place. Additionally, users can tag, annotate, and bookmark the ensembles (<http://qcd.nersc.gov>).

*31st International Symposium on Lattice Field Theory - LATTICE 2013*  
*July 29 - August 3, 2013*  
*Mainz, Germany*

---

\*Speaker.

## 1. Introduction

The Gauge Connection is one of the most popular repositories of lattice QCD ensembles. It has been operated by the National Energy Research Scientific Computing Center [1] (NERSC) since 1998, and currently stores over 16 Terabytes of data. The popularity of the Gauge Connection can be attributed to the amount of data it stores, and its easy-to-use web interface which lowers the barrier of entry for lattice QCD researchers.

At the core of the Gauge Connection architecture is the High Performance Storage System [2] (HPSS) data archive. The HPSS system provides a tape archival storage library for the QCD data. The content of the archive is exposed using an FTP interface in conjunction with a CGI-based HTTP download service.

The original web interface consisted of static HTML files linked against this CGI service that would pull data from the HPSS archive. In 2011 the web interface underwent a major revision and the static pages were replaced by dynamically generated pages. The content of the pages comprised of free text mined from the original HTML files, which was made editable in a wiki-like interface. The pages were augmented with metadata obtained from the file catalog itself and stored in a database. The new interface allowed users to search the data more easily, comment on the data, and download ensembles in batch.

In this paper we present a new version of this service which we will refer to as NERSC3. Apart from user interface changes, the main functional change is an expanded search capability which allows users to search both local and remote repositories of ensembles.

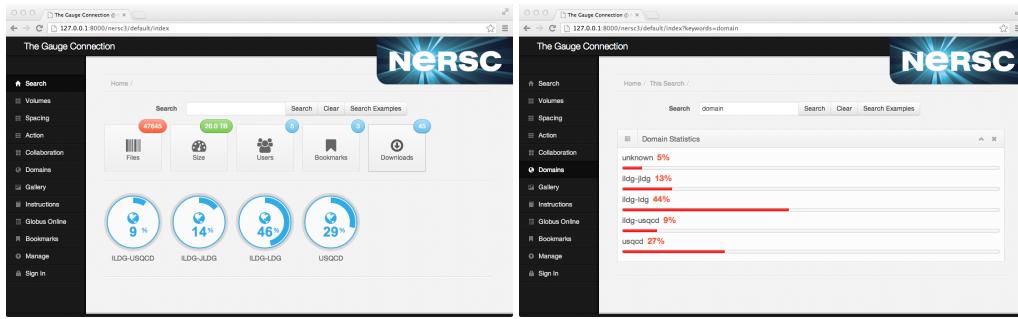
Lattice QCD physicists formed the International Lattice Data Grid (ILDG) [3] collaboration in 2000, and developed protocols and infrastructures to annotate and share lattice QCD ensembles. They created a file format to store gauge ensembles along with standardized metadata, and a network of ILDG metadata catalogs and file catalogs. This enabled searching against the ensembles and their metadata, and the ability to download the files.

The ILDG infrastructure is powerful but compared to the NERSC archive, it presents two major barriers of entry: 1) By design, the metadata catalogs do not inter-operate but provide a common search API; therefore search is decentralized. 2) The file catalogs are built on top of GridFTP [5] and can only be accessed using ILDG-tools, and only after obtaining a valid X509 grid certificate.

In our latest revision of the NERSC archive we have been addressing the first issue. The new system periodically connects to all of the ILDG metadata catalogs and copies the ensemble metadata to a local database. This allows a visitor of the site to search for both local ensembles and remote ILDG ensembles in one place. While the local data can be retrieved directly from the web site or via the provided batch download script, remote data is only referenced, and the user must use the ILDG tools to query the remote file catalog and use GridFTP to download the data.

Moreover, conversion to the ILDG data format has been slow: the bulk of the data is stored in other formats, e.g. NERSC or MILC while only a fraction is stored in ILDG format. Additionally, one of the key improvements to NERSC3 includes the ability to convert file formats post-download.

We have designed the new website to provide additional capabilities beyond data access, including wiki capabilities to annotate the data, link external work derived from the data (derived



**Figure 1:** The main web page for `qcd.nersc.gov` allows browsing and searching for gauge ensembles (left) and visualizing statistical data about the ensembles (right).

data, software, tutorials, and publications), and bookmark interesting data. The new system keeps usage statistics for analytics purposes.

Figure 1 and figure 2 show some screenshots from NERSC3.

## 2. Architecture Overview

The Gauge Connection is hosted at NERSC on the High Performance Storage System (HPSS). HPSS is a hierarchical tape storage system designed to store multiple petabytes which be accessed using a custom command line client (HSI), a C API, an FTP interface, a parallel FTP interface, as well as a GridFTP interface. The latter can also be accessed using Globus Online, a service that enables scheduled data transfers between GridFTP endpoints.

NERSC3 periodically performs directory dumps of the HPSS tapes and reproduces the folder structure in the database. NERSC3 has no direct access to the files because staging them from the tape archive is expensive. Yet it can infer some information about the content of the files from the names of the folders, the names of the files, and their sizes. Most of the files stored at NERSC are produced by the MILC collaboration which is one of the major sources of public gauge ensembles in the world. The MILC collaboration adopts a practical naming convention for the file names which include information about the lattice volume, the kappa value, and the quark masses (for dynamical ensembles) and a progressive configuration counter. NERSC3 can link this information to additional metadata obtained from other sources which is inserted in the system manually e.g. links to papers describing the algorithm used to generate each ensemble.

The NERSC3 database stores information about two types of ensembles: the ensembles stored at NERSC described above and ILDG ensembles stored at other locations.

The ILDG infrastructure consists of the following components:

- A description of metadata which accompanies each ensemble. This includes name of the group/collaboration, description of the action and algorithms used to generate it, simulation parameters (lattice volume, lattice spacing, dynamical quark masses) and links to relevant papers. The information is encoded in a custom XML-based schema.

- A file format for storing data along with metadata. ILDG uses a custom encoding protocol similar to TAR, called LIME.
- A metadata catalog service which exposes SOAP services to query the catalog for local ensembles, their metadata, and their file content.
- A file catalog service which exposes additional SOAP services, protected by WS-Security, to convert logical filenames into actual URLs for downloading the files.

ILDG is comprised of 5 realms: CSSM (Australia), LDG (Europe), JLQCD (Japan), UKQCD (UK), USQCD (US). Each realm runs at least one metadata catalog service and one file catalog service.

In ILDG each ensemble is identified by a Unique Resource Identifier (URI) starting with the `mc:` prefix. Each logical file name is identified by a URI starting with the `lfn:` prefix. The actual file names are not URIs but they are URLs. They usually have one of the following prefixes: `http:`, `ftp:`, `https:`, `ftps:`, and `srm:`. `srm:` is a GridFTP based web service protocol which operates over HTTP but is designed to act as a referrer for multiple endpoints and effectively load-balances transfers over multiple nodes, allowing transfers to scale. It is very similar in scope to the BitTorrent protocol, i.e. it splits data transfers over multiple connections. The SRM protocol is supported by the *dCache* [7] tape storage system used at Fermilab.

The metadata catalog services are public. NERSC3 periodically connects to these services and downloads a complete list of available ensembles, their metadata, and their logical filenames. This information is stored in a local database. In this way, when using the NERSC3 web interface one can search both the local ensembles and all the ensembles made available via ILDG.

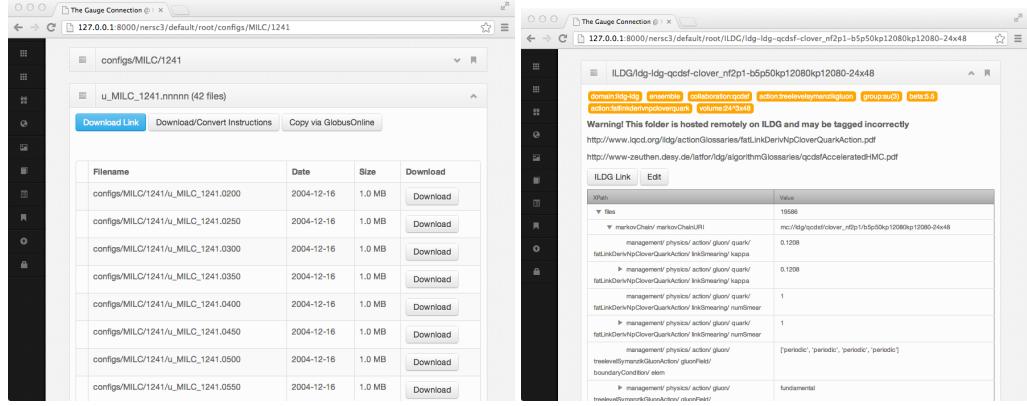
The ILDG file catalogs are not public. This means that the conversion between logical filename to actual URLs cannot be performed without a valid GridFTP certificate, and each realm has its own policies and procedures for assigning certificates. Accessing the actual URLs also requires a certificate. Thus, while NERSC3 allows for searching of remote data and can point the user to a specific remote ensemble, it does not help the user with accessing the data itself.

Only the data stored at NERSC is truly public in the sense that it can be accessed by anybody without explicit permission from an ILDG certificate authority.

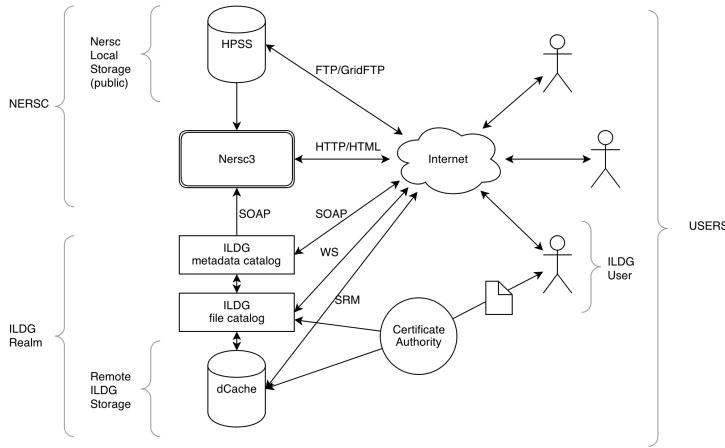
Figure 3 shows an overview of the architecture described here.

In NERSC3 tags are associated with ensembles, rather than individual files. Each tag is divided into two parts separated by a colon (`:`). e.g. `volume:24^3x48`. The first part of the tag (in this example `volume`) is a key, while the second part (in this example `24^3x48`) is a value. Tags are all optional and key names are not dictated by the system but chosen by the users. Some keys may not have a corresponding value. More examples of tags include `beta:2.1`, `collaboration:milc`, and `flavor:2+1`. When clicking on a tag, the user is presented with a list of ensembles sharing the same tag.

Users can search by key (any key) and they are presented with a list of all possible values for that key along with statistical information. They can search by tag (key:value) and they are presented with a list of ensembles with the corresponding tag. They can search for multiple tags and they are presented with a list of ensembles that have all the requested tags. Here are some examples of search strings:



**Figure 2:** For each local ensemble, the NERSC3 interface can list the ensemble files (left). For remote ensembles, the NERSC3 interface can list the ILDG metadata and the FLN URI.



**Figure 3:** Overview of the NERSC3 and ILDG architecture.

```

1 volume
2 volume:12^3x12
3 volume is 12^3x12
4 volume is 12^3x12 and domain is usqcd and collaboration is milc

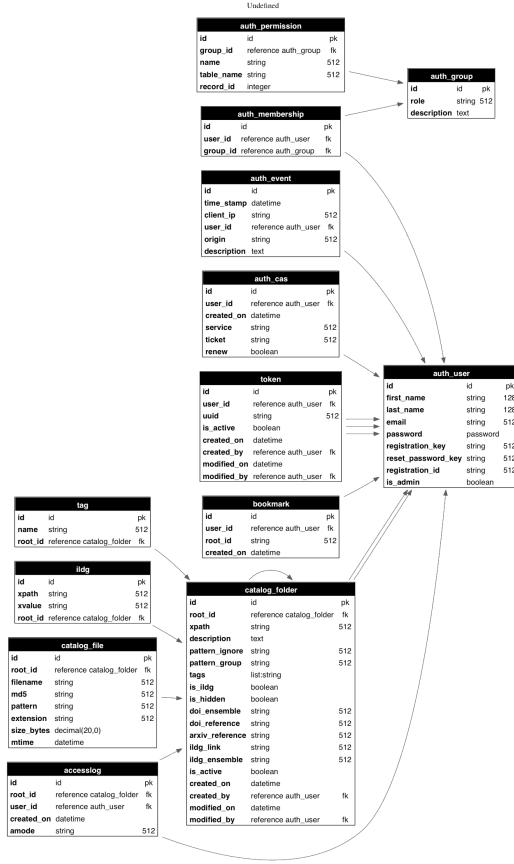
```

In this case the domain “usqcd” refers to the set of ensembles that are hosted at NERSC as opposed to the ILDG ensembles.

The search string understands the “is” and “and” english words. A search string can also be the name of an ensemble.

Figure 4 shows the database architecture of the NERSC3 application. It includes tables to store users, groups, memberships and permissions. We implement role based access control for administrators. It includes a table `catalog_folder` (which represents an ensemble) and a table `catalog_file` which stores links to individual files (the files themselves are in NERSC HPSS or ILDG). `ildg metedata`, `accesslogs`, `bookmarks`, and `tags` reference the `catalog_folder`

POS (LATTICE 2013) 427



**Figure 4:** Overview of the table structure of NERSC3. The table `catalog_folder` represents ensembles and the table `catalog_file` represents files in the ensembles.

table. Logged in users are also issued `tokens` which are used to authenticate users for batch downloads. NERSC3 is based on the web2py [8] framework.

### 3. The batch download script

NERSC3 allows users to download local ensemble files individually or in batch. Individual downloads can be performed directly from the web pages. For batch downloads, the web page of each local ensemble provides a link to a JSON file containing a list of files in the ensemble. A user does not need to open this file, but only needs to copy its URL, and can then use the `qcdutils_get` utility from `qcdutils` [9] to download all files referenced by the link. For example if the link is `http://qcd.nersc.gov/nersc/api/files/demo` then the user would download the files with the following command:

```

1 > qcdutils_get.py http://qcd.nersc.gov/nersc/api/files/demo
2 http://qcd.nersc.org/nersc/api/files/demo
3 target folder: demo
4 total files to download: 1

```

```

5 downloading demo.nersc
6 demo.nersc 100% |#####
7 completed download: 1/1

```

The `qcdutils_get` script will download the `demo` file, read its contents, and sequentially download all the files listed in there. It shows download progress and keeps a record of completed downloads. If restarted, it resumes from the last completed download.

It also has the ability to convert ensembles to different formats. For example, here we ask to convert the most recent downloads to single precision IDLG format.

```

1 > qcdutils_get.py --convert ildg --float demo/demo.nersc
2 converting: demo/demo.nersc -> demo/demo.nersc.ildg
3   (precision: f, size: 4x8x8x8)
4 100% |#####

```

It keeps a log of all completed operations to avoid duplication of work. One can query `qcdutils` for a log of the completed tasks. For example:

```

1 > qcdutils_get.py demo/qcdutils.catalog.db
2 demo.nersc created on 2011-06-17T13:42:30.876812
3   [14e7cf9106bfcc16388aeac285ccdad9]

```

We refer to the `qcdutils` [9] code and manual for details.

## 4. Conclusions and Outlook

In this paper we have presented a new interface to the Lattice QCD Gauge Connection at NERSC. The purpose of the new interface is to lower the barrier of entry for lattice QCD researchers, and to allow users of the archive to search both local and remote ensembles of lattice gauge configurations. It also allows scientists to access metadata about the ensembles, and to contribute to the metadata by editing wiki entries associated to them.

### Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

## References

- [1] <http://qcd.nersc.gov>
- [2] <http://www.hpss-collaboration.com>
- [3] Maynard, Chris M., arXiv preprint arXiv:1001.5207 (2010).
- [4] Beckett, Mark G. *et al.*, Comp.Phys.Comm. 182 (2011) pp.1208-1214
- [5] Yildirim, Esma *et al.*, High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion: IEEE, 2012.
- [6] Di Pierro, Massimo, Nucl. Phys. B-Proceedings Supplements 106 (2002): 1034-1036.
- [7] Agarwal, A., et al. Journal of Physics: Conference Series. Vol. 219. No. 7. IOP Publishing, 2010.
- [8] Di Pierro, Massimo, Computing in Science and Engineering 13.2 (2011): 64-69.
- [9] Di Pierro, Massimo, arXiv preprint arXiv:1202.4813 (2012).

# Portable Parallel Programs with Python and OpenCL

**Massimo Di Pierro** | DePaul University

Open Common Language (OpenCL) runs on multicore GPUs, as well as other architectures including ordinary CPUs and mobile devices. Combining OpenCL with numerical Python (numPy) and a new module—*ocl*, a Python-to-C converter that lets developers use Python to write OpenCL kernels—creates a powerful framework for developing efficient parallel programs for modern heterogeneous architectures.

In the past five years, consumer CPU clock speed has stalled, hitting a practical limit of 3 GHz. Beyond that limit, cooling the CPU becomes impractical. CPU computing power has nonetheless continued to grow thanks to an increased number of computing cores per CPU. At the same time, we've seen the emergence of heterogeneous architectures, in which the same device can host multiple CPUs and GPUs. To date, a state-of-the-art AMD Opteron 8380 hosts 32 computing cores and an Nvidia Tesla GPU K20X hosts 2,688 CUDA cores. Programming these devices can be a challenge, as traditional programming languages are not well equipped to deal with this level of concurrency.

In our guest editors' introduction for the November/December 2012 issue of *CiSE*, David Skinner and I gave examples of how some modern programming languages—such as Clojure, Erlang, and Haskell—tackle the problem of scaling on multiple cores.<sup>1</sup> Clojure uses transactional memory to simplify multithreaded programming. Erlang uses lightweight threads that can communicate only via message passing. Haskell uses a library of distributed data structures (a read-eval-print loop, or *REPL*) optimized for different types of architectures. However, these approaches aren't as fast as C code, and they aren't designed to work on GPUs. Nvidia developed the CUDA framework specifically for its GPUs.<sup>2</sup> CUDA consists of a C-like programming language that developers can use to write computing kernels, which are compiled and

executed on the GPU cores in a multithreaded-like fashion (although CUDA threads are managed differently than regular threads). CUDA code can also target LLVM (formerly Low-Level Virtual Machine; see <http://llvm.org>), while MCuda<sup>3</sup> enables CUDA to run on multicore CPUs.

The Kronos Consortium—supported by Nvidia, Advanced Micro Devices (AMD), Intel, and ARM—has developed the Open Common Language framework. OpenCL<sup>4</sup> borrows many ideas from CUDA but promises more portability and support for different architectures, including Intel/AMD CPUs, Nvidia/ATI GPU, and ARM chips such as those used on mobile phones. OpenCL consists of a C99 programming dialect. Similar to CUDA, OpenCL programs define kernels that are queued to be executed in parallel on available devices. Kernels running on the same device have access to a shared memory area, as well as local memory areas. OpenCL provides an API that performs loading and saving operations between main memory and device memory. The operations of loading and saving data, and queuing and running kernels, are performed by a host program. This program is usually written in C/C++, but it's also possible to run CUDA and OpenCL kernels from a host program written in other languages, such as Erlang and Python.

Here, I focus on running OpenCL from Python using a library called *pyOpenCL*, created by Andreas Klöckner.<sup>5</sup> I'll also discuss a Python library, called *ocl*, that I developed, which allows just-in-time (JIT)

conversion of Python code into OpenCL code. This approach has no overhead because, once the code is converted, it runs as native OpenCL code. Because the Python ocl module performs a conversion purely at the syntactical level—not at the semantic level—any handwritten OpenCL code can be expressed using the corresponding Python syntax without loss of performance. I don't include benchmarks here, because for any handwritten OpenCL code, I would be able to write Python code that generates the same exact target OpenCL code (including the same variable names). However, I have included examples from this article in the online GitHub repo for ocl so you can easily try them (see <https://github.com/mdipierro/ocl>).

Ocl isn't based—but takes inspiration from—the Cython library,<sup>6</sup> which converts Python modules into C code.

## A First Example with PyOpenCL

To better understand how it works, I should mention that pyOpenCL is built on top of the Python library for efficient array manipulations (that is, numerical Python, or *NumPy*) and OpenCL. It lets programmers embed OpenCL code into a Python program in the form of a string containing the kernel code. It also provides APIs to load a NumPy array on a computing device (GPU or CPU), queue and run the kernel, and retrieve the computation results. In pyOpenCL, object cleanup is tied to the lifetime of objects, which makes it easier to write correct leak- and crash-free code. Moreover, OpenCL errors are automatically translated into Python exceptions.

In the following examples, I won't use pyOpenCL directly, but rather the abstraction layer provided by the ocl library. Here's an example of a simple Python program that we want to parallelize:

```

1 import numpy
2
3 n = 50000
4 a = numpy.random.rand(n).astype
   (numpy.float32)
5 b = numpy.random.rand(n).astype
   (numpy.float32)
6 c = numpy.zeros(n,dtype=numpy.float32)
7
8 for i in range(0, n):
9     c[i] = a[i] + b[i];
10
11 assert numpy.linalg.norm(c - (a + b))
   == 0

```

```

1 from ocl import Device
2 import numpy
3
4 n = 50000
5 a = numpy.random.rand(n).astype(numpy.float32)
6 b = numpy.random.rand(n).astype(numpy.float32)
7
8 device = Device()
9 a_buffer = device.buffer(source=a)
10 b_buffer = device.buffer(source=b)
11 c_buffer = device.buffer(size=b.nbytes)
12
13 kernels = device.compile("""
14 __kernel void sum(__global const float *a, /* a_buffer */
15                     __global const float *b, /* b_buffer */
16                     __global float *c) { /* c_buffer */
17             int i = get_global_id(0); /* thread id */
18             c[i] = a[i] + b[i];
19         }
20     """)
21
22 kernels.sum(device.queue, [n], None, a_buffer, b_buffer, c_buffer)
23 c = device.retrieve(c_buffer)
24
25 assert numpy.linalg.norm(c - (a+b)) == 0

```

**Figure 1.** Scalar product using pyOpenCL.

The program creates three NumPy arrays (*a*, *b*, *c*), and fills the first two with random numbers and the latter with zeros. It then loops and adds *a*[*i*] + *b*[*i*] into *c*[*i*]. Finally, it uses *numpy.linalg* (the linear algebra module) to verify that the norm of *c* - (*a*+*b*) is zero.

The “computing intensive” or kernel part of this program is the *for* loop, which we want to parallelize into a kernel (see Figure 1).

This program is similar to the previous one, and they have many lines in common; however, it also does several new things:

- defines a device object, which encapsulates the OpenCL device context and the OpenCL queue;
- defines device buffers and copies *a* into buffer *a*, and *b* into buffer *b*;
- declares and compiles the kernels (*kernels* = *device.compile(...)*);
- runs the kernel function *sum* (*kernels.sum(...,[n],...)*); and
- retrieves the array *c* from the device.

Notice how the kernel body consists of C99 code, but it's decorated using special macros: “*\_\_kernel*,” which indicates that the function is indeed a kernel; and “*\_\_global*,” which indicates that the arrays to be passed as arguments to the function are in the

```

1 import numpy
2 import pyopencl as pcl
3
4 class Device(object):
5     flags = pcl.mem_flags
6     def __init__(self):
7         self.ctx = pcl.create_some_context()
8         self.queue = pcl.CommandQueue(self.ctx)
9     def buffer(self, source=None, size=0, mode=pcl.mem_flags.READ_WRITE):
10        if source is not None: mode = mode|pcl.mem_flags.COPY_HOST_PTR
11        buffer = pcl.Buffer(self.ctx, mode, size=size, hostbuf=source)
12        return buffer
13    def retrieve(self, buffer, shape=None, dtype=numpy.float32):
14        output = numpy.zeros(shape or buffer.size/4, dtype=dtype)
15        pcl.enqueue_copy(self.queue, output, buffer)
16        return output
17    def compile(self, kernel):
18        return pcl.Program(self.ctx, kernel).build()

```

**Figure 2.** Part of the ocl library, the Device class creates an abstraction layer on top of pyOpenCL.

device's global memory. The function `get_global_id()` returns the ID of the running kernel. The number of running kernels is specified by the `[n]` argument of the call to `kernels.sum`. The last three arguments of this call are passed to the kernel function `sum`.

The `Device` class is defined in the `ocl.py` file and, in terms of the lower level pyOpenCL APIs, it looks like Figure 2.

Here, `self.ctx` is the device context and `self.queue` is the device queue. The functions `buffer`, `retrieve`, and `compile` map into the corresponding pyOpenCL functions `Buffer`, `enqueue copy`, and `Program`, but use a more compact syntax. For more details, see the official pyOpenCL documentation (<http://documentacion.de/pyopencl>).

### Laplace Solver

We can now implement a more complex algorithm. Specifically, given a 2D discretized scalar field  $q$  as input, we want to solve the following 2D Laplace equation in  $u$ :

$$(\partial_x^2 + \partial_y^2)u(x, y) = q(x, y). \quad (1)$$

If we discretize the second derivatives

$$\partial_x^2 u(x, y) = (u(x-h, y) - 2u(x, y) + u(x+h, y)) / h^2$$

$$\partial_y^2 u(x, y) = (u(x, y-h) - 2u(x, y) + u(x, y+h)) / h^2$$

substitute in Equation 1, and solve in  $u(x, y)$ , we obtain:

$$u(x, y) = 1/4(u(x+h, y) + u(x-h, y) + u(x, y+h) + u(x, y-h) - h^2 q(x, y)). \quad (2)$$

We can thus solve Equation 1 by iterating Equation 2 until convergence. Here,  $q$  is the input, and the initial value of  $u$  is irrelevant because it will decorrelate during the convergence process.

We can now write a solver for this problem using pyOpenCL+ocl as shown in Figure 3.

This code differs from the previous code in many ways:

- First,  $u$ ,  $w$ , and  $q$  are 2D arrays ( $w$  does not appear in the equations, but it will be used as temporary storage).
- Next,  $q$  is initialized by setting its value at  $+1$  or  $-1$  at some random sites.
- The kernel `solve` runs on a 2D grid, and therefore each running instance is identified by two numbers,  $x$  and  $y$ , determined by the calls to `get_global_id(0)` and `get_global_id(1)`, respectively.
- The kernel is compiled at runtime, therefore the value of  $n$  is passed to the kernel via string replacement. This is a trick, but it illustrates the possibility of dynamically generating OpenCL code at runtime.
- In the kernel,  $xy$  is a variable that stores the 1D index corresponding to the  $x, y$  coordinates.

```

1 from ocl import Device
2 from canvas import Canvas
3 from random import randint, choice
4 import numpy
5
6 n = 300
7 q = numpy.zeros((n,n), dtype=numpy.float32)
8 u = numpy.zeros((n,n), dtype=numpy.float32)
9 w = numpy.zeros((n,n), dtype=numpy.float32)
10
11 for k in range(n):
12     q[randint(1, n-1),randint(1, n-1)] = choice((-1,+1))
13
14 device = Device()
15 q_buffer = device.buffer(source=q, mode=device.flags.READ_ONLY)
16 u_buffer = device.buffer(source=u)
17 w_buffer = device.buffer(source=w)
18
19
20 kernels = device.compile("""
21         __kernel void solve(__global float *w,
22                             __global const float *u,
23                             __global const float *q) {
24             int x = get_global_id(0);
25             int y = get_global_id(1);
26             int xy = y*WIDTH + x, up, down, left, right;
27             if(y!=0 && y!=WIDTH-1 && x!=0 && x!=WIDTH-1) {
28                 up=xy+WIDTH; down=xy-WIDTH; left=xy-1; right=xy+1;
29                 w[xy] = 1.0/4.0*(u[up]+u[down]+u[left]+u[right] - q[xy]);
30             }
31         }
32 """.replace('WIDTH',str(n)))
33
34 for k in range(1000):
35     kernels.solve(device.queue, [n,n], None, w_buffer, u_buffer, q_buffer)
36     (u_buffer, w_buffer) = (w_buffer, u_buffer)
37
38 u = device.retrieve(u_buffer,shape=(n,n))
39
40 Canvas().imshow(u).save(filename='plot.png')

```

**Figure 3.** Laplace solver with pyOpenCL using the Device class.

Notice that, from the device's viewpoint, all buffers are 1D C-style arrays.

- Finally, the `solve` kernel is called within the loop in `k`. The `[n,n]` argument specifies that we want to run one kernel for each  $x,y$ , where  $x = 0 \dots n - 1$  and  $y = 0 \dots n - 1$ . Because the kernel uses `u` as input and as output, to avoid concurrency issues, we store the output in a temporary array `w` and swap `u` with `w` after each iteration. For didactic purposes, we don't check convergence; we simply iterate Equation 2 a fixed number of times.

We use the `Canvas` library (<https://github.com/mdipierro/canvas>), built on top of the `matplotlib`

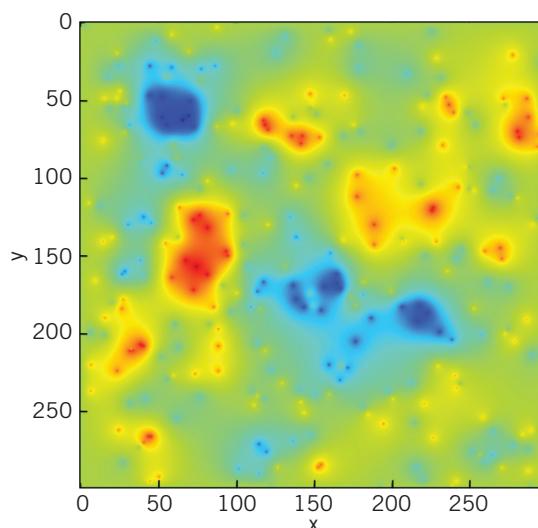
plotting library, to generate an 2D image of the output field `u`. Figure 4 shows the output of a random input. In this implementation, we assume a choice of units such that  $h \equiv 1$ .

### Laplace Solver with ocl

Mixing Python syntax with C syntax can make programs difficult to read and error prone (a syntax error in the kernel might result in an obscure runtime error, for example). A possible solution to this problem consists of coding the kernels themselves in Python. There are two libraries for doing this: `ocl` and `Clyther` (<https://github.com/srossross/Clyther/>) both are based on the Meta decompiler.

**Table 1.** Converting ocl to C99/OpenCL.

Ocl	C99/OpenCL
a = new_type(...)	type a = ...;
a = new_ptr_type(...)	type *a = ...;
a = new_ptr_ptr_type(...)	type **a = ...;
None	Null
ADDR(x)	&x
REFD(x)	*x
CAST(prt_type,x)	(type*)x



**Figure 4.** The output of the Laplace program. This output represents the 2D electrostatic potential for a random charge distribution.

Here, we use ocl, which I specifically designed to work with pyOpenCL.

We rewrite the Laplace solver as Figure 5 shows.

Most of the code is the same as in Figure 3, except for the `solve` kernel; here, the kernel is a Python function “decorated” with `@device.compiler.define_kernel(...)`. This decorator performs introspection of the function at runtime, converts the body of the function into an Abstract Syntax Tree (AST), serializes the AST into C99/OpenCL code, and compiles it using pyOpenCL. This lets us write OpenCL code using Python syntax.

One complication is that C99/OpenCL is a statically typed language, while Python is only strongly typed. We thus need a way to declare the type for variables in Python. For the function

arguments, we can do this in the decorator arguments. For example:

```
w = "global:const:ptr_float"
```

is converted to

```
__global const float *w.
```

For local variables, we do this through the pseudo-casting operators, such as `new_int` and `new_float`. For example:

```
xy = new_int(x*n+y)
```

is converted into

```
int xy = x*n+y;.
```

Variable types must be declared when first assigned. The converter checks that this is the case to prevent further and more obscure compile time errors. The return type is determined automatically, but you must return a variable (or `None`), not an expression. Table 1 shows how other types are converted.

The goal of ocl isn’t to translate arbitrary Python code into C99/OpenCL, but rather to allow the use of Python syntax to write OpenCL. This means that only C99/OpenCL variable types are allowed; Python types, such as lists and dictionaries, aren’t. Moreover, all functions called in the Python code that constitutes the body of the `solve` function are intended to be OpenCL functions, not Python functions.

With ocl, you can define multiple kernels within the same program and call them when needed. You can also use the `@device.compiler.define(...)` decorator to define other functions to be executed on the device. They will be visible and callable by the kernels, but they won’t be callable from the host program.

We can pass constants from the Python context to the OpenCL context as follows: `device.compile(..., constants = dict(n = n))`. Additional C files can be included using the syntax, `device.compile(..., includes = ['#include <math.h>'])`, where `includes` is a list of `#include` statements.

The ocl library also contains a decorator to convert a Python function into C99 and compile it at runtime without using pyOpenCL, as well as a decorator to convert a Python function into a JavaScript function (which is beyond this article’s scope).

```

1 from ocl import Device
2 from canvas import Canvas
3 from random import randint, choice
4 import numpy
5
6 n = 300
7 q = numpy.zeros((n,n), dtype=numpy.float32)
8 u = numpy.zeros((n,n), dtype=numpy.float32)
9 w = numpy.zeros((n,n), dtype=numpy.float32)
10
11 for k in range(n):
12     q[randint(1, n-1),randint(1, n-1)] = choice((-1,+1))
13
14 device = Device()
15 q_buffer = device.buffer(source=q, mode=device.flags.READ_ONLY)
16 u_buffer = device.buffer(source=u)
17 w_buffer = device.buffer(source=w)
18
19 @device.compiler.define_kernel(
20     w='global:ptr_float',
21     u='global:const:ptr_float',
22     q='global:const:ptr_float')
23 def solve(w,u,q):
24     x = new_int(get_global_id(0))
25     y = new_int(get_global_id(1))
26     xy = new_int(x*n+y)
27     if y!=0 and y!=n-1 and x!=0 and x!=n-1:
28         up = new_int(xy-n)
29         down = new_int(xy+n)
30         left = new_int(xy-1)
31         right = new_int(xy+1)
32         w[xy] = 1.0/4*(u[up]+u[down]+u[left]+u[right] - q[xy])
33
34 kernels = device.compile(constants=dict(n=n))
35
36 for k in range(1000):
37     kernels.solve(device.queue, [n,n], None, w_buffer, u_buffer, q_buffer)
38     (u_buffer, w_buffer) = (w_buffer, u_buffer)
39
40 u = device.retrieve(u_buffer,shape=(n,n))
41
42 Canvas().imshow(u).save(filename='plot.png')

```

**Figure 5.** Laplace solver with ocl.

Often, the addition of a new layer adds complexity and makes debugging more difficult. That isn't the case here. Although I use Python syntax to generate OpenCL code, the generated code is statement-by-statement equivalent to Python's source code, including variable names. The conversion layer helps debugging, because it prevents typical typographical errors, such as unbalanced brackets, missing semicolons, and the use of undefined symbols. If the Python code is syntactically valid (and that's easier to check than the OpenCL code), then the generated OpenCL is also syntactically valid. Logic errors are still possible, as are misspellings

of external function names. The generated code is saved and it can be debugged as native OpenCL. Mapping the generated code back into the source Python code is trivial.

OpenCL is the only framework for writing portable applications that work on CPU, GPUs, and mobile devices. Yet programming into native OpenCL code embedded into C/C++ code is very complex and is a major barrier to entry for scientists. Further, while Python—thanks to the numPy libraries—has established itself as a new

standard for scientific computing, it can't take advantage of modern hardware because the interpreter serializes all running code (including multithreaded code). Fortunately, pyOpenCL solves this by letting scientists combine the best of both worlds: using Python for the overall workflow, input/output, and plotting; and using OpenCL for the code's computing-intensive parts. Here, I took this a step further with ocl, which converts Python code into OpenCL code at runtime. ■

### Acknowledgments

I thank Andreas Klöckner for pyOpenCL and the excellent documentation, and Alan Etkin and Jonathan Gemmell for proofreading this article.

### References

1. M. Di Pierro and D. Skinner, "Concurrency in Modern Programming Languages," *Computing in Science & Engineering*, vol. 14, no. 6, 2012, pp. 8–12.
2. Nvidia Developer Zone, *CUDA C Programming Guide*, Nvidia; <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>.
3. J.A. Stratton et al., "MCUDA: An Efficient Implementation of CUDA Kernels for Multi-Core CPUs," *Proc. Int'l Workshop Languages and Compilers for Parallel Computing*, LNCS 5335, 2008, pp. 16–30.
4. J.E. Stone, D. Gohara, and G. Shi, "OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems," *Computing in Science & Engineering*, vol. 12, no. 3, 2010, pp. 66–73.
5. A. Klöckner et al., "PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation," *Parallel Computing*, vol. 38, no. 3, 2012, pp. 157–174.
6. S. Behnel et al., "Cython: The Best of Both Worlds," *Computing in Science & Eng.*, vol. 13, no. 2, 2011, pp. 31–39.

**Massimo Di Pierro** is an associate professor at DePaul University's School of Computing, where he manages the master's program in computational finance and teaches courses in scientific computing, Monte Carlo simulation, parallel programming, and Web development. Di Pierro has a PhD in high-energy theoretical physics from the University of Southampton, UK. Contact him at mdipierro@cs.depaul.edu.



*Selected articles and columns from IEEE Computer Society publications are also available for free at <http://ComputingNow.computer.org>.*

# NEW IEEE computer society STORE

Find the latest trends  
and insights for your

- presentations
- research
- events

[webstore.computer.org](http://webstore.computer.org)

Save up to  
**40%**

on selected articles, books,  
and webinars.



666

**Part IV**

**Grant Applications**



# High Performance Computing in the Classroom

January 21, 2005

# CONTENTS

<b>A Cover Sheet (Auto)</b>	<b>A-3</b>
<b>B Project Summary</b>	<b>B-1</b>
<b>C Table of Content (Auto)</b>	<b>C-1</b>
<b>D Project Description</b>	<b>D-1</b>
D.1 Project Context . . . . .	D-1
D.2 Project Activities . . . . .	D-4
D.3 Dissemination . . . . .	D-9
D.4 Participants . . . . .	D-10
D.5 Project Management and Evaluation Plan . . . . .	D-12
D.6 Facilities . . . . .	D-14
<b>E References Cited</b>	<b>E-1</b>
<b>F Biographical Sketches</b>	<b>F-4</b>
F.1 Co-PI: Massimo Di Pierro . . . . .	F-1
F.1.1 Professional Preparation . . . . .	F-1
F.1.2 Appointments . . . . .	F-1
F.1.3 Related Publications . . . . .	F-1
F.1.4 Other Publications . . . . .	F-1
F.1.5 Synergistic Activities . . . . .	F-2
F.1.6 Collaborators & Other Affiliations . . . . .	F-2
F.2 PI: Ljubomir Perkovic . . . . .	F-1
F.2.1 Professional Preparation . . . . .	F-1
F.2.2 Appointments . . . . .	F-1
F.2.3 Publications . . . . .	F-1
F.2.4 Synergistic Activities . . . . .	F-2
F.2.5 Collaborators & Other Affiliations . . . . .	F-2

<b>G Budget</b>	<b>G-1</b>
G.1 Budget Description . . . . .	G-1
G.1.1 Indirect costs . . . . .	G-1
<b>H Current and Pending Support</b>	<b>H-1</b>
<b>I Facilities, Equipment and Other Resources</b>	<b>I-1</b>
I.1 DePaul CTI . . . . .	I-1
I.1.1 Laboratories . . . . .	I-1
I.1.2 Computers and Networks . . . . .	I-2
I.1.3 Other resources . . . . .	I-3
I.2 Fermilab . . . . .	I-4
I.3 Argonne National Laboratory . . . . .	I-4
<b>J Supplementary Documents</b>	<b>J-1</b>

## A. COVER SHEET (AUTO)

## B. PROJECT SUMMARY

This project addresses curriculum development in several emerging high performance computing models: cluster, grid and Peer-to-Peer (P2P) computing. Much of the high-performance computing (HPC) know-how is limited to a small number of research labs and academic institutions. This project proposes to synthesize this know-how in collaborations with high performance computing experts at Argonne and Fermilab, and to develop courses and integrated sets of educational materials that will be used to train computer professionals into emerging high performance technologies. This project will consist of the following steps: 1) the transfer and synthesis of the methodologies behind the cutting edge research in distributed and parallel computing technologies and applications, 2) the development and testing of materials and tools for training students into these methodologies, and 3) the dissemination of these materials and tools to the educational, scientific and business communities to facilitate the education in high performance computing.

**Intellectual Merit:** While most original applications of high performance computing are scientific, these advanced technologies are finding more and more applications in engineering and financial fields, but the huge potential of these technologies is yet to be realized. It is therefore important to train students into the science of developing distributed and parallel technologies and applications.

**Broader Impact:** Given that more than 40% of Information Technology Masters students in the state of Illinois get their degree from DePaul CTI, the regional impact of these new courses will be large and immediate. Moreover the course material developed will be disseminated and will benefit the educational community by facilitating the creation and the teaching of courses in high performance computing at other universities. The project will also benefit the nation as a whole because of it will broaden the audience of the cutting-edge research performed at Argonne, Fermilab, other national research labs. This audience will consists of computer science students who will then apply the methods and tools developed by the labs in their workplace; this will benefit the students and the business community.

## C. TABLE OF CONTENT (AUTO)

## D. PROJECT DESCRIPTION

### D.1. Project Context

Many of today's scientific projects, in fields such as particle physics, astrophysics, bioinformatics and meteorology, involve the production, distribution, storage, and analysis of vast amounts of data. The enormity of these tasks has required the development, by computer and other scientists as well, of new distributed and parallel high performance computing (HPC) technologies. Universities and National Laboratories are playing a leading role in the development of HPC technologies, using an open source software development model. While most original applications of HPC are scientific, HPC is being adopted in the engineering and financial fields as well, thus indicating its huge economic potential. We believe it is becoming increasingly important to train students into the science of developing distributed and parallel technologies and applications. We have identified cluster, grid and P2P computing technologies to be the three most active and promising areas of HPC research and development.

Until a few years ago, large scale numerical computations were only possible on expensive, shared supercomputers, making it difficult and time-consuming for many scientists and enterprises to run numerical computations. Thanks to the availability of fast Personal Computers (PCs) and network hardware, it is becoming possible to combine dozens or hundreds of very cheap desktop PCs into one parallel "supercomputer". Computer and other scientist are quickly jumping on this opportunity and started developing systems and applications that take advantage of the resources of these computer clusters. Massimo Di Pierro (co-PI) is involved in this field of research and, in particular, in the applications of clusters to the simulation of various physical systems. He is the author of Matrix Distributed Processing, a C++ library for fast development of parallel applications, and FermiQCD, a collection of parallel algorithms for lattice based computations (including parallel inverters and solvers for systems of partial differential equations). There are two basic paradigms for parallel programming: parallel virtual machines (PVM) and Message Passing Protocols (for example MPI). The first is more suitable to old-style shared memory supercomputing architecture.

The second is the preferred choice for Cluster Computing. Parallel Programming based on message passing is more difficult than regular programming because the programmer has to explicitly code the communications between the parallel processes. Moreover communication bandwidth and latency are typically the limiting factors in parallel computations. One the one side the programmer wants to abstract the algorithms and hide communication as much as possible, on the other side he has to modify the algorithms to include communication in the most efficient way. Typical research on Cluster Computing includes communication algorithms, communication protocols that minimize latency, job scheduling and parallel programming.

Grid computing [16] is another high performance computing technology that has recently been developed. It **has found application in many interdisciplinary** areas of research, such as astrophysics, high energy physics, bioinformatics, medical informatics, weather prediction, national defense and terrorist simulations, nuclear reaction simulations, sun surface simulations, geological (earthquake) simulations, and more [2, 3, 4, 7, 9, 10, 20, 22, 23]. Grid computing enables scientists and other users to gain access to computing resources in a fashion similar to the electrical power grid. It utilizes computing resources (either commodity computers or large super computers) from several different institutions, and combines them into one large virtual supercomputer. This virtual machine (VM) is then shared by the members of the different institutions. Since most time on the machines is wasted idling, this increases the availability and the power of the resources available to everyone. Additionally, scarce, novel, or expensive equipment connected to one machine (such as a shake table for geological research) can be more readily accessed by a wider audience.

Current grid computing research includes the development of grid environments and middleware, and the development of algorithms and toolkits for applications. Dave Angulo, one of the PIs, has participated in the following research projects: (1) Grid computing: development of the next-generation Grid environments, and the monitoring and performance evaluation of applications (including Bioinformatics applications) using computational Grids; and (2) Bioinformatics: development of toolkits for biologists to use to research computationally intense problems on Grids, and usage of the Illinois Bio-Grid to allow individual development and running of new applications. Additionally, Ljubomir Perkovic, one of the PIs, is working with his Ph.D. student Gabriele Garzoglio on developing middleware components for the grid computing project at Fermilab that is supporting the research of hundreds of particle physicists.

Peer-to-peer (P2P) computing is a model for distributed computing in which resources and computations are shared between personal computers (like grids)

that are typically at the edges of the internet, and often not even connected to it **on permanent basis** (unlike grids). P2P computing has gained tremendous popularity as distributed filesystems such as Napster and Gnutella use this model to allow the sharing of MP3 music files between Internet users. In addition to distributed filesystems, P2P computing has been used to combine the number crunching power of millions of PCs to analyze data and run simulations in a variety of application domains. SETI@home [28] is a P2P project that analyzes radio waves to search for alien life; Folding@home [29] and Blueprint's Distributed Folding Project [30] is another P2P project that runs protein folding simulation in order to find cures for folding related diseases; distributed.net [31] is a P2P system used to crack cryptography codes and therefore study the security of the cryptographic protocols; Oxford University's ScreenSaver Lifesaver project [32] is the one of the largest P2P projects yet, with more than two million computers joining together to screen 3.5 billion molecules for cancer-fighting potential.

As science, industry and business produce data at a faster and faster rate, the need for CPU cycles to analyze this data and to run simulations increases. P2P systems provide a practical and inexpensive platform for such jobs. However, the task of dividing the work between peers is not always easy. One fundamental problem that confronts decentralized P2P systems is to efficiently locate the peer that stores a particular data item [24]. Another problem that P2P developers face is that some files may be writable by more than one user [11]. **I would remove since repeated later in same words:** This requires files to be replicated and the question then is: how does one maintain consistency between the replicas, efficiently? While some analysis problems, like the one SETI@home is solving can easily be divided into subtasks that can be independently solved, requiring little or no communication or synchronization between the peers, most problems cannot be divided in such a way and more complex algorithms need to be developed to allow the problems to be solved over a WAN in which messages may take a long time [21]. Current research in P2P computing is concerned with solving the communication, naming, synchronization, caching and replication, consistency and fault tolerance problems that appear in the development of P2P systems for more complex data analysis and simulation problems.

High performance computing technologies such as cluster, grid, and P2P computing, have originally been developed for handling today's massive scientific experiments. More and more, however, today's businesses and industries have a need to process large data sets. A pharmaceutical company may need to test millions of molecules for a cure; an investment bank would like to run simulations to predict markets; a large engineering company could benefit from collaborative

engineering software. In each of these cases, an HPC solution can be developed, whether it is using the P2P, cluster or grid model. So, whether it is for science or finance, advanced HPC systems and applications will need to be developed. Unfortunately, HPC expertise is rarely found outside research labs and academia. What is needed are HPC experts who are able to transfer the research into real systems that solve valuable problems. The goal of this project is to satisfy this pressing, national need by developing materials that can be used to train students to become high performance computing professionals.

## D.2. Project Activities

This project aims to develop course materials for introducing advanced undergraduate and first and second year graduate students to the development of high performance distributed and parallel computing technologies and applications. In particular we plan to:

1. synthesize the methodologies behind cutting edge research in the field,
2. develop material and tools for training students into these methodologies, and
3. disseminate our material and tools to the educational, scientific and business communities to facilitate the education in this field.

We will develop lecture notes, exercises, sample software and other teaching material for the following three courses: Cluster Computing, Grid Computing and P2P Computing. This project will be a collaboration between the PI, the co-PIs, the Globus team at Argonne National Laboratory (one of the leading organization behind the development of grid and P2P computing), and the Computing Division at Fermi National Accelerator Laboratory (one of the leading organizations behind the advancement of clusters technologies and grid-based tools applied to particle physics research).

We now describe, separately for each course, the new knowledge, competencies and skills students will have as a result of our courses.

### **Cluster Computing**

**I would remove: The class of problems that can be solved with analytical methods is very small. For many problems of practical interests an analytical solution is not known and they require numerical solutions. Examples span a broad variety of fields such as physics (weather**

**forecasts can be reduced to the problem of solving a non-linear differential equation) biology (protein folding is a typical minimization problem) and finance (pricing exotic derivatives is equivalent to solving a stochastic differential equation).**

In the last ten years the computer power of commodity PCs has doubled every 18 months and network speed has doubled every 12 months. These empirical laws (known as Moore's laws) are having the effect of making numerical methods more and more viable and affordable. Large scale PC clusters, built out of commodity components and running free open source software such as Linux and MPI, score today among the world's fastest supercomputers. Many universities, research centers and small private companies are adopting PC clusters to satisfy their computational needs.

Massimo Di Pierro will develop the course of Cluster Computing and relative teaching material. The course will cover, among others, the following subjects:

- job migration and Linux kernels (openMosix)
- message passing protocols (MPI and GM)
- batch queue systems (PBS)
- job schedulers (such as MAUI)
- distributed file systems (NFS, MFS)
- cluster monitoring tools (such as Ganglia)
- cluster management tools (for example Oscar)
- libraries for parallel programming (including Matrix Distributed Processing [14] developed by the co-PI)
- examples of applications in physics, biology and finance

We will additionally develop: (1) a parallel simulator to initiate students to parallel programming and algorithms; this will be an extension of the PSIM [27] simulator already developed by the co-PI, and (2) a CD containing a collection of selected open source software packages necessary and sufficient to setup a production grade PC cluster. The CD will also contain custom developed installation scripts to simplify the installation process, the developed teaching material and additional technical documentation.

The material developed will enable the students who have taken the course to connect any numbers of PCs in a network, setting up a cluster, manage it and develop generic parallel applications in C/C++.

### **Grid Computing**

**I would remove:** Grid computing is an area of immense interest to the research community. Pressing problems such as gene analysis and protein folding warrant the development of new algorithms and the use of a large number of resources to perform the necessary calculations. Currently, however, there are few courses on grid technology being taught in universities, and those that exist are being taught as mostly research courses at the PhD level at a few research universities. One goal of this project is to develop a package of Grid tools and other pedagogical applications to be utilized by educators in university courses at the graduate and upper undergraduate level. The aim is to teach grid techniques for use in production applications, rather than just in Computer Science research.

Setting up and utilizing a Grid for a production environment can be quite a challenge. Most Grid use today is by Computer Science research projects or large research sites with large professional IT staff. One deliverable for this portion of the project will be to provide an easy to install set of software tools for educators, based on a set of tools already available, such as the PACi Grid-In-A-Box. This set of tools will be customized for typical educational institutional use. It will have few optional packages, set to be installed by a faculty member with portions being installed in individual student accounts. It will also come with pedagogical textual material to explain to students (and educators) what the different pieces are, along with specific installation and usage instructions (customizable for the site).

The package will also contain tools for development of the next-generation Grid environments, tools for monitoring and performance evaluation of applications using computational Grids, tools for the development of portals for the transparent access to Grids by nonexperts, tools to handle the security aspects within computational Grids, and sample applications (especially Bioinformatics applications). Specific topics to be included will be determined in year 1 with the advice and help of the consultants at Argonne and elsewhere. However, those topics will certainly include frameworks [18, 25], scheduling and resource discovery [12, 19], dynamic adaptation [5, 6, 8], security, data management, visualization, communication, job monitoring, portals, and applications.

This development of course materials will build on current Computer Science and Bioinformatics research that is being conducted by one of the PIs, his col-

laborators at Argonne National Laboratory, and his graduate and undergraduate students. The development of course materials will also build on software packages available in the industrial sector, for example, IBM's WebSphere and Oracle's 10G. The team will recruit consultants from industry, such as Steve Nick of IBM, to help guide the development of the pedagogical materials and the integration of existing or planned industry packages into those materials.

### **Peer-to-Peer Computing**

**I would remove:** Peer-to-peer (P2P) computing is a model for distributed computing in which resources are shared between non-dedicated personal computers (PCs) that are typically at the edges of the internet, and often not even connected to it. P2P computing can be used to combine the number crunching power of millions of PCs to analyze data and run simulations in a variety of application domains. As science, industry and business produce data at a faster and faster rate, the need for CPU cycles to analyze this data and to run simulations increases. P2P systems provide a practical and inexpensive platform for such jobs. However, the task of dividing the work between peers is not always easy and current P2P computing research is focused on developing paradigms, algorithms and technologies to facilitate problem solving using P2P computing. **I would remove:** The course materials we plan to develop are going to focus on the issues and problems inherent in P2P computing, and the methods and solutions being currently developed.

For example, one fundamental problem that confronts decentralized P2P systems is to efficiently locate the peer that stores a particular data item. The Napster P2P system uses a centralized indexing server to allow such searches. However, a central naming server is not always a viable nor advisable solution, whether it is because it is not fault-tolerant or a question of politics. Current research is being done to develop efficient protocols to allow efficient searching of data items without a central server (e.g.Chord [24]).

Another problem that P2P developers face is that some files may be writable by more than one user. Gnutella, for example, does not face that problem because MP3 files are typically not modified. In more general applications, files may be modifiable, and often by more than one user may need to access and write to the file (example: collaborative scientific analysis.) This means files need to be replicated and a question current P2P research is focusing on [11] is: how does one maintain consistency between the replicas, efficiently?

Some data analysis problems, like the one SETI@home is solving can easily be divided into subtasks that can be independently solved, requiring little or no communication or synchronization between the peers. Unfortunately, most

problems cannot be divided in such a way and more complex algorithms need to be developed to allow the problems to be solved over a WAN in which network delay is a limiting factor [21].

Because of a lack of central server and the fact that replication is easily achieved in P2P systems by widely propagating data, it would seem that P2P systems are naturally fault tolerant. Unfortunately, this is not true because errors can also then be widely propagated, and those errors can adversely affect the behavior of the system [17].

Overall, current P2P research is concerned with solving the communication, naming, synchronization, caching and replication, consistency and fault tolerance problems that appear in the development of P2P systems for more complex data analysis and simulation problems. We propose to synthesize the methodologies used in the latest P2P research into materials for a course on P2P distributed system development. Past and current P2P systems such as (Napster, Gnutella, SETI@home, distributed.net, Chord, Pastry, Tapestry, Ivy ...) will be used to illustrate the problems and solutions covered. Students will also gain experience in building P2P applications using the project JXTA technology and the PlanetLab platform.

Overall, we will produce the following deliverables for each course:

- **Lecture notes**
- **Exercises with solutions**
- **CD containing notes, exercises and selected software packages**
- **Project web pages where all material can be freely downloaded**

The course materials for the three courses will be tested in new graduate and upper-level undergraduate classes at DePaul University, the home institution of the three PIs. DePaul University has grown to be one of the largest universities in the Chicago area and, according to 2002 data, DePaul CTI (the School of Computer Science, Telecommunications, and Information Systems) offers the largest Information Technology program in the country. The undergraduate program enrolls 2,118 students and offers six different degrees. More than 1039 students are enrolled in the graduate program, which offers nine different master's degrees. Forty percent of IT Master's degrees in the state of Illinois are given by DePaul CTI. DePaul CTI also features a Ph.D. program in computer science that currently enrolls about 50 students. DePaul CTI employs more than 80 full-time faculty and more than 150 part-time ones.

The newly developed classes, Cluster Computing, Grid Computing, Peer-to-Peer Computing, will strengthen CTI's B.S. program in Computer Science and M.S. programs in Computer Science, Distributed Systems, and Software Engineering. Given the size of the student body and the proportion of Illinois students getting IT Master's degrees at DePaul CTI, the regional impact of these new courses will be immediate. The course materials will benefit the educational community because they will facilitate the creation and teaching of courses in high performance computing at other universities. The project as a whole will benefit Argonne, Fermilab, other research labs, and the nation as a whole, because it will ultimately provide another national audience for research done at the labs. This is especially important given that this research is funded, mostly, by the DOE and other federal grants. The audience will consist of computer science students who will then apply the methods and tools developed at the research labs in their workplace, which benefits the students, the business community and, therefore, the whole nation.

In addition, this project is also going to help increase the participation of underrepresented groups in high performance computing because of the diversity of DePaul's student body. DePaul University has a highly diverse student body, and in a 1999 Princeton Review was ranked 2nd out of 331 colleges surveyed in student diversity, and 7th in interaction between students of different races and socio-economic classes. DePaul was also recognized in a 2001 issue of "Black Issues in Higher Education" as one of the top 100 universities in America for awarding bachelor's degrees to minorities. DePaul has accomplished this through its ongoing commitment to providing a quality education to all students, by offering special programs such as the NSF funded scholarships for low-income students entering the IT field, and by creating a learning environment in which interaction between students and faculty members is highly valued.

### D.3. Dissemination

**Once the didacted material mentioned above will have been developed and field tested in our classrooms it will be disseminated in the following ways:**

- It will be posted on a dedicated web page for free download
- It will be presented at National and International conferences on Computer Science Education such as those typically sponsored by the ACM and the IEEE.

- It will be presented to the local teaching and research community (including the University of Chicago, UIC, IIT, Loyola, Northwestern University, Argonne National Laboratory, Fermi National Accelerator Laboratory and selected community colleges) through a workshop organized at DePaul University and supported by this grant.

#### D.4. Participants

*Ljubomir Perkovic* (PI) is an assistant professor at DePaul University. He advises Gabriele Garzoglio, a part-time Ph.D. student and full-time computer professional at Fermilab, whose dissertation focuses on the development of middleware components for a grid computing project supporting the work of particle physicists at FermiLab. Ljubomir is active in the development of the M.S. in Distributed Systems program at DePaul CTI. He has developed two new courses in Distributed Systems. The first course is Foundations of Distributed Systems II, a course on fundamental issues in distributed computing such as synchronization, replication and consistency, and fault tolerance. For this course, he has developed a series of assignments and projects that give students the opportunity to build a simple distributed filesystem and explore the fundamental issues covered in the lectures. The second is Distributed algorithms, a theoretical course on distributed algorithms and the complexity of problems in distributed systems. In addition to Distributed Systems, Ljubomir Perkovic has an interest in the area of graph algorithms in which he has numerous publications.

*Massimo Di Pierro* (co-PI). Before joining CTI, he has been working an Associate Scientist at Fermi National Accelerator Laboratory. His main research fields are High Energy Particle Physics and Lattice Quantum Chromo Dynamics (LQCD) and Parallel Programming. Di Pierro is the author of two software packages for parallel programming (Matrix Distributed Processing [14] and FermiQCD [13]) which are currently used by the LQCD community worldwide to the purpose of performing parallel computations of masses and decay times of sub-nuclear particles. Di Pierro has also developed a number of software packages for didactic purposes, including the Algorithm Animator [26] (distributed with the book "Algorithms" by M. Schaefer and R. Johnsonbaugh) and the Parallel Simulator [27] (for testing and developing parallel algorithms in Python on a single processor machine). At CTI Di Pierro has taught "Object Oriented Programming in C++", "Foundations of Computer Science II", "Analysis and Design of Algorithms" and "Monte Carlo Simulations: Algorithms and Applications".

*Dave Angulo* (co-PI), **Professor at DePaul University, has a back-**

**ground in educating graduates and undergraduates students and an acclaimed research history.** He was a leading member of a team that won two of the three Grand Challenge Awards at SuperComputing 02 by running Grid enabled Bioinformatics applications on a Grid of over 7000 processors spanning 14 countries and 5 continents. He is also the chair of the Life Sciences Grid Research Group in the Global Grid Forum. The Global Grid Forum is the international standards body for Grid computing. He is also the founder of the Illinois Bio-Grid,a Grid of shared computational resources amongst several educational institutions, bio-technology incubator parks, and museums. The Illinois Bio-Grid is currently utilized by investigators for their work on three supported grants. The Illinois Bio-Grid is a Grid of shared computational resources amongst several educational institutions, bio-technology incubator parks, and museums. This Grid is currently utilized by users supported by three grants, viz. (1) NIH R01 MH65560-01, Fine Genomic Mapping of 13q32 in Bipolar Disorder, P.I.: Elliot Gershon, (2) NIH 7R01GM054651-04 Simulations of Cholesterol in Lipid Bilayers, P.I.: H. Larry Scott, and (3) NSF 0228675, ATOL: Early Bird: A Collaborative Project to Resolve the Deep Nodes of Avian Phylogeny, P.I.: Shannon Hackett. In the past five years he has mentored nine undergraduate students and approximately 30 graduate students. This includes two undergraduates through an NSF funded project (GrADS) to support distributed Grid programming (Globus) on the University of Chicago machines, seven undergraduates at Loyola to gain experience in a parallel and distributed computing and Internet technologies, about 15 graduate students at University of Chicago performing research on Bioinformatics applications on the Grid, and about 15 graduate students at DePaul University performing research on Bioinformatics applications on the Grid.

**A preliminary selection of external consultants who have agreed to participate to our project and have submitted a letter of endorsement (attached) include:**

*Rick Stevens*, founder of the Futures Lab at Argonne which investigates problems in large-scale scientific visualization and advanced collaboration environments and from which was produced the widely deployed Access Grid collaboration system. He is the division director of the Math and Computer Sciences division of Argonne National Laboratory and has been the lead PI in many Bioinformatics projects using Grid technology. He is also project director for the National Science Foundation supported TeraGrid project to build the US's most comprehensive open scientific computing infrastructure.

*Ian Foster*, the co-founder of the Globus Project, the widely acclaimed leading middleware for Grid Computing. He won both the Ada Lovelace award and the R&D 100 award in 2002 for his work in Grid Computing. He also won the Gordon

Bell award at SuperComputing 01 for Grid Computing techniques. Dr. Foster is an Associate Division Director for the Math and Computer Sciences division of Argonne National Laboratory.

*Gregor von Laszewski*, the founder of the Globus Java CoG project, which is the core of the newest release of Globus (the leading middleware for Grid technology). He has also been the lead PI in many projects at the forefront of Grid technology. Dr. von Laszewski is a leading member of the Globus Project and a member of the Math and Computer Sciences division of Argonne National Laboratory.

*James Simone*. He received his Ph.D. in theoretical high energy physics from the University of California, Los Angeles and was a Postdoctoral Researcher with Edinburgh University, Fermi National Accelerator Laboratory and the University of Illinois, Urbana. He is presently a Scientist with the Fermilab Computing Division working on heavy quark physics and lattice QCD.

*Don Holmgren*. He received his Ph.D. in experimental condensed matter physics from the University of Illinois in Urbana-Champaign in 1987 and he currently works in the Computing Division at the Fermi National Accelerator Laboratory. Since 1999 he has been the leader of the Lattice Gauge Computing Project in the Fermilab Computing Division, funded by the DOE SciDAC program, that aims to design, construct, and operate large production clusters dedicated to running code for Lattice Quantum Chromo Dynamics.

*Gabriele Garzoglio*, a part-time Ph.D. student in Computer Science at DePaul University, advised by Ljubomir Perkovic (the PI). He works full-time at Fermilab as a Computer Professional in the field of Large Distributed Computing / Grid Systems. During the past two year, he has participated in the design and development of the SAM-Grid, an integrated grid infrastructure for CDF and DZero, the two experiments running at the Tevatron. The goal of this project is providing the end users seamless access to Petabytes of data on a Petaflop-scale computational grid.

We also plan to consult other experts in the HPC field which are not listed here.

## D.5. Project Management and Evaluation Plan

The PI and the co-PIs will divide the work over the three year project as follows: Ljubomir Perkovic will be in charge of P2P computing, Massimo DiPierro will be in charge of cluster computing and David Angulo will be in charge of grid computing.

### **Year 1 Plan: Gathering and Synthesis of current research**

- organize, over the course of the academic year, working meetings, colloquia and mini-workshops in which distributed and parallel computing researchers from Argonne, Fermilab and other research institutions present the state of the art of their area of expertise.
- collect papers, software, documentation.
- synthesize the distributed and parallel computing methodologies through a collaborative effort between the faculty and the researchers and developers at the two labs.
- develop a web site for the project that will make publicly available the results of the communications with experts and the review of the current research.

**Year 1 Evaluation criteria:**

- completeness of the collected materials
- a summary overview of the areas based on the materials collected

**Year 2 Plan: Development and testing of course materials**

- develop course materials in cluster computing, grid computing, and P2P computing including lecture notes, software tools, sample code, assignments and projects.
- test the course materials with advanced undergraduate and first and second year graduate students in 3 separate courses taught at CTI by the three PIs.

**Year 2 Evaluation criteria:**

- completeness of course materials
- student evaluations of courses

**Year 3: Completion and dissemination of course materials**

- complete and publish lecture notes, software, CDs...
- develop a web site to make materials available through downloading
- write, present and publish papers on the materials at appropriate conferences and journals

- organize a workshop of the transfer of high performance computing to the classroom.
- advertise the course materials in appropriate journals.

**Year 3 Evaluation criteria:**

- acceptance of our papers at educational conferences and journals
- acceptance of our materials in courses nationwide;
- success of our workshop; and
- success of our web site measured through web site hits and communication with colleagues at other academic institutions.

**D.6. Facilities**

DePaul CTI is fully committed to this project and offered to buy a new computer cluster dedicated to it. The cluster is in the process of being ordered. The cluster consists of 20 computing nodes connected by Gigabit Ethernet through a Gigabit switch. Each node is based on the latest Intel Pentium 4 processor with hyperthreading technology and is equipped with 2GB Ram and 200GB hard disk. In total this CTI cluster will have 4TB of storage, 40GB of ram and more than 100GFlops of computing power.

The cluster will be running Linux and other Open Source software and it will be dedicated to testing and development of parallel, grid and p2p applications. CTI students and faculties with interest in computation will have access to the cluster.

## E. REFERENCES CITED

## BIBLIOGRAPHY

- [1]
- [2] Alaoui, S; Frieder, T; El-Ghazawi, Bellaachia; and BenSaid, A. "A parallel genetic algorithm for task mapping on parallel machines". Future Generation Computer Systems. Elsevier Science, North-Holland. 2001.
- [3] Allen, G, Angulo, D, Goodale, T, Kielmann, T, Merzky, A, Nabrzysky, J, Pukacki, J, Russell, M, Radke, T, Seidel, E, Shalf, J, Taylor, I. GridLab: Enabling Applications on the Grid. Grid Computing - GRID 2002, Third International Workshop, Baltimore 39-45
- [4] Angulo, D.. Utilizing the XML DOM in C for Grid Applications. Invited Featured Speaker at the 2003 International WebServices/XML Conference & Expo, Toronto.
- [5] Angulo, D, Scott Kuehn.Intelligent Grid Migration (Poster), Proceedings of 2003 Midwest Software Engineering Conference, Chicago.
- [6] Angulo, D and Gerd Lanferman. Nomadic Grid Applications: The Cactus WORM. Invited talk at the High Performance Distributed Computing conference and Globus Retreat, August 2001
- [7] Angulo, David S.; Sickel, Jeff; Steele, Adam. Enhancing Dynamic Fluid Flow Analysis With Computational Grids (Poster). Proceedings of 2003 Midwest Software Engineering Conference, Chicago
- [8] Allen, G; Angulo, D; Foster, I; Lanfermann, G; Liu, Chuang; Radke, T, Seidel, E; and Shalf, J. The Cactus Worm: Experiments with Dynamic Resource Selection and Allocation in a Grid Environment. International Journal of High-Performance Computing Applications, Vol. 15 (4) 2001
- [9] Angulo, D, Parsad, N, Goodale, T, Allen, G, Seidel, E Computing the Smith-Waterman Algorithm on the Illinois Bio Grid (Poster).. Proceedings of 2003 Midwest Software Engineering Conference, Chicago.

- [10] Baldi, P; and Pollastri, G. Machine Learning Structural and Functional Proteomics. *IEEE Intelligent Systems*. Special Issue on Intelligent Systems in Biology, 17, 2, 28-35, (2002).
- [11] Bhattacharjee, B; Chawathe, S; Gopalakrishnan, V; Keleher, P; and Silaghi, B. Efficient Peer-To-Peer Searches Using Result-Caching, Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS'03), Volume 2735, 2003.
- [12] Dail, Holly; Otto Sievert, Fran Berman, Henri Casanova, Asim YarKhan, Sathish Vadhiyar, Jack Dongarra, Chuang Liu, Lingyun Yang, Dave Angulo, and Ian Foster. Scheduling in the Grid Application Development Software Project in Grid Resource Management, Kluwer, 2003
- [13] DiPierro, M. FermiQCD: a Toolkit for Parallel Lattice QCD Applications. 19th International Symposium on Lattice Field Theory (Lattice 2001), Berlin, Germany, 19-24 Aug 2001. Published in Nuclear Physics Proc. Suppl. 106:1034-1036, 2002. e-Print Archive: hep-lat/0110116
- [14] DiPierro, M. "Matrix Distributed Processing." Computer Physics Communications 141, 2001. e-Print Archive: hep-lat/0004007
- [15] Foster, I. Designing and Building Parallel Programs: "Concepts and Tools for Parallel Software Engineering", Addison Wesley.
- [16] Foster, I; Kesselman, C; Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001.
- [17] Hildrum, K; Kubiatowicz, J. "Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-Peer Networks", DISC 2003.
- [18] Kennedy, K, Mazina, M, Mellor-Crummey, J, Cooper, K, Torczon, L, Berman, F, Chien, A, Dail, H, Sievert, O, Angulo, D, Foster, I, Gannon, D, Johnsson, L, Kesselman, C, Aydt, R, Reed, D, Dongarra, J, Vadhiyar, S, and Wolski, R. Toward a Framework for Preparing and Executing Adaptive Grid Programs. April 2002, Proceedings of NSF Next Generation Systems Program Workshop (International Parallel and Distributed Processing Symposium 2002), Fort Lauderdale, FL.
- [19] Liu, C, Yang, L, Foster, I, Angulo, D. Design and Evaluation of a Resource Selection Framework for Grid Applications. HPDC-11, the Symposium on High Performance Distributed Computing, July 2002, Edinburgh, Scotland

- [20] Scheraga, H.A; et. al. Evolution of physics-based methodology for exploring the conformational energy landscape of proteins. *J. Comput. Chem.* 2002, 23, 28-34.
- [21] Shirts, M; and Pande, V. "Screen savers of the world, Unite!" *Science*, Volume 290, Number 5498, pp. 1903-1904, 2000.
- [22] Steele, A, Angulo, D The Illinois BioGrid: A Prototype for Industry-Academe Collaboration.. Proceedings of 2003 Midwest Software Engineering Conference, Chicago.
- [23] Steele, A, Angulo, D, Finkelman, S. Phylogenetics on the Illinois Bio-Grid (Poster). Proceedings of 2003 Midwest Software Engineering Conference, Chicago.
- [24] Stoica, I; Morris, R; Karger, D.; Frans Kaashoek, M; and Balakrishnan, Hari. "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", ACM SIGCOMM 2001, pp. 149-160, San Deigo, CA, August 2001.
- [25] Wolski, Rich; Spring, Neil; and Hayes, Jim. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing., *Journal of Future Generation Computing Systems*, Volume 15, Numbers 5-6, pp. 757-768, October, 1999
- [26] Distributed with the book "Algorithms", M. Schaefer and R. Johnsonbaugh, Prentice Hall, 2003. [www.phoenixcollective.org/mdp/index\\_csc321.html](http://www.phoenixcollective.org/mdp/index_csc321.html)
- [27] [www.phoenixcollective.org/mdp/index\\_psim.html](http://www.phoenixcollective.org/mdp/index_psim.html)
- [28] [setiathome.ssl.berkeley.edu](http://setiathome.ssl.berkeley.edu)
- [29] [folding.stanford.edu](http://folding.stanford.edu)
- [30] [www.blueprint.org](http://www.blueprint.org)
- [31] [www.distributed.net](http://www.distributed.net)
- [32] [www.chem.ox.ac.uk/curecancer.html](http://www.chem.ox.ac.uk/curecancer.html)

## F. BIOGRAPHICAL SKETCHES

## **F.1. Co-PI: Massimo Di Pierro**

Address: School of Computer Science, Telecommunications and Information Systems, DePaul University, 243 S Wabash Av., Chicago, IL 60604 - Phone: 1-312-375-6536 - Fax: 1-312-362-6116 - Email: mdipierro@cs.depaul.edu - Web Page: <http://www.phoenixcollective.org/mdp/index.html>

### **F.1.1. Professional Preparation**

- Ph.D. in Physics, June 1999. University of Southampton, Southampton, UK
- BS-MS in Physics, June 1996. University of Pisa, Pisa, Italy

### **F.1.2. Appointments**

- Assistant Professor, DePaul University, School of Computer Science, Telecommunications and Information Systems, 2002-present
- Postdoctoral fellow, Fermilab, Theory Division (working on Lattice QCD), 1999-2002
- Graduate student, University of Southampton (Southampton, UK), 1996-1999

### **F.1.3. Related Publications**

- [www.fermiqcd.net](http://www.fermiqcd.net) by Massimo Di Pierro. Aug 2003. Proceedings of the Lattice 2003 conference, Tsukuba, Japan.
- A bird's eye view of Matrix Distributed Processing ICCSA Conference Proceedings, Springer Ed. e-Print Archive: cs.dc/0303031
- FermiQCD: a Toolkit for Parallel Lattice QCD Applications. 19th International Symposium on Lattice Field Theory (Lattice 2001), Berlin, Germany, 19-24 Aug 2001. Published in Nuclear Physics Proc. Suppl. 106:1034-1036, 2002. e-Print Archive: hep-lat/0110116
- Matrix Distributed Processing and FermiQCD. 7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2000), Batavia, Illinois, 16-20 Oct 2000. e-Print Archive: hep-lat/0011083
- Matrix Distributed Processing. Computer Physics Communications 141, 2001. e-Print Archive: hep-lat/0004007

### **F.1.4. Other Publications**

- High Precision Lattice QCD Confront Experiment by HPQCD Collaboration and UKQCD Collaboration and MILC Collaboration and Fermilab Lattice Collaboration (C.T.H. Davies et al.). Apr 2003. 4pp. e-Print Archive: hep-lat/0304004

- Excited Heavy-Light Systems and Hadronic Transitions by Massimo Di Pierro, Estia Eichten (Fermilab). FERMILAB-PUB-01-033-T, Apr 2001. 42pp. Published in Physical Review D64:114004,2001. e-Print Archive: hep-ph/0104208
- An Exploratory Lattice Study of Spectator Effects in Inclusive Decays of  $\Lambda_b$  Baryon by UKQCD collaboration (Massimo Di Pierro et al.). 12pp. Published in Physics Letters B468:143,1999. e-Print Archive: hep-lat/9906031
- Toward a Lattice Determination of the  $g_{B^*B\pi}$  Coupling by UKQCD Collaboration (G.M. de Divitiis et al.). 20pp. Published in Journal of High Energy Physics 9810:010,1998. e-Print Archive: hep-lat/9807032
- Mass, Confinement and CP Invariance in the Seiberg-Witten Model by Massimo Di Pierro and Kenichi Konishi. 11pp. Published in Physics Letters B388:90-96,1996. e-Print Archive: hep-th/9605178

#### **F.1.5. Synergistic Activities**

Massimo Di Pierro is author of the following computer programs/libraries:

- Matrix Distributed Processing. A C++ toolkit for fast development of parallel applications. Project web page: [http://www.phoenixcollective.org/mdp/index\\_mdp.html](http://www.phoenixcollective.org/mdp/index_mdp.html)
- FermiQCD. A collection of parallel programs and algorithms for Lattice Computations. The library is currently used at Fermilab and other research centers around the world. Project web page: <http://www.fermiqcd.net>
- Algorithm Animator. A didactic software distributed with the book “Algorithms” written by R. Johnsonbaugh and M. Schaefer, published by Prentice Hall, 2003. Project web page: [http://www.phoenixcollective.org/mdp/index\\_csc321.html](http://www.phoenixcollective.org/mdp/index_csc321.html)

Massimo Di Pierro has thought Object Oriented Programming in C++, Design and Analysis of Algorithms, Foundations of Computer Science II and has developed a class on Monte Carlo Simulations.

#### **F.1.6. Collaborators & Other Affiliations**

Chris Sachrajda, University of Southampton, Southampton, UK (Ph.D. advisor); Adriano Di Giacomo, University of Pisa, Pisa, Italy (Undergraduate advisor); Kenichi Konishi, University of Pisa, Pisa, Italy (Undergraduate thesis advisor); Other collaborators in alphabetic order: Christine Davies, Luigi Del Debbio, Giulia De Divitiis, Alex Dougall, Aida El-Khadra, Estia Eichten, Jonathan Flynn, Steven Gottlieb, Andreas Kronfeld, Peter Lepage, Paul Mackenzie, Masataka Okamoto, Mehmet Oktay, James Simone.

## **F.2. PI: Ljubomir Perkovic**

Address: School of Computer Science, Telecommunications and Information Systems, DePaul University, 243 S Wabash Av., Chicago, IL 60604 - Phone: 1-312-362-8337 - Fax: 1-312-362-6116 - Email: lperkovic@cs.depaul.edu - Web Page: <http://reed.cs.depaul.edu/lperkovic>

### **F.2.1. Professional Preparation**

- Ph.D. in Algorithms, Combinatorics and Optimization, December 1998. Carnegie Mellon University, Pittsburgh, PA
- BA in Computer Science and Mathematics, June 1990. Hunter College of the City University of New York, New York City, NY.

### **F.2.2. Appointments**

- Assistant Professor, DePaul University, School of Computer Science, Telecommunications and Information Systems, 2000-present
- Assistant Professor, Drexel University, Department of Mathematics and Computer Science, 1998-2000
- Research Fellow, University of Wisconsin-Madison, Center for the Mathematical Sciences, Summer 1998
- Research Assistant, Carnegie Mellon University, School of Computer Science, 1990-1997

### **F.2.3. Publications**

- A. Berthiaume, T. Bittner, L. Perkovic, A. Settle, and J. Simon. Bounding the Firing Synchronization Problem on a Ring. Accepted by Theoretical Computer Science, 2004.
- B. Huang, L. Perkovic, and E. Schmutz. Inexpensive d-dimensional matchings. Random Structures and Algorithms, 11, 1:50-58, 2002.
- L. Perkovic and B. Reed. Algorithms for Finding Tree Decompositions of Small Width. International Journal on Foundations of Computer Science, 11, Part 3:365-372, 2000.
- J. Chen, I. Kanj, L. Perkovic, E. Sedgwick, and G. Xia. Genus characterizes the complexity of graph problems: some tight results. In Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP '03), Lecture Notes in Computer Science (LNCS), 2719:845-856, 2003.
- I. Kanj and L. Perkovic. Improved Parameterized Algorithms for Planar Dominating Set. In Proceedings of the 27th Annual Symposium on Mathematical Foundations of Computer Science (MFCS'02), Lecture Notes in Computer Science (LNCS), 2420:399-410, 2002.

#### **F.2.4. Synergistic Activities**

- Advising Ph.D. student and Fermilab scientist Gabriele Garzoglio on his Ph.D. dissertation “A Globally Distributed System for Job, Data and Information Handling for High Energy Physics Applications“, which researches the numerous issues, problems and solutions when developing and running a grid distributed system.
- Developed and taught two new distributed systems courses at DePaul University: CSC322/DS422, Foundations of Distributed Systems II, an undergraduate/graduate course on advanced topics in distributed Systems, and DS591, Distributed Algorithms, an advanced graduate course on distributed algorithms.
- Participated in the development of DePaul’s M.S. in Distributed Systems program, and the updating or creation of new distributed systems (graduate and undergraduate) courses; member of the M.S. in Distributed Systems Program Committee which oversees the M.S. in D.S. curriculum.
- Providing research opportunities to non-traditional students such as part-time graduate and undergraduate students, groups underrepresented in computer science, mathematics, and technology.

#### **F.2.5. Collaborators & Other Affiliations**

Ph.D. advisors: Bruce Reed (McGill U.), Dana Scott (Carnegie Mellon U.);

Ph.D. student: Gabriele Garzoglio (DePaul University and FermiLab);

Other collaborators in alphabetic order: A. Berthiaume (DePaul U.), T. Bittner (DePaul U.), J. Chen (Texas A&M U.), B. Huang (Rutgers U.), I. Kanj (DePaul U.), A. Settle (DePaul U.), E. Schmutz (Drexel U.), E. Sedgwick, (DePaul U.), J. Simon (U. of Chicago), and G. Xia (Texas A&M U.).

## **G. BUDGET**

### **G.1. Budget Description**

#### **G.1.1. Indirect costs**

DePaul University's federally negotiated indirect cost rate is 52% of salaries, wages and benefits.

## H. CURRENT AND PENDING SUPPORT

# I. FACILITIES, EQUIPMENT AND OTHER RESOURCES

## I.1. DePaul CTI

### I.1.1. Laboratories

The School of Computer Science, Telecommunications and Information Systems operates specialized laboratories for research and instruction in artificial intelligence, computer-supported collaborative work, distributed systems, high performance computing, human-computer interaction, information systems and electronic commerce, networked multimedia, software engineering and languages and telecommunications. The school offers a total of approximately 800 student laboratory workstations - most of which are Pentium III or Pentium IV based. These workstations run Windows 2000, Windows XP, Red Hat Linux and Sun Solaris operating systems. The School also operates seven high-performance multi-processor Xeon-based servers with large RAID drives and an IBM ES 9000/9221.

DePaul CTI is fully committed to this project and offered to buy a new computer cluster dedicated to it. The cluster is in the process of being ordered. The cluster consists of 20 computing nodes connected by Gigabit Ethernet through a Gigabit switch. Each node is based on the latest Intel Pentium 4 processor with hyperthreading technology and is equipped with 2GB Ram and 200GB hard disk. In total this CTI cluster will have 4TB of storage, 40GB of ram and more than 100GFlops of computing power. The cluster will be running Linux and other Open Source software, and it will be dedicated to testing and development of parallel, grid and p2p applications. CTI students and faculties with interest in computation will have access to the cluster.

In addition, two laboratories may particularly be relevant for this project:

- The software research lab, which has been in existence for a few years, allows users to test experimental programs without jeopardizing the rest of the system. Currently, the lab has six PCs for installation and development of software.

- The Multimedia Networking Research Laboratory (MNLAB). This laboratory houses 20 Multimedia Workstations, including 14 Sun Ultra 10 workstations running Solaris 7 and 4 PCs running Window NT. The MNLAB workstations are connected with 100 Mbps switch which connects to the MBONE. The lab is supported in part by CTI, Sun Micro Systems and IONA Technologies

The IS division also provides printers and overhead projects for labs and residence halls. In addition to 750 lab workstations, information services maintains over 1,200 residence halls workstations and hundreds of faculty and staff computing platforms

### **I.1.2. Computers and Networks**

DePaul University maintains an extensive technological infrastructure available for students, faculty and staff. In addition, many departments maintain their own resources dedicated for use by its own constituents. The University's existing computer and information infrastructure is in an exceptionally strong position to support the proposed project.

DePaul University is connected by an OC3c ATM-based circuit to the Chicago SBC/AADS network access point (NAP) located in downtown Chicago. The Chicago NAP is one of the largest Internet exchanges in the world. DePaul University maintains approximately fifty peering sessions at the Chicago NAP with institutions of varying sizes and locations including the University's Internet2/Abilene connector MREN. DePaul was one of the first University's in the nation to deploy Juniper router technology at its Internet border, taking advantage of Juniper's award winning design and performance. DePaul University uses BGP, MBGP, PIM-SM and MSDP protocols with its external peering partners to support full line-rate IPv4 unicast and multicast routing.

DePaul University also maintains a physical and logically diverse backup Internet connection to UUNET approximately thirty-five miles north at its Barat Campus in Lake Forest, IL.

The DePaul University metropolitan area backbone network connects campuses at Chicago downtown, Chicago Lincoln Park, Barat campus, Lake Forest, Naperville and Rolling Meadows using Gigabit Ethernet service with a inter-campus speed of 1 gigabit/second.

Almost all of the wired local area networks (LANs) on each DePaul University campus run at 10 and 100 Mb/s switched Ethernet with gigabit uplinks to core backbone devices linking buildings and campuses together. There are also approximately thirty-five 802.11b wireless LAN access points deployed throughout

the University network with more to come as demand increases.

Fully transparent IPv4 unicast and multicast connectivity is available by default for all end hosts using DePaul University's public 140.192.0.0/16 IP address block. There are over 250 subnets within the entire DePaul University network. Critical networks and services are protected with network firewalls, host firewalls or various intrusion detection systems.

All primary uplinks and interconnection points on the DePaul network are managed through SNMP. Monitoring of link utilization, packet drop rates, error rates, device environments, protocol usage and summarized flow statistics are collected. Links and devices are also monitored for availability with alerts sent to pagers or to monitored email accounts. DePaul University's internal Networks and Telecom Group (NTG) in Information Services is responsible for installing, managing and monitoring all internet devices and connections. NTG consists of over twenty full-time staff members performing support duties on a 24x7 on-call basis for routers, LAN switches, cabling, telephony, network security, groupware servers and other various network services.

Each staff member is provided with one desktop computer and one laptop computer.

### **I.1.3. Other resources**

DePaul University provides library services at all campuses, with over 600,000 volumes and 8,000 serial holdings. Delivery of information and materials is linked through computer databases that are all available via Web-based interfaces. The Instructional Technology Development (ITD) Group is staffed with over a dozen members whose role is to help promote, train and develop technological solutions in support of education and research at the University. Library administrators and the ITD group are instrumental in helping bring technologies to faculty throughout the University.

The majority of classrooms are equipped with the latest recording and distance learning technologies including video cameras, document cameras, video cassette recorders, workstation screen capture software, white-board input and microphones. All online class content is automatically managed through the Black Board application including prerecorded video, audio and other multimedia materials used in class. DePaul University is undertaking numerous strides in developing next generation online course content that can be made accessible to students all over the country and throughout the world.

The School of Computer Science, Information Systems and Telecommunications is part of a large liberal-arts university including a School of Education,

giving us ready access to the expertise of curriculum experts. Additionally, being centrally located in the Chicago metropolitan area provides ready access to the expertise of faculty from the University of Chicago, Northwestern, the University of Illinois at Chicago, Loyola University, and the Illinois Institute of Technology. Further, DePaul and CTI enjoy a close relationship with corporations in Chicago, and either employ many IT professionals as adjunct faculty or have them as Alumni.

## **I.2. Fermilab**

Fermilab is a DOE national laboratory and it is operated by the Universities Research Association, Inc., a consortium of 89 research universities in the U.S. and abroad. Fermilab is the largest high-energy physics laboratory in the United States, and is second in the world only to CERN, the European Laboratory for Particle Physics.

Fermilab's Tevatron is the world's highest-energy particle accelerator and collider. In the Tevatron, counter-rotating beams of protons and antiprotons produce collisions allowing scientists to examine the most basic building blocks of matter, and the forces acting on them. Particle physics research has grown into an international effort, with experiment collaborations numbering in the hundreds.

The Fermilab Computing Division furnishes and operates a laboratory wide network and several computing facilities, including central facilities in the Feynman Computer Center. These facilities include different types of parallel machine and, in particular, large computer clusters dedicated to simulation, reconstruction and analysis of scientific data. They also include data storage capacities of more than one Petabyte (in the form of robotic tape storage) and about hundreds of Terabytes of disk storage.

The Fermilab Computing Division leads more than 250 computer professionals, engineers, technicians and physicists in the Computing Division, whose work include R&D projects to prepare computing for future of High Energy Physics programs.

## **I.3. Argonne National Laboratory**

Argonne is one of the U.S. government's oldest and largest science and engineering research laboratories and is the largest in the Midwest. For the past half-century, the University of Chicago has overseen operation of Argonne for the United States Department of Energy and its predecessor agencies. Argonne has four major mission areas, each of which fulfills important governmental and Department of

Energy responsibilities, as well as provides important benefits to society at large. These areas are:

1. Conducting basic scientific research to further understanding of the world we live in. Argonne conducts basic experimental and theoretical scientific research in the physical, life, and environmental sciences.
2. Operating national scientific facilities to help advance America's scientific leadership. Argonne operates world-class research facilities like the Advanced Photon Source.
3. Enhancing the nation's energy resources to ensure America's energy future. Argonne is working to develop and evaluate advanced energy technologies.
4. Developing better ways to manage environmental problems. Argonne is at the forefront in developing new ways to manage and solve the nation's environmental problems and to promote environmental stewardship.

At the same time, Argonne is committed to help the public understand science and to enhance American science, engineering, and mathematics education by helping to train nearly 1,000 college graduate students and post-doctoral researchers every year as part of normal research and development activities.

The Math and Computer Science division at Argonne has an outstanding computational environment that includes high-performance scientific workstations, massively parallel supercomputers, a scalable Linux cluster system, a multimedia laboratory, mass storage systems, and a scientific visualization laboratory.

Argonne is internationally renowned for its activities in Grid Computing. Grid Computing researchers there recently received the R&D100 award, the Ada Lovelace award, and the Gordon Bell prize for research in Grid technology.

## J. SUPPLEMENTARY DOCUMENTS

# National Computational Infrastructure for Lattice Gauge Theory

A proposal in response to Office of Science Notice DE-FG02-06ER06-04 and Announcement Lab 06-04: Scientific Discovery through Advanced Computing.

**Lead Institution:** University of California, Santa Barbara  
Santa Barbara, CA 93106

**Lead Principal Investigator and DOE Contact:** Robert Sugar  
**Address:** Department of Physics  
University of California  
Santa Barbara, CA 93106  
**Email:** sugar@physics.ucsb.edu  
**Phone:** 805-893-3469

**Office of Science Programs Addressed:** High Energy Physics and Nuclear Physics

**Office of Science Program Office Technical Contacts:** Craig Tull and Sidney Coon

## Participating Institutions and Principal Investigators:

### Physics:

Boston University\*, Richard Brower † ‡ and Claudio Rebbi †  
Brookhaven National Laboratory\*, Michael Creutz † ‡  
Columbia University\*, Norman Christ † ‡  
Fermi National Accelerator Laboratory\*, Paul Mackenzie † ‡  
Indiana University\*, Steven Gottlieb ‡  
Massachusetts Institute of Technology\*, John Negele † ‡  
Thomas Jefferson National Accelerator Facility\*, David Richards † and William (Chip) Watson ‡  
University of Arizona\*, Doug Toussaint ‡  
University of California, Santa Barbara\*, Robert Sugar † ‡  
University of Utah\*, Carleton DeTar ‡  
University of Washington, Stephen Sharpe †

### Computer Science:

DePaul University\*, Massimo DiPierro ‡  
Illinois Institute of Technology\*, Xian-He Sun ‡  
University of North Carolina\*, Daniel Reed ‡  
Vanderbilt University\*, Theodore Bapty ‡

\* Institution submitting an application

† Project Principal Investigator, Member of Lattice QCD Executive Committee

‡ Institution Principal Investigator

# 1 Executive Summary

Our long range objective is to construct the computational infrastructure needed for the study of quantum chromodynamics (QCD). Nearly all theoretical physicists in the United States involved in the numerical study of QCD are participating in this effort [1], as are Brookhaven National Laboratory (BNL), Fermi National Accelerator Laboratory (FNAL) and Thomas Jefferson National Accelerator Facility (JLab), and computer scientists at DePaul University, the Illinois Institute of Technology, the University of North Carolina and Vanderbilt University. A very successful start was made under the first phase of the Department of Energy's Scientific Discovery through Advanced Computing Program (SciDAC-1). We propose to build on this success to address new challenges that must be met in order to capitalize fully on the exciting opportunities now available for advancing the study of QCD.

QCD is the component of the Standard Model of elementary particle physics that describes the strong interactions. The Standard Model has been enormously successful; however, our knowledge of it is incomplete because it has proven extremely difficult to extract many of the most important predictions of QCD, those that depend on the strong coupling regime of the theory. To do so from first principles and with controlled systematic errors requires large scale numerical simulations within the framework of lattice gauge theory. Such simulations are needed to address problems that are at the heart of the DOE's large experimental programs in high energy and nuclear physics. Our immediate objectives are to 1) calculate weak interaction matrix elements to the accuracy needed to make precise tests of the Standard Model; 2) determine the properties of strongly interacting matter under extreme conditions such as those that existed in the very early development of the universe, and are created today in relativistic heavy ion collisions; and 3) calculate the masses of strongly interacting particles and obtain a quantitative understanding of their internal structure. The infrastructure we propose to build is essential to achieve these objectives.

The bulk of our effort in SciDAC-1 was devoted to software development, and that will continue to be the case under this SciDAC-2 proposal. Under SciDAC-1 a QCD Applications Programming Interface (QCD API) was developed, which enables lattice gauge theorists to make effective use of a wide variety of massively parallel computers, including those with switched and mesh architectures. The QCD API was optimized for the custom designed QCD on a Chip (QCDOC) computer, and for commodity clusters based on Pentium 4 processors. Under this proposal, optimized versions of the QCD API will be created for clusters based on multi-core processors and Infiniband communications networks, and for the Cray XT3, the IBM BlueGene/L and their successors. The QCD API will be used to enhance the performance of the major QCD community codes and to create new applications. A QCD physics toolbox will be constructed which will contain sharable software building blocks for inclusion in application codes, performance analysis and visualization tools, and software for automation of physics work flow. New software tools will be created for managing the large data sets generated in lattice QCD simulations, and for sharing them through the International Lattice Data Grid consortium. A common computing environment will be developed for the dedicated lattice QCD computers at BNL, FNAL, and JLab. Work on multi-scale algorithms recently begun in collaboration with members of the Terascale Optimal PDE Simulations (TOPS) Center will be extended.

The lattice QCD infrastructure effort has included the development of hardware as well as software, because for the study of QCD it has proven more cost effective to build specialized computers than to make use of general purpose supercomputers. We have pursued both customized clusters constructed from commodity components and the development of fully customized computers. Research and development work on commodity clusters was carried out under our SciDAC-1 grant at FNAL and JLab. The experience gained with these prototype clusters will enable us to build highly cost effective terascale clusters in the coming year. In parallel with SciDAC-1, but funded separately by the DOE, a 12,288 processor QCDOC computer was constructed at BNL for use by the U.S. lattice QCD community. In SciDAC-2 we propose to continue to track the evolving commodity and semi-commodity marketplace and to undertake design of a fully customized successor to the QCDOC. A four year Lattice QCD Computing Project began on October 1, 2005 with funding from the DOE's High Energy Physics and Nuclear Physics Programs. The purpose of this Project is to construct and operate dedicated computers for the study of QCD. Both the hardware research and development and the software development we propose are critical to the success of this Project and to research in lattice QCD in the U.S.

## 2 Physics Goals

### 2.1 Tests of the Standard Model

Despite its extraordinary success, the Standard Model is believed to be only the low energy (long distance) limit of a more fundamental theory. Therefore, a major component of the experimental program in high energy physics is devoted to making precise tests of the Standard Model in order to determine its range of validity and search for indications of new physics beyond it. Many of these tests require both accurate experiments and accurate lattice QCD calculations of the effects of the strong interactions on weak interaction processes. In almost all cases, the precision of the tests are limited by the uncertainties in the lattice calculations, rather than in the experiments. Our objective is to bring the lattice errors down to, or below, the experimental ones.

The greatest challenge to performing accurate numerical calculations of QCD is to include the full effects of vacuum polarization due to light (up, down and strange) quarks. Significant progress has been made in meeting this challenge during the past five years through the use of improved formulations of QCD on the lattice and through rapid growth in the computing resources available to the field [2, 3, 4]. Among the notable results have been calculations of the leptonic decay constants of the  $\pi$  and  $K$  mesons [5] and mass splittings in the charmonium [6] and bottomonium [7] spectra to an accuracy of 3% or better; the first determination of the light quark masses to fully include their vacuum polarization effects [8, 9]; the calculation of the strong coupling constant [10] and the Cabibbo-Kobayashi-Maskawa (CKM) matrix element  $V_{us}$  [11, 5] to the same accuracy as their experimental determinations. The lattice gauge theory community has moved from the validation of techniques through the calculation of quantities that are well known experimentally to the successful prediction of quantities that had not previously been measured. Three cases in which predictions were subsequently confirmed by experiment were the calculations of the leptonic decay constant [12] and semi-leptonic form factors [13] of the  $D$  meson, and the mass of the  $B_c$  meson [14]. The decay constants and form factors for  $B$  mesons play important roles in tests of the Standard Model, but are very difficult to measure experimentally. The lattice calculations are similar for D and B mesons, since only the masses of the heavy quarks change. Thus, the successful calculations for  $D$  mesons provide important validation of those for  $B$  mesons which are now in progress.

Measurement	CKM Matrix Element	Hadronic Matrix Element	Non-Lattice Errors	Lattice Errors 2004	Lattice Errors Current	Lattice Errors 6.0 TF-Yr	Lattice Errors 40. TF-Yr
$\varepsilon_K$ ( $\bar{K}K$ mixing)	$\text{Im } V_{td}^2$	$\hat{B}_K$	9%	20%	12–20%	5%–8%	3%–4%
$\Delta M_d$ ( $\bar{B}B$ mixing)	$ V_{td} ^2$	$f_{B_d}^2 B_{B_d}$	6%	30%	22%	8%–10%	6%–8%
$\Delta M_d / \Delta M_s$	$ V_{td}/V_{ts} ^2$	$\xi^2$	—	12%	8%	6%	3%–4%
$B \rightarrow (\frac{p}{\pi}) l v$	$V_{ub}$	$\langle \frac{p}{\pi}   (V - A)_\mu   B \rangle$	7%	15%	14%	5.5%–6.5%	4%–5%
$B \rightarrow \binom{D^*}{D} l v$	$V_{cb}$	$\mathcal{F}_{B \rightarrow \binom{D^*}{D} l v}$	2%	4.4%	3%	1.8%–2%	1%–1.4%

Table 1: The impact of improved lattice QCD calculations on the determination of CKM matrix elements.

The results quoted above indicate that we are in a position to make very significant progress over the next five years. The current lattice and experimental uncertainties in some key quantities are shown in Table 1, along with the reduction in lattice errors expected as more computational resources become available, as well as expected improvements in ancillary theoretical calculations of operator normalization factors. All quantities in the table have had first calculations which fully include the effects of vacuum polarization due to light quarks. The error estimates in Table 1 are based on our experience with the improved staggered formulation of lattice quarks, as were the successful calculations cited above, with the exception of the  $\varepsilon_K$  estimates which are based on domain wall quarks as well.

## 2.2 Matter under extreme conditions

At very high temperatures and/or densities, one expects to observe a phase transition or crossover from ordinary strongly interacting matter to a plasma of quarks and gluons. A primary motivation for the construction of the Relativistic Heavy Ion Collider (RHIC) at BNL was to observe the quark–gluon plasma and determine its properties. During the early development of the Universe matter was in the plasma state, and the quark-gluon plasma may be a central component of neutron stars today. The behavior of strongly interacting matter in the vicinity of the phase transition or crossover is inherently a strong coupling problem, which can only be studied from first principles through lattice gauge theory calculations. Among the issues that can uniquely be addressed by lattice calculations are the nature of the transition, the temperature at which it occurs, the properties of the plasma, and the equation of state. Indeed, it is the lattice that has given us the best estimates of the temperature of the deconfinement transition [15]. Lattice results will continue to be crucial to the interpretation of ongoing heavy-ion experiments in the United States and Europe.

A major goal of our research program is to investigate the properties of matter under the extreme conditions of high temperature and high density. Important progress has been made in the last several years [15] in determining the transition temperature [16, 17], the phase diagram [17] and the equation of state [18, 19, 20, 21]. However, as the deconfinement process occurs at a temperature of order 175 MeV, a new scale is introduced, as well as new potential lattice artifacts. Thus, these calculations are computationally quite demanding. Those at zero baryon density are well understood theoretically, and only require sufficient computational resources to reach high precision results; but, calculations at non-zero baryon density are at a much earlier stage of development.

The finite density problem introduces algorithmic issues that remain unresolved and require exploratory work on new ideas such as calculations at fixed quark number rather than fixed chemical potential. Moreover, several new approaches to this long-standing problem [18, 19, 20] have been suggested that are applicable at high temperature and small values of the baryon density, the regime relevant to the RHIC experiments. Also, at vanishing baryon density, new exact algorithms have been developed for staggered fermions. These new techniques have to be explored and implemented into existing code packages.

The main goal of the ongoing studies at zero baryon density is to extend calculations of the transition temperature, the equation of state and the properties of the high temperature phase to an almost realistic quark mass spectrum on large lattices with small lattice spacings. This will allow a controlled extrapolation to the continuum and thermodynamic limits. This research effort is using a large portion of the resources provided by the DOE QCDOC supercomputer at Brookhaven. Specifically, the lattice group at BNL and the MILC collaboration use improved staggered fermion actions (p4-action, asqtad) with smeared links to reduce flavor symmetry breaking and the cut-off distortion of thermodynamic observables [22].

Code packages for studies of thermodynamics at non-zero baryon density are implemented in these calculations. The specific approach used by the BNL and MILC groups is based on a high order Taylor series expansion. This will allow exploration of the QCD phase diagram also at high temperature and non-zero baryon density.

A somewhat different computational set-up is required to study hadron properties at high temperature. The goals here are two-fold. In the heavy quark sector one wants to understand the stability of charmonium states in the finite temperature plasma and determine the temperature at which these states get dissolved [23]. In the light quark sector, the emphasis is on calculating thermal dilepton and photon rates. Currently these calculations are being performed on large quenched lattices using Wilson fermions. These calculations will be extended to improved Wilson fermions, which reduce the distortion effects resulting from so-called Wilson doublers. Moreover, preparations for the first exploratory studies of these effects with dynamical quarks are underway.

## 2.3 Structure and interactions of hadrons

A major scientific goal of our collaboration is to achieve a quantitative, predictive understanding of the structure and interactions of strongly interacting particles (hadrons) from lattice QCD. This will achieve key objectives of the DOE Strategic Plan and the Nuclear Science Long Range Plan, which respectively highlight the goals of developing a quantitative understanding of how quarks and gluons provide the binding and spin

of the nucleon based on QCD and of connecting the observed properties of nucleons with the underlying theoretical framework provided by QCD. Hadronic observables calculated from first principles are directly relevant to experiments at Bates, JLab, RHIC-spin, SLAC, and FNAL, and will have significant impact on future experiments at the JLab 12 GeV upgrade and electron-ion collider.

Past accomplishments have established the methodology and laid the groundwork for hadron structure and spectroscopy calculations. Since the cost of full QCD calculations in a volume large enough to contain a pion grows roughly as  $m_\pi^{-7} - m_\pi^{-9}$ , initial calculations were restricted to the “heavy pion” domain of pion masses in excess of 500 MeV. In this domain, form factors, the lowest three moments of quark, spin, and transversity distributions, and generalized form factors corresponding to the lowest three moments of generalized parton distributions have been calculated [24, 25]. Salient achievements include separating the contributions of the quark spin and orbital angular momentum to the nucleon spin [24] and observing strong dependence of the transverse size on the nucleon on the longitudinal momentum fraction [25]. The transition form factor between the nucleon and Delta has been calculated to explore the role of deformation [26, 27]. In spectroscopy, techniques to calculate extended sources within the appropriate representation of the hypercubic group have been developed and utilized to calculate ground and excited states in each symmetry channel [28, 29] and pentaquark states were calculated using a complete set of local sources [30].

An essential step toward the chiral regime with light quarks has recently been taken using a hybrid calculation combining computationally economical staggered sea quark configurations generated by the MILC collaboration and domain wall valence quarks that have lattice chiral symmetry. The axial charge has recently been calculated for pion masses as light as 350 MeV. Since this is in the regime of applicability of chiral perturbation theory, analytic expressions for the mass and volume dependence were used to extrapolate to the physical pion mass and infinite volume, obtaining the axial charge to a precision of 6.8% and in agreement with experiment. Hybrid calculations of the pion form factor were also performed [32]. In a first step in studying hadron-hadron interactions, the  $I = 2 \pi - \pi$  scattering length [33] and nucleon-nucleon  $^1S_0$  and  $^3S_1 - ^3D_1$  scattering lengths [34] have also been calculated in this chiral regime.

Building on this solid foundation, we propose an extensive program of precision calculations of the hadron observables described above and exploratory calculations of more demanding observables. Using MILC configurations at lattice spacings of 0.12, 0.09 and 0.06 fm and pion masses down to 250 MeV, form factors, moments of quark, spin, and transversity distributions, generalized form factors, and transition form factors will be calculated with careful control of the errors associated with the lattice spacing, lattice volume, and quark mass. When computational resources permit, the hybrid combination of staggered and chiral quarks will be replaced by fully consistent sea and valence quarks.

Important new algorithms and observables will also be explored. The spectroscopy of so-called “missing” baryon resonances and of mesons with exotic quantum numbers has great potential impact on our understanding of QCD, and on the current and future spectroscopy program at JLab. Exploration of the baryon and meson spectrum into the chiral regime requires further development. Multi-hadron operators and the use of a range of lattice volumes will be used to study the properties of unstable resonances. Stochastic all-to-all quark propagators with dilution and exactly-determined low eigenvectors will be used to facilitate both these spectroscopy calculations and the calculation of disconnected diagrams for hadron structure observables. Nucleon-nucleon scattering lengths will be calculated in the chiral regime and the static potentials between heavy-light hadrons will be explored. Since past efforts to calculate important gluon observables in hadrons have been overwhelmed by the large fluctuations of the gluon field, new approaches will be investigated. Hadron calculations in the chiral regime also open the door to understanding the physical origin of observed structure, and the role of mechanisms such as diquark correlations and the quark zero modes associated with topological excitations will be explored. This combination of well-understood observables that can be calculated with confidence given the requisite resources and more speculative exploration of new physics offers exciting opportunities for the fundamental understanding of hadronic physics.

## 2.4 Lattice quarks and other physics directions

In recent years, much of the progress in the numerical study of QCD has come about through the use of improved formulations of quarks on the lattice. There are a number of different formulations that appear to be promising, each of which has its advantages and disadvantages. Most of the work cited above made use of the staggered formulation of lattice quarks. This formulation has the advantage of enabling simulations

at quite small quark masses with current computers, but in order to have the correct number of quarks in the continuum limit one must perform simulations with the fourth-root of the quark determinant. It has been suggested that taking the fourth-root of the determinant at finite lattice spacing might give rise to unphysical non-localities that persist in the continuum limit [35]. The excellent agreement of existing results with experiment, as well as a growing body of direct discussions of the issue [36, 37], give us confidence that no fundamental problem exists. However, further work is warranted, and in progress. We have started an extensive set of simulations with domain wall quarks, which is likely to continue through most, or all, of the proposed grant [4]. This formulation has the advantage of having nearly exact chiral symmetry on the lattice, but requires significantly more computing resources than staggered quarks for the same parameter regime. Studies with domain wall quarks will increase the range of quantities that can be computed, will provide critical tests of the staggered quark results, and in the long run may increase the accuracy of those results [38].

The methods used for QCD can also be adapted for other strongly coupled theories. Noteworthy examples include QCD with a large number of colors, where it may be possible to build a bridge to analytic methods based on string theory and the AdS/CFT correspondence; a strongly coupled Higgs sector; and proposed models for physics beyond the standard model involving strongly coupled gauge interactions such as supersymmetry and “little Higgs” models. These other applications are generally more challenging than QCD, and work is at an early stage. We expect, however, that an increasing fraction of the US community will work on such theories during the next five years.

### 3 SciDAC-1 Software: The QCD Applications Programming Interface

Under its SciDAC-1 grant, the U.S. lattice gauge theory community has created a unified program environment that enables its members to achieve high efficiency on terascale computers. Among the design goals were to enable users to quickly adapt codes to new architectures, easily develop new applications and incorporate new algorithms, and preserve their large investment in existing codes. These goals were achieved through the development of the QCD Applications Programming Interface (QCD API), which is illustrated in Fig. 1.

All of the fundamental components of the QCD API have been implemented and are in use on the U.S. QCDOC hardware at BNL, on both the switched and mesh architecture Pentium 4 clusters at FNAL and JLab, and on a number of general purpose supercomputers. The QCD API is being used by a growing number of physicists in the U.S. and abroad. The software code and documentation can be found at the USQCD <http://www.usqcd.org/usqcd-software>. Here we briefly describe each of its components.

The QCD API has a layered structure which is implemented in a set of independent libraries. Level 1 provides the code that controls communications and the core single processor computations. To obtain high efficiency on terascale facilities, much of this layer may have to be written in hardware specific assembly language. However versions exist in C and C++ using MPI for transparent portability of all application codes.

**Message Passing:** QMP defines a uniform subset of MPI-like functions with extensions that (1) partition the QCD space-time lattice and map it onto the geometry of the hardware network, providing a convenient abstraction for the Level 2 data parallel API (QDP); (2) contain specialized routines designed to access the full hardware capabilities of the QCDOC network and to aid optimization of low level protocols on networks in use and under development on clusters. There is a basic test suite to verify each implementation.

**Linear Algebra:** All lattice QCD calculations make use of a set of linear algebra operations in which the basic elements are three-dimensional complex matrices, elements of the group SU(3). These operations are local to lattice sites or links and do not involve inter-processor communications. We have collected them into a single Level 1 library called QLA. The QLA routines can be used in combination with QMP to develop complex data parallel operations in QDP or in existing C or C++ code. The C implementation has about 19,000 functions generated in Perl, with a full suite of test scripts. The C++ implementation makes considerable use of templates, and so contains only a few dozen templated classes (the required specific classes are generated on demand by the compiler). For both C and C++ it is important to optimize the code for the most heavily used linear algebra modules.

**Data Parallel Interface:** Level 2 (QDP) contains data parallel operations that are built on QMP and QLA. The C implementation is being used to improve performance of the MILC code, a large, publicly available suite of applications. Despite the fact that the MILC code has been carefully optimized over its fifteen year lifetime, rewriting computationally intensive subroutines in QDP makes a significant improvement in its performance. Chroma, an entirely new application code base, has been written *di novo* in the C++ implementation of QDP. QDP allows extensive overlapping of communication and computation in a single line of code. By making use of the QMP and QLA layers, the details of communications buffers, synchronization barriers, vectorization over multiple sites on each node, etc. are hidden from the user.

**Level 3 Subroutines:** A very large fraction of the resources in any lattice QCD simulation go into a few computationally intensive subroutines, most notably the repeated inversion of the Dirac operator, a large sparse matrix. To obtain the level of efficiency at which we aim, it is necessary to optimize these subroutines for each architecture. For example, on the QCDOC, the assembly coded inverter for the Domain Wall and Asqtad quark actions, the two quark formulations that are being used in initial work, is as high as 42% and 45% of peak, respectively. (The precise performance depends on the number of lattice sites assigned to each processor). These percentages correspond to total sustained performances of 4.1 and 4.4 teraflop/s for the full 12,288 processor machine. Level 3 codes written with SSE2 instructions achieve up to 3.0 gigaflop/s per processor for the most recent cluster built at JLab, which has 3.0 GHz dual core Pentium 4 processors.

**Data Management:** A very large fraction of the computing resources used in lattice QCD calculations go into Monte Carlo simulations that generate representative configurations of the QCD ground state. The same configurations can be used to calculate a wide variety of physical quantities. Because of the large resources needed to generate configurations, the U.S. lattice community has agreed to share all of those that are generated with DOE resources. To enable this sharing we have created standards for file formats, and written an I/O library (QIO) that adheres to them. We are charter members of the International Lattice Data Grid (ILDG), which is setting a basic set of meta-data and middleware standards to enable international sharing of data. By June 2006, the U.S. lattice gauge theory community will be fully capable of archiving and retrieving data on the ILDG.

## 4 SciDAC-2 Software

The full benefits of the SciDAC-1 infrastructure are just beginning. To capitalize on the accomplishments to date will require continued work on porting, optimization, testing and distribution of software libraries. In addition, there is a new set of requirements and challenges as we prepare for the petaflop/s era. We propose to extend the QCD API and its related libraries under SciDAC-2 to meet these challenges. As a guide to the discussion below, we summarize the proposed API in Fig. 1.

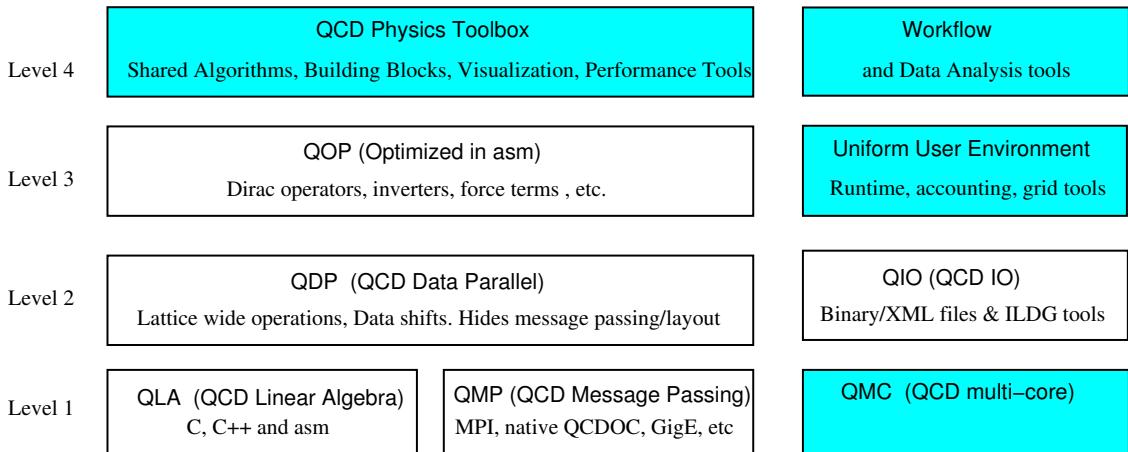


Figure 1: Proposed SciDAC-2 QCD API — The SciDAC-1 components are shown in white, and the new SciDAC-2 components in aqua.

## 4.1 Machine specific software

The basic MPI and C/C++ code is highly portable, but to achieve high performance may require machine specific software for both the Level 1 and 3 routines. Such software has been written under SciDAC-1 for the QCDOC and clusters based on single core Pentium 4 processors. Machines that will be targeted in the first stages of this grant are clusters based on multi-core processors and Infiniband communications fabrics, and the Cray XT3, the BlueGene/L and their successors. Attention will also be paid to emerging technology, such as the Cell processor, so that we are ready to take advantage of any major new developments that might emerge.

**QMC: Threaded libraries for multi-core processors:** All of the principal manufacturers of commodity microprocessors, including Intel, AMD, and IBM, have started the move toward multi-core processors in the last two years. The latest SciDAC-1 prototype cluster is based on an Intel dual core microprocessor, and the first large scale cluster constructed under the LQCD Computing Project will make use of a multi-core processor as well. By 2007, the majority of processors sold to the commodity market are expected to be dual core, with a planned movement to quad and higher cores. The BlueGene/L has dual core PowerPC processors, and we anticipate a rapid expansion in the number of cores in the BlueGene/P and the QCDOC-2 described below.

In the short run, lattice QCD application codes can take advantage of multi-core microprocessors by simply treating the cores as independent processors. That is, a lattice QCD application implemented with QMP or MPI can run using a separate process on each of the cores. Message passing between the processes running on the cores relies on the shared memory structures provided by MPI implementations for SMP systems. Although communications between the processes requires copying data from one process to the shared memory and back from the shared memory to the second process, the scaling observed using this technique is very encouraging. However lattice QCD codes tend to be memory bandwidth limited, so we anticipate that threaded code will be necessary for peak performance. Unfortunately, standard implementations of POSIX threads often have high overhead for locks, or in some cases may not be available on specialized architectures. We therefore propose to develop a new light-weight Level 1 multi-core library or threaded interface standard (QMC). This is conceptually on the same level as QLA and QMP. It will provide an abstraction for threads that can be implemented for portability in the POSIX standard, but will have native implementations for highest performance and for unconventional multi-core architectures.

The QDP and Level 3 codes based on QMC will be improved by avoiding the memory copies used by MPI to pass messages between processes. We will begin by carefully studying the options. Several techniques are available to avoid the memory copies: (i) Have independent processes on each of the cores use the same shared memory area to store the sub-lattice, with each process performing calculations on a fraction of the sub-lattice; (ii) Use OpenMP or a similar parallel compiler to implicitly thread key code loops. In this case, one process runs on a given machine, with one thread spawned per core, performs calculations on a fraction of the sub-lattice; (iii) Use explicit multi-threading with a thread on each core handling a fraction of the sub-lattice. These techniques vary in difficulty and in level of effort required. Further, these optimizations may be done at either the API library level (QLA, QDP), or at the application level (Level 3), or both. Modeling and software prototyping will be required to investigate the costs and benefits of the approaches.

**QMP: Native implementations for Infiniband and BlueGene:** The latest lattice QCD clusters constructed at JLab and FNAL under our SciDAC-1 grant, and the initial ones to be constructed in the LQCD Computing Project are based on Infiniband fabrics. For lattice QCD codes, Infiniband delivers superior price/performance. It offers the highest communications bandwidth between computers available on the market and exhibits low short-message latencies, both critical for lattice QCD applications.

The Infiniband software stack provides several communications protocols, including TCP/IP, channel-based communications (remote direct memory access, or RDMA), and message-based communications (verbs application program interface, or VAPI). The later two, RDMA and VAPI, deliver the best performance in terms of highest bandwidth, lowest latency, and lowest burden to the host processor. Two open source MPI implementations are available which are based on combinations of RDMA and VAPI: MVAPICH, from the Ohio State University, and MPICH-VMI, from the National Center for Supercomputing Applications.

The JLab and FNAL Infiniband clusters currently rely on the implementation of QMP that uses MPI for the underlying communications. Simple benchmarks show that for the message sizes of interest on lattice QCD codes, communications using native RDMA and VAPI calls have lower latencies than those using MVA-

PICH or MPICH-VMI. Careful evaluation of lattice QCD codes using computationally intensive kernels, such as the conjugate gradient inversion routines, will be used to infer whether a “native” QMP over RDMA and/or VAPI has a sufficient performance improvement over an MPI version to warrant the manpower for a full implementation. This software prototyping would be a continuation of work started under the SciDAC-1 grant.

The BlueGene/L, a direct descendant of the QCDOC, has considerable potential for the study of QCD. Given the plan to install a large BlueGene/P at Argonne National Laboratory, it seems worthwhile to develop a native version of QMP for the BlueGene line. We propose to use the single tower BlueGene/L’s a Boston University and Massachusetts Institute of Technology, and our close collaboration with IBM in this project.

**QLA: Optimized Linear Algebra Routines:** As indicated above, it is important to optimize the most heavily used linear algebra routines. This optimization is common to and can be shared between the C and C++ implementations. The approach depends on the specific processor being used.

A limited number of QLA routines have been optimized for the Pentium 4 processors using SSE instructions. The new Intel-based clusters at JLab and FNAL can be run in either 32-bit (“IA32”) or 64-bit (x86-64) mode. Preliminary single node testing in the x86-64 mode indicates improved performance in the non-SSE portions of the code. The existing SciDAC-1 QLA library code compiles and runs correctly in the 64-bit environments. The SSE optimizations in QLA can be further improved by taking advantage of the larger register file (16 SSE registers, compared to 8 in the 32-bit mode). We propose to do so.

GNU and commercial compilers now generate SSE code under high optimization levels. Because the compilers are not aware of the registers used by the QLA inline-SSE codes, there can be register conflicts and as a result, incorrect results generated. The QLA SSE codes currently use inline gcc assembler macros. These routines should be rewritten so that register conflicts with compilers no longer occur. Further, the 64-bit environment no longer uses the stack to pass operands and results, but instead uses the larger register files available in 64-bit mode. To simplify the maintenance of the existing QLA SSE codes, the inline assembler macros will gradually be replaced with GNU assembler code.

It is important to optimize key QLA routines for the Opteron processors used in the Cray XT3. Initial experience indicates that the large XT3s at Oak Ridge National Laboratory (ORNL) and the Pittsburgh Supercomputer Center (PSC) are superb tools for the study of QCD, and with its planned upgrade, the ORNL machine has the potential to become the single most powerful computer available to lattice gauge theorists. Although the C version of our codes obtains good performance on the XT3, approximately 800 megaflop/s per processor, the SSE coded routines do not provide the boost in performance seen on Intel processors. Collaboration members are discussing this issue with Cray, and a concerted effort to optimize QLA routines for the Opteron appears warranted. Another reason for investigating Opteron-specific optimizations is that these processors have proven to be very cost competitive, and may become components of clusters built in the LQCD Computing Project. This optimization work is related to the 64-bit work on Intel processors described above.

So far much of the optimized QLA code has been hand written, either directly in assembly code or as the input for an assembly code writing tool such as BAGEL [39]. The rest of the QLA routines are automatically generated in C or C++ code by Perl scripts or expression templates. It would be very beneficial to combine these two steps to facilitate a more rapid optimization of any desired linear algebra module. This requires studying existing techniques for code optimization, such as extending the expression template techniques, and then developing our own tools that will assist in generating optimized code. Initially, we propose to start a feasibility study to automate the generation of QLA by developing a prototype of a generic code generating tool with a back end for the BlueGene/L processor, adding support for more architectures as the technology matures.

**QOP: Optimized Level 3 Routines:** As previously indicated, the bulk of the floating point operations in any lattice QCD calculation are concentrated in a few routines. Because of the very large computational resources involved, it is worthwhile to hand code these routines for the major platforms that will be used by our field. Under SciDAC-1 the primary focus was on inverters for the Dirac operator on Pentium 4 based clusters and the QCDOC. Under SciDAC-2 the work on Level 3 routines needs to be extended in two directions. First, Level 3 inverters need to be written for the three new platforms which we expect to play major roles in our research: clusters based on multi-core processors and Infiniband communications fabrics, and the Cray XT3, BlueGene/L and their successors. Second, for some improved actions, routines other than the Dirac inverter take enough computing resources to warrant Level 3 coding. The fermion force

routine in the Asqtad is a prime example. A Level 3 routine has recently been written for it on the QCDOC, and this work needs to be extended to other platforms and other routines.

## 4.2 Infrastructure for physics applications

The development of a common QCD API had to be performed while preserving the large investment in application codes and maintaining a continuous production environment for applications. There are three large scale, freely available application code suites developed by members of the U.S. lattice gauge theory community:

- MILC: The MILC code is an integrated package of some 150,000 lines of scientific application codes and a library of generic supporting codes. It has been in use and freely available to the public since the early 1990's, and is widely used outside the MILC Collaboration. It is written in C, and can be compiled with either the QMP or MPI message passing libraries. It can be downloaded from <http://www.physics.utah.edu/~detar/milc/>.
- CPS: The Columbia Physics System software begun in 1995 is a comprehensive lattice QCD code primarily used by Columbia, BNL, RIKEN-BNL Research Center and UKQCD lattice theorists. It is written in C++ and targeted for the QCDSF and QCDOC computers. It is also capable of running on clusters, using either QMP or MPI for message passing. It can be downloaded from [http://qcdoc.phys.columbia.edu/chulwoo\\_index.html](http://qcdoc.phys.columbia.edu/chulwoo_index.html).
- Chroma: Chroma is a new application code written entirely in C++ on top of the SciDAC-1 QCP API. It is being developed by JLab along with major U.S. and international collaborations. It can be downloaded from <http://www.jlab.org/~edwards/chroma/>.

### Support of the QCD API

**Integration and optimization of QCD API:** The three major application codes have different designs and application foci. Indeed the basic structure of the QCD API benefited tremendously from the collective experience of the developers of these three different, highly optimized and portable QCD codes. We are in the middle of an evolutionary process of bringing the full benefits of the SciDAC-1 QCD API to them, and propose to accelerate this process under SciDAC-2. This work will include writing additional Level 3 routines callable from all three codes, greater integration of the API into the MILC code, and expansion of Chroma. In addition, we propose to develop a new Physics Toolbox (Level 4), a set of building blocks needed by the entire community to develop new applications and algorithms. Finally, we propose to develop a set of data analysis tools that will enable lattice gauge theorists to handle efficiently the very large data sets they are producing.

**Documentation:** There is clearly a need for additional documentation of both the QCD API and the publicly available applications codes. As the QCD API moves from the development stage of SciDAC-1, where users were either developers or close colleagues of them, to SciDAC-2 with a rapidly expanding user community, the need for adequate documentation is magnified. Some excellent documentation exists for critical components of the API, but a uniform and complete set is now urgent. Similarly, each of the application codes listed above has a large, highly distributed user community. As these communities grow well beyond the groups that developed the software, the need to upgrade the existing documentation does as well. Three levels of documentation are needed for each software component: documentation for installation, a user's guide for running the software, and a developer's guide for extending the software.

Documentation is a necessary part of releasing quality software. Unfortunately it is also burdensome to write, is difficult to maintain (especially in the case of highly distributed development) and requires substantial expertise (the author of the documentation has to actually know the software components quite well in order to give an accurate description). We propose to undertake a substantial upgrade of the documentation of both the QCD API and the application codes under SciDAC-2.

**Testing:** We propose to build a comprehensive test framework for all of the QCD API libraries. Testing frameworks in general and test driven development in particular tend to produce cleaner code and reduced coupling between software components. A test environment is needed for nightly builds for codes directly

from the source code repository and should target many different architectures, such as single node workstations, clusters and the QCDOC. The test system should also provide a (nightly) regression test framework to insure correctness. Finally, API tests that verify implementation can determine, among other things, whether calls to optimized Level 3 routines reproduce those to standard C or C++ code.

## **QCD Physics Toolbox**

We propose to construct a QCD physics toolbox which will contain a set of basic software building blocks and tools to aid in the development of application codes, algorithm studies and data analysis. Our objective is to enable users to focus on physics by minimizing the coding effort needed to explore it. We believe that this toolbox has the potential to greatly expedite the development of new application codes and new algorithms.

**Shared algorithms and Building blocks:** A considerable amount of software can be shared among applications. This includes commonly used routines for reunitarization, gauge fixing, the evaluation of low lying eigenvalues of the Dirac operator, and a host of measurements. It also includes more specialized and intricate routines, such as the determination of the fermion force for improved actions with non-nearest neighbor gauge links, and important new algorithms, such as RHMC. We propose to collect these into the toolbox, from which they can be called by any application code that conforms to the QCD API. A few of the tools that have substantial impact on performance, such as the fermion force in the Asqtad action, should be coded at Level 3, but most can be coded directly in C or C++ on top of the Level 2 QDP/QDP++ interface. We designate this new set of common building blocks and algorithms Level 4 to distinguish them from the relatively few instances in which it is worthwhile to write hand-coded Level 3 routines.

In addition to its role in expediting the development of application code, the Level 4 software will provide important support for rapid exploration and testing of new algorithms. A persistent problem in algorithm research is that to test the efficacy of a new approach often requires simulations and benchmarking on systems of a size used in state of the art of the physics calculations. Thus, the ability to rapidly produce high performance code to test new algorithms is extremely valuable. We therefore propose to build Level 4 routines that will further enhance the ability of the QCD API to support algorithm research.

**Graphics and Visualization:** Another component of the Level 4 toolbox will be a set of graphics routines that can be called from code that conforms to the QCD API standards. Lattice QCD computations comprise multiple steps, creating very large datasets, but the final result is typically encompassed in a small set of numbers with the analysis performed in an automated way. While an automated procedure may be beneficial in efficiency, the ability to visualize the data being analyzed is important both as an aid to the analysis, and as a means of acquiring insight into the physics. Visualization of lattice data has already provided important insights into QCD: pictures of the four-dimensional action densities and topological charge have revealed the complexities and structure of the QCD vacuum, the energy densities between a heavy quark and anti-quark, and between three heavy quarks, have shown the emergence of flux tubes.

Crucial to the success of the graphics-visualization initiative will be a close collaboration between physicists to devise and interpret visualization of physically important quantities, and computer scientists to provide the appropriate visualization toolbox. Questions that visualization might address are many: can we understand how flux-tube formation observed with infinitely heavy quarks extends to hadrons where one or more of the quarks is light; what is the distribution of charge within a nucleon; can we display the distribution of spin and magnetism within a hadron? In the longer term, can we visualize the interactions of hadrons?

Currently, no general-purpose package is available tailored to the display of lattice data. Thus a software package will be developed with a general GUI capable of reading a set of four-dimensional lattice quantities, and taking their ensemble average; performing a projection into a real four-dimensional vector; interpolating the 4-D vector into a continuous four-dimensional field; taking three-dimensional slices of a four-dimensional field; displaying the data using density plots, iso-surfaces, and 2-D projections; and displaying the evolution of data, both in simulation time for four-dimensional quantities, and as the evolution of three- and two-dimensional slices in the remaining coordinates.

The software will support two types of plug-ins: type-1 plug-ins that perform specific physics measurements and output a real 4-D vector, and type-2 plug-ins that take the interpolated 3-D field and generate specific types of plots.

Most of the research underlying this project will consist of identifying a set of physical measurements suitable to be implemented as type-1 plug-ins. The visualization techniques for the type-2 plug-ins are very

similar to standard techniques used for representation of 3-D geophysical data and, when possible, we will incorporate existing libraries into the development of our plug-ins.

The system will be developed in C++ and take advantage of existing graphics and visualization libraries such as the Trolltech QT libraries and the Visualization Tool Kit (VTK) library. The plug-ins will be callable from C or C++ code conforming to the QCD API, and will form another component of our Level 4 QCD Toolbox. The system will be capable of reading datasets in the SciDAC/ILDG format and the MILC format.

**Workflow and Data analysis:** Data processing for lattice QCD is carried out via analysis campaigns. An analysis campaign consists of an input dataset (e.g., an ensemble of gauge configurations) and a set of interdependent processing steps (e.g., the generation of valence quark propagators, and the resulting measurements via two- and three-point correlators) that can be expressed as a directed acyclic graph (DAG). This DAG can be considered to be the workflow specification for the analysis campaign. Given the complexity of current lattice QCD analysis campaigns, which can involve hundreds of input files and thousands of intermediate and final files, it is very desirable to more closely manage these workflow specifications, and to use them to automate many aspects of executing analysis campaigns.

We propose to define a subsystem that allows workflow to be specified in a domain-specific way and later be turned into a set of instructions that can be carried out or executed on a lattice QCD compute platform. Execution includes configuration, submission, progress tracking, and accounting of an analysis campaign. It also includes input staging and storage of results. We also propose to develop a coherent data analysis package for the toolbox.

**Performance analysis:** As the size and complexity of the emerging high-performance computing (HPC) systems continue to grow, it is increasingly difficult to achieve a high fraction of peak performance for lattice QCD applications. Multi-core processors, complex memory hierarchies, multiple processor nodes, and complex software stacks all contribute to this difficulty, exacerbated by the rapid scaling of systems to tens of thousands of processors. To better understand lattice QCD code performance and to exploit HPC systems, we propose a set of performance studies, which will be led by the University of North Carolina computer scientists in our collaboration. Emphasis will be on: 1) performance of new generation SciDAC codes; 2) the impact of modern architectures; and 3) novel techniques for performance study using visualization.

We will develop a profiling library for QDP routines. Similar to the PQMP library, a QMP profiling library developed in SciDAC-1, the PQDP will intercept calls to QDP functions during execution and capture the performance data for such functions. It will record total time duration and the time spent in communication for each QDP call. The goal is to reveal the communication overhead for the QDP routines and to improve the overlapping of computation and communication in these routines.

We will also extend C++ support in SvPablo and conduct performance analysis for Chroma code. We will apply SvPablo and our profiling tools to carry out detailed performance studies for Chroma on various HPC systems, as was done for the MILC code, and make cross-platform performance comparisons. Furthermore, we will compare the performance of the same physics kernels running in both MILC and Chroma based on various performance metrics, and optimize the performance of these codes using SvPablo and other performance analysis tools that are being developed at the Renaissance Computing Institute.

**Multiscale algorithm collaboration with TOPS:** If past history is a guide, new algorithms will in the long run be as important as faster hardware in advancing research in lattice QCD. Thus, a small but essential part of developing infrastructure for the petaflop/s era should include research into fundamentally new algorithms. Indeed, one benefit of increased computational power is that by exposing more details of the short distance physics, it expands the opportunities for the use of multi-scale methods. We have begun to explore multi-level methods for QCD in collaboration with the TOPS multi-grid algorithm team.

In the early 1990's, a number of attempts were made to introduce multi-scale algorithms to QCD [40, 41, 42], which resulted in substantial theoretical progress, but failed for the most part to produce significant advantages for actual QCD simulations. However, members of our software team have recently begun working with applied mathematicians from the TOPS ISIC on this problem, and have obtained impressive preliminary results [43] using a new class [44] of adaptive algebraic multi-grid tools. It is important to continue this work, as even modest gains in performance would have a major impact on the science. In addition, Lüscher has recently introduced a blocking method based on the Schwarz alternating procedure that also shows promise. This approach too warrants further exploration. We therefore plan to continue our work on multiscale algorithms with applied mathematicians in the TOPS ISIC. To facilitate this effort we

propose to extend the capabilities of the QCD API by developing an infrastructure for multi-scale algorithms. In particular we will begin to develop at Level 4 a set of methods for rapid prototyping of multi-level algorithms. The new objects in C++ will be built on top of the QDP++ Level 2 that allows a concise paradigm to express parallelism, domain decomposition, etc.

### 4.3 Uniform computing environment

The Department of Energy is funding a set of terascale computers dedicated to the study of lattice QCD. These machines are being located at BNL, FNAL and JLab. There is considerable value in providing their users with a common development and job execution environment. Just as the SciDAC QCD API aims at application code portability, the uniform computing environment aims at portability within the users' working environment. This is not only a convenience, but it offers the potential to improve overall efficiency by optimizing the mix of jobs on the different architectures at the three laboratories.

#### Common runtime environment

One of the outputs of the SciDAC-1 lattice QCD project was the specification of a draft Common Runtime Environment [45]. This specification covers file system naming and access, the interactive environment, the batch script environment, and the parallel execution environment. In SciDAC-2 we propose to implement the QCD Common Runtime Environment at each laboratory, and to enhance and develop tools to support this specification, with a particular emphasis on meta-facility operations.

**Data management:** We will select or develop tools in the following areas: (1) File staging to and from the computational resource, including tools to split a single, lattice oriented file into multiple parts, and re-assemble a split (parallel) file into a single file; (2) Local file management, including migration of files to and from tertiary storage, and pinning and unpinning files; (3) Grid file management, including uploading meta-data extracted from a lattice standard file to the meta-data catalog, and domain specific graphical and command line meta-data catalog query tools. The latter will extend to retrieval and queries on the International Lattice Data Grid (ILDG).

**Computational grid:** Lattice QCD jobs have domain specific features which make them adjustable onto various sized parallel machines. The QMP library allows an executable to determine the size and shape of the machine on which it is running, but current batch and grid tools do not do a good job of expressing this task flexibility. Realizing this flexibility will require these developments: (1) Develop (or extend) an XML schema to describe the job's optimal machine and range of flexibility in machine parameters; (2) Modify an existing batch system scheduler, or develop a pre-processor, to deal with the flexibility in machine size, while preserving standard batch system properties (fair share, accounting, etc.). (3) Extend this onto a grid environment, with late binding to a particular resource (as opposed to the more common immediate match/binding to the currently least busy matching resource). Throughout this sub-task, efforts will be made to exploit mature grid technologies as building blocks for the lattice meta-facility.

**Monitoring and controlling large systems:** Lattice QCD jobs are composed of long-running, interdependent tasks. Failure of a single processor can halt progress on all processors assigned to a given job. When hardware failures occur, an application-specific set of tasks are done, such as killing the job and restarting from a checkpoint. When done manually, this approach is expensive, slow to respond, and limits scalability. We propose to develop an automated fault monitoring and mitigation system to perform these "babysitting jobs". This "Cluster Nanny" should have the following properties: (1) It should be coupled to the application. Mitigation actions depend on the properties of the application and its overall workflow. (2) It should closely monitor performance and the status of jobs, and work together with a workflow subsystem to ensure good progress for the larger analysis campaign that is being conducted. (3) It should trigger workflow re-planning, to allow for resource optimization, as components fail. This will include interactions with real-time scheduling systems. (4) It should monitor the health (performance, utilization, state) of all processors and networks in the system. In addition, tools will be developed to define the operations of lattice QCD systems and their associated fault mitigation actions. These tools will also analyze the systems and help identify single points-of-failure and resource bottlenecks.

**Software for emerging hardware:** As discussed in Section 5.1, this project will include the acquisition and testing of prototype hardware supporting the large procurements to be undertaken by the DOE Lattice QCD Computing Project. These hardware prototypes will include new processors and motherboards, as well

as high performance network fabrics. Software development will be necessary for the evaluation of these prototypes. Such development will include low level drivers, instrumentation for performance profiling, and the porting of hardware specific portions of the SciDAC lattice QCD libraries.

**Accounting tools:** A final part of the common user environment is the users' interaction with accounting systems. In the initial years of this project, users will perceive multiple accounting systems (one per site), and will likely have site specific allocations. By the third year of the project, as portable jobs are executed on the meta-facility, it will be helpful to users to have a single meta-facility allocation and view. This will require the development of a few simple tools to extend the single site accounting tools to cover multiple sites in a fault tolerant manner. One technology option is to grid enable QBank, a companion to the Maui scheduler used at both FNAL and JLab, which presents an abstraction of accounting to the scheduler.

#### 4.4 Software task schedule

The scheduling of software tasks is given in the Gantt chart of Fig. 2. The budget requests support for 15.7 FTE per year for the software effort. This is broken down among the three main divisions of software work as follows: 4.9 FTE for Machine Specific Software, 7.7 FTE to support Infrastructure for Application Code, and 3.1 FTE for Uniform Computing Environment. These resources are nearly doubled by the contributions of physicists and software engineers at the participating institutions with no direct SciDAC support. In Appendix A.3 we briefly describe the tasks to be undertaken, the major milestones for the first two years,

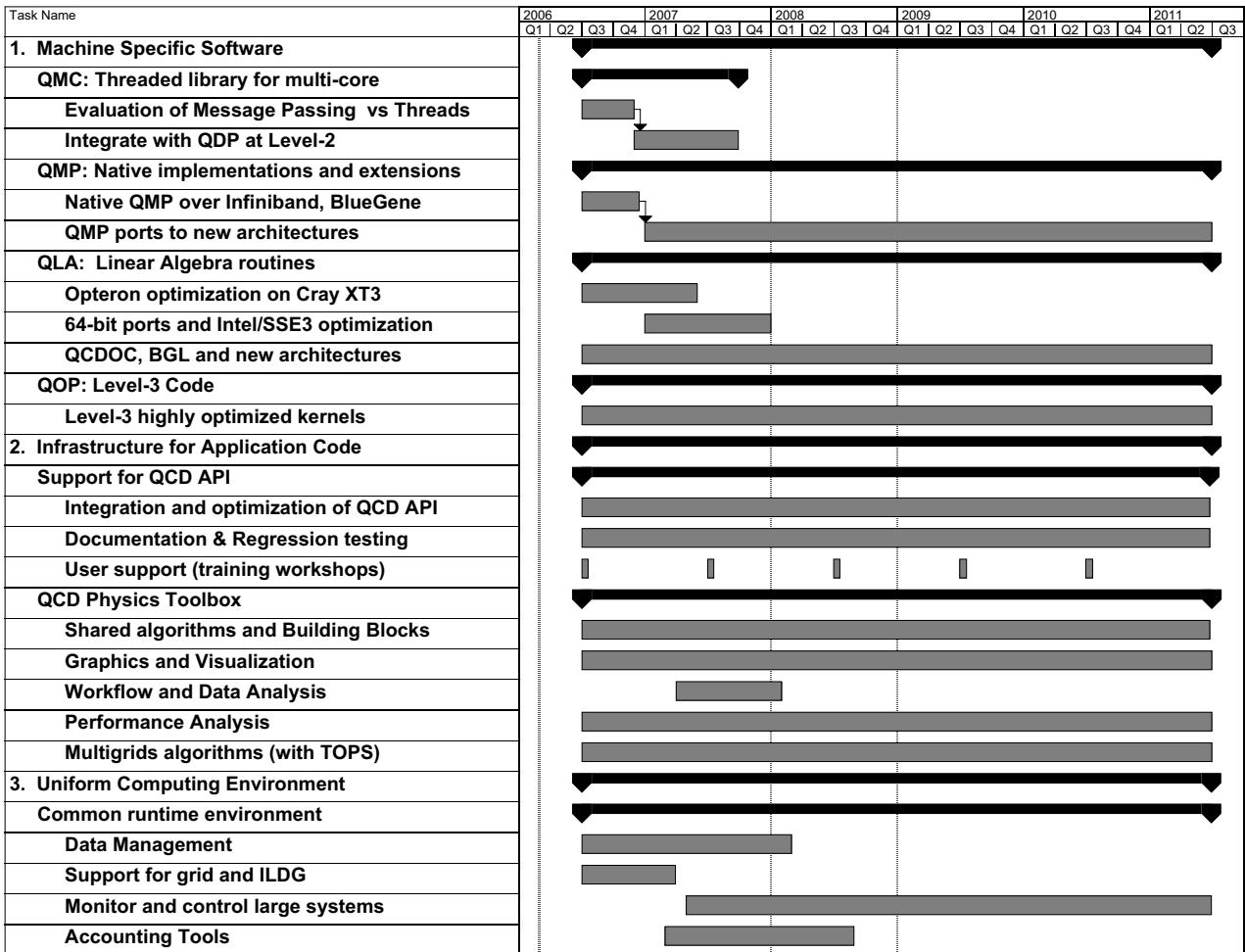


Figure 2: The schedule of software tasks.

and the FTE assignments for each participating institution.

As indicated in the Gantt chart, our software project involves both near term milestones to provide time critical software components, and long term development tasks that are to be pursued on a continuous basis throughout the grant. Important software milestones to be achieved during the first year of the grant include: (i) Design of a multi-core library interface (QMC), and the evaluation of its performance relative to simply treating each core as a separate processor. If the decision is to go forward, then the software will be developed and integration with QLA and QDP will begin; (ii) Port of QMP to Infiniband and the BlueGene/L toroidal network; (iii) Optimization of essential components of QLA for the AMD Opteron and the IBM dual core PowerPC processor; (iv) Conversion of the basic modules of the MILC code to QLA/C, introduction of templates into CPS and their restructuring in Chroma; (v) Identification of physical attributes to be visualized, cataloging of relevant data sets and design of a prototype interface; (vi) Implementation of a basic common runtime environment; and (vii) Initial design work on the workflow software and the fault monitoring and mitigation system.

The tasks that we will pursue over the full five years of the grant are equally important. They include the continuous adaptation of the low level API (QMC, QMP, QLA) to new architectures; the integration of the higher level API (QDP) into the major, freely available application codes; the definition and expansion of the common QCD physics toolbox; the design of more sophisticated algorithms with critical components optimized; and the documentation, regression testing and distribution of software libraries. As the project proceeds critical evaluations of the cost/benefit of each task may substantially alter priorities and allocation of our FTE resources. The metrics for success are a continued growth in the number of application codes written in the ACD API, the performance of these codes, and the number of physicists who use the QCD API to obtain important research results. The growth in the user community is already very encouraging.

## 5 Hardware Research and Development

Over the past twenty years computers specifically optimized for lattice QCD have achieved record price-performance and provided platforms for frontier physics calculations. From the Caltech Cosmic Cube to the QCDOC and the SciDAC-1 clusters, technological opportunities have been exploited to provide economical, large-scale simulation capabilities. These and similar activities carried out in Germany, Italy, Japan, the UK and the US, have also played an important role in the development of commercial supercomputers. For example, the QCDSF and QCDOC machines developed and demonstrated the architecture that is now the basis for the highly successful IBM BlueGene computers.

Work under our SciDAC-1 grant demonstrated that commodity clusters, optimized for QCD can be powerful, highly cost-effective research tools. The SciDAC-1 grant also had a large impact on the use of more specialized QCD computers, making the QCDOC machines usable by those in the U.S. lattice QCD community outside of the group of machine designers and close collaborators. We propose to continue to track the evolving commodity and semi-commodity marketplace in order to provide vital input for the parallel Lattice QCD Computing Project, and to undertake design of a fully customized successor to the QCDOC.

### 5.1 Investigation of Cluster Components

**Background:** The SciDAC-1 project undertook investigations of commodity hardware for lattice QCD. By selecting the most cost effective and appropriately balanced combinations of processor and network interconnect, as opposed to the products which individually had the best performance, and by taking advantage of the modest requirements for memory size and disk bandwidth, the SciDAC-1 project built large scale clusters dedicated to lattice QCD calculations with better price/performance than any existing general purpose parallel computing platform. The processors investigated during the project included Intel Pentium, Xeon, and Itanium, AMD Athlon and Opteron, DEC Alpha, and IBM PPC970. The project also investigated several high performance networks, including Myrinet, gigabit ethernet meshes, and Infiniband. Each year the most promising technologies were chosen to build prototype production clusters listed in Table 2.

The DOE Lattice QCD Computing Project (the “facilities project”), which started October 2005, will procure and operate large scale systems. This project is planned for FY2006 through FY2009, with funding of

Site	Cluster	Processor	Network
JLab	2m	Xeon (single)	Myrinet
JLab	3g	Xeon (single)	3D GigE
JLab	4g	Xeon (single)	5D GigE
JLab	6n	Pentium (dual core)	Infiniband
FNAL	w	Xeon (dual)	Myrinet
FNAL	qcd	Xeon (dual)	Myrinet
FNAL	pion	Pentium (single)	Infiniband

Table 2: Prototype production clusters built under SciDAC-1.

\$9.2 million from the High Energy and Nuclear Physics Programs of the DOE. Approximately \$6 million of this funding will be used for commodity hardware, specifically clusters in the first year, and most likely clusters for the subsequent three years. The designs of the first clusters to be built by the project in 2006 are derived directly from the prototype clusters assembled during the SciDAC-1 project. Continued hardware prototyping by the SciDAC-2 project will provide critical information for the platforms to be procured by the facilities project in FY2007–FY2009.

The prototype clusters from the SciDAC-1 project have proven to be very successful in delivering physics results. Operation for physics production of many of these clusters, specifically the 3g, 4g, and 6n clusters at JLab, and the qcd and pion clusters at FNAL, is now part of the facilities project. These clusters have an aggregate capacity of nearly two teraflop/s.

**Prototyping Tasks:** To support the cluster or other commodity or semi-commodity designs of the facilities project, investigations of commodity processors, chipsets, and high performance networks will be performed. These investigations will focus on those aspects most important to lattice QCD codes: memory bandwidth, floating point processing, and network performance.

During the next two years, vendor roadmaps indicate a number of technology changes which could have important benefits for lattice QCD computing. All of the principal commodity processor vendors have started the move toward multi-core designs. In 2006, dual core processors will very likely take the performance lead over single core designs, and will certainly take the lead in cost effectiveness, based on prototyping with Intel dual core processors at the end of the SciDAC-1 project. By 2007, quad core processors will be introduced.

The use of multiple processing cores will increase the floating point capabilities of commodity systems. In order to be cost effective for lattice QCD calculations, this increase in capability must be balanced by concurrent increases in memory bandwidth and improvements in network latency and bandwidth. In 2006, Intel will introduce the first systems with chipsets supporting the new fully buffered DIMM (FBDIMM) technology. Also in 2006, AMD is expected to change the integrated memory controllers on their Opteron processors to support DDR2 memories. Both of these design changes should provide significantly improved memory performance.

In addition to these mainstream processor developments, it will be valuable to track the evolution of other novel architectures, such as the Cell processor or a successor to the BlueGene/L machine. Because of the high computational capacity of these platforms, there is a possibility that either will prove to be an even more cost effective platform than clusters.

The network performance of commodity computers depends upon the input-output (I/O) bus design and on the network interfaces attached to the I/O buses. The clusters to be constructed in 2006 by the facilities project will use PCI Express (PCI-E) I/O buses and Infiniband network interfaces. Important alternatives to these buses include the higher speed PCI Express 2.0 specification, expected to appear in products in 2007, and the HyperTransport (HTX) bus, which is only available on some Opteron processor motherboards. Important alternatives to the 10 gigabit per second signal rate Infiniband fabrics used on the 2006 clusters include double and quad data rate Infiniband (16 gigabit/sec and 32 gigabit/sec data bandwidths), Infinipath, Myrinet, and Quadrics.

During each year of the SciDAC-2 project, the project will select the most important commodity or semi-commodity technologies to evaluate, that is, technologies showing the greatest promise for accelerating Lattice QCD cost effectiveness and performance. These might include new processors, new network technologies, or even novel architectures such as the Cell processor. This project will acquire modest amounts of hardware to support the porting of significant lattice QCD kernels to evaluate and optimize performance and to help determine the optimal designs for subsequent facilities project machines. The prototyping and evaluation tasks will be shared by JLab and FNAL in a complimentary way. For example, during the first year of the grant FNAL will investigate the latest AMD Opteron systems and the Pathscale Infinipath interconnect, while JLab will study the Intel dual core “Woodcrest” processor, and double data rate Infiniband fabrics. This approach leverages to the greatest extent the capabilities of these two labs, and helps to maintain the expertise needed by the facilities project. The proposed budget for hardware for each of JLab and FNAL is \$40,000 per year; this is sufficient to acquire one or two small machines of sufficient size (8 or 16 nodes) to test I/O capabilities while communicating in multiple dimensions. The level of effort at each site will be 0.25 FTE.

## 5.2 Specialized computers for lattice QCD

**Background:** Over the past twenty years very substantial cost-performance benefits have resulted from the construction of computers specifically designed for lattice QCD. These performance and cost advantages have been achieved by utilizing special chips, often from the graphics market, inter-node communications strategies not available in standard commercial systems and integration/packaging targeted at the specific scale and operating environment for such QCD machines.

In order to evaluate whether special purpose machines continue to offer significant physics opportunities, we must consider the expected alternatives. The clusters planned for the LQCD Computing Project are expected to sustain several teraflop/s on production code, with scalability to tens of teraflop/s (funding constrained). On the high end, hundred-teraflop/s performance is available from BlueGene and Cray machines, and, given their success, these commercial machines should aggressively advance to the petaflop/s level. However, enormous scientific potential lies at the petaflop/s scale, and it is unlikely that in the next five years lattice QCD research budgets will approach the \$100M level required to obtain dedicated computers of this scale. The lesson of the past twenty years is that innovative exploitation of trends in microelectronics, driven by the scientific imperatives of a clean, fundamental problem like lattice QCD, can offer substantial rewards at the frontier of particle and nuclear physics, and can add an important ingredient to the general advance of scientific computing.

Of course, this requires that our proposed project break new ground. Thus, we plan to begin with wide-ranging study of a number of possible directions. Presuming that a compelling approach is identified, we will then proceed to detailed design and prototype construction—activities supported in part by this SciDAC proposal. A follow-on proposal for the construction of a large-scale machine would fall outside of the SciDAC program and would be made to the base programs in High Energy and Nuclear Physics. Such a large-scale proposal would be enabled by this SciDAC-supported research and development effort, and would be driven by the scientific opportunities offered by the resulting computer.

**Overall strategy:** In order to justify the effort and expense associated with this design and prototyping effort, and the risks associated with the construction of a large-scale machine, substantial benefits in cost-performance must be offered by such a project. Expecting a project of this sort to require 4–5 years for completion, we must identify a direction that can yield a substantial enhancement over the expected cost-performance of commercial machines or clusters available in this time frame. Their cost performance might be optimistically predicted by applying Moore’s law to the \$1 per Mflop/s (sustained) in 2005 by a SciDAC-1 cluster, which over five years yields \$0.1 per Mflop/s using an 18 month halving period. Thus, an appropriate target for this design effort is an order of magnitude better at that time, \$0.01 per Mflop/s.

This level of cost-performance would make sustained petaflop/s available for lattice QCD in the next 4–5 years, a goal with very substantial scientific rewards. Realizing such a goal will require advances in a combination of hardware and software technology, and, very likely, a further degree of specialization in the resulting machine. Since there are scaling factors in the computational cost of generating gauge configurations which do not appear in the calculation of quantum observables on those configurations (associated

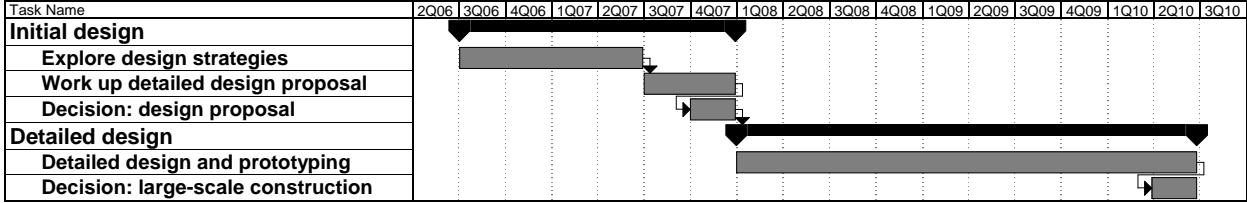


Figure 3: Proposed schedule for the design work for a next QCD machine.

with autocorrelation time and evolution step-size), the overwhelming computational needs of configuration generation will grow proportionally larger as smaller quark masses and finer lattice spacings are achieved. The generation of such gauge configurations is typically done with highly optimized code that is held stable and run for 1–2 years or more.

Thus, large scientific benefit may be realized by an innovative machine whose floating point and memory architecture yield very high performance for carefully optimized code. Even if efficient code generation for such a machine is outside the reach of current compiler technology, existing software tools (for example Peter Boyle’s BAGEL code generator) demonstrate that an effective programming environment can be provided to expert users for such a machine. Thus, in order to achieve the substantial performance boost that our scientific goals demand, we should consider such architectures.

**Technological opportunities:** There are at least three promising directions for which we propose further study and straw man designs. The first uses a commercial “superchip” with high floating-point performance and external memory bandwidth. The SONY/IBM CELL processor, Broadcom’s BCM1480 4-core chip and ClearSpeed’s 50 gigaflop/s CSX600 are current examples. To be useful for QCD these chips require the design of a communications companion chip that would provide mesh communications, perhaps in 6-dimensions, and an interface to commercial memory of appropriate size and cost. The second approach exploits advances in small, low-power DSP-like cores, for example the Cortex-A8/NEON of ARM, to create a 64- to 128-processor QCD chip with substantial on-chip memory. This would be a system-on-a-chip design similar to QCDOC but more aggressive in complexity and power management. The third approach uses a very large number of small chips. These small chips could be of our own design with a single processor, multiple floating point cores, simple memory interface and support for mesh communications. Here the power and space drawbacks of such an approach may be compensated by the cost and simplicity of the chip design and the reduced requirements for the memory interface. Alternatively, this “small” chip could be a future multi-core, low-power Intel mobile chip with a specially designed companion communications chip.

As anticipated above, none of these three approaches would provide the full, integrated RISC-floating point processor environment present in the current QCDOC design. However, a price-performance point of  $\approx \$0.01$  per sustained megaflop/s may be possible with these three directions. Here we propose to develop the detailed technical designs to explore whether this is indeed possible.

**Project plan:** Beginning with the start of this proposed SciDAC-2 grant, we will undertake the design study outlined above. During the first year and one-half, the three technologies described above would be investigated, and one or more carried to the point of a detailed technical design whose performance, cost and risks could be reasonably established. During the final quarter of that period, this design will be reviewed by a committee appointed by the Lattice QCD Executive Committee. With input from this review, the Executive Committee will decide whether to proceed to actual design and prototyping. If a compelling proposal is made and accepted, then actual design and prototyping work would begin, culminating in a substantial prototype in two and one half years. Based on the performance of this prototype and the potential of the design to advance research in QCD, the Executive Committee will decide whether to develop a proposal to construct a large-scale machine. This design/decision flow is indicated in Gantt chart of Figure 3.

This project is of interest internationally, and we will encourage strong foreign groups with expertise and similar interests to work with us. During the first and second years, Columbia University, the lead institution for this project in the U.S., and Brookhaven National Laboratory, will coordinate these planning and design activities with the University of Edinburgh, the RIKEN BNL Research Center and Regensburg University.

From these five institutions, and possibly others which may join, we will attempt to form a design team of 5-6 principal members, and, as part of this proposal and others being made to RIKEN, PPARC and German funding agencies, we plan to add 3-4 postdoctoral-level participants.

For the first year, the costs of the proposed design effort are entirely personnel. The development of the detailed design is planned to include the procurement of development hardware, design and simulation software, as well as non-recurring engineering (NRE) costs associated with ASIC, printed circuit board and cabinet design. Funds cover a postdoctoral-level physicist dedicated to this activity at Columbia, as well as hardware, software and NRE costs. Anticipating support obtained by our possible collaboration partners in this design (RBRC and Edinburgh), costs of these final items are one-third of the total estimated on the basis of our earlier experience. These costs will become more precise as the work proceeds, and the level of support obtained at our collaborating institutions becomes known. Funding for a large-scale prototype that would be constructed at the end of this project is not included in this proposal, and will be sought outside of the SciDAC program. A final (5th year) of postdoc support is included to provide continuity between the design effort and construction of a large-scale machine.

While this effort to reach a sustained performance of \$0.01/Mflop/s (\$10M/sustained petaflop/s) entails considerable risk, the scientific rewards of providing this level of computational power to lattice QCD, and the influence of such a project on overall scientific computing make this a compelling direction to explore. This proposal is structured to allow the exploration to be carried out rapidly at minimal risk. The close integration of this project with the overall effort of the U.S. lattice QCD community and the software development activities described elsewhere in this proposal ensure that if this project is successful it will be of immediate and substantial benefit to the entire U.S. lattice QCD research effort.

## 6 Management Plan and Budget Narrative

Overall responsibility for this effort will be vested in the Lattice QCD Executive Committee, whose members (R. Brower, N. Christ, M. Creutz, P. Mackenzie, J. Negele, C. Rebbi, D. Richards, S. Sharpe, and R. Sugar) will serve as Principal Investigators. The Executive Committee sets the project's goals, and draws up plans for meeting them. It determines priorities, decides on the distribution of funds, and ensures that work is completed on schedule. At the end of each project year, it develops a rolling two year road map for specific tasks. Tasks, milestones for the first two years, and initial FTE assignments for each institution are summarized in Appendix [A.3](#). Schedule slips of more than two months must be reported to the Executive Committee, which will then decide if a reallocation of resources or a scope change is needed. The Executive Committee has been leading the effort to construct computational infrastructure for the U.S. lattice gauge theory community for over seven years. It holds approximately two conference calls per month, and communicates via email between calls. A consensus has been reached on nearly all issues that have come before the Executive Committee. When consensus is not reached, decisions are made by majority vote, with the Chair's vote deciding the outcome in case of a tie. The Chair of the Executive Committee, Robert Sugar, serves as spokesperson and principal contact with the Department of Energy. Each institution receiving funds under this grant has a local principal investigator, who has first level responsibility for the work carried out at his institution. The spokesperson will submit quarterly reports to the DOE on the progress of the project. He will be assisted in preparing these reports and in tracking the grant budget by Dr. Bakul Bannerjee of FNAL.

The Executive Committee has formed a number of committees to assist it in the management of the project. Their responsibilities are set out below, and a list of members of each committee is given in Appendix [A.4](#).

**Scientific Program Committee:** The Scientific Program Committee monitors the scientific progress of the project, and provides leadership in setting new directions. The Committee organizes an annual meeting of the user community to review progress and obtain input on future directions. It solicits proposals for use of the dedicated computational resources available to the U.S. lattice gauge theory community: the SciDAC-1 prototype clusters, the QCDOC and the computers acquired through the LQCD Computing Project. The Committee reviews the proposals and makes preliminary allocations based on its reviews. It then organizes an open meeting of the user community to discuss the proposals and the preliminary allocations. The Committee makes final allocations following this meeting. The objective of this process is to achieve the greatest scientific benefit from the resources through broad input from the community.

**Software Coordinator and Software Coordinating Committee:** The Software Coordinator, Richard Brower, has overall responsibility for the software effort, providing direction and coherence to the work, and monitoring progress on all tasks. The Software Coordinator provides quarterly reports for the Executive Committee on the progress of the software effort.

The Software Coordinating Committee works with the Software Coordinator to provide overall leadership of the software effort. The Committee meets weekly in conference calls to track progress of software tasks, to discuss technical approaches for completing them, and to further clarify the tasks. The Software Coordinating Committee works in consultation with the Executive Committee to insure that critical components are available in time to keep the overall software and hardware infrastructure project on track, proposing changes in task priority and schedule to the Executive Committee as appropriate. The Software Coordinator has set up a website, <http://physics.bu.edu/~brower>, on which all agenda, minutes and working documents of the Software Coordinating Committee are posted, and he has also established a mail archive, (qcdapi@physics.bu.edu), for interchange of information among all members of the collaboration.

**Oversight Committee:** The Oversight Committee is charged with reviewing plans and priorities from the perspective of the user community, tracking progress in all aspects of the project, and making recommendations regarding alternative approaches or new directions. It meets via conference calls, which are scheduled so that the Committee can review on going progress, and provide timely advice before important decisions are taken. The Chair of the Executive Committee participates in these conference calls to obtain the advice of the Oversight Committee at first hand. The Chair of the Oversight Committee, Steven Gottlieb, maintains regular contact with all aspects of the project to keep the Committee informed of developments, and to schedule meetings appropriately.

**Management of Hardware Research and Development:** Donald Holmgren will oversee the investigation of cluster components, and Norman Christ the research into the design of a new specialized computer for lattice QCD. As is the case with the Software Coordinator, they will provide quarterly reports to the Executive Committee on the progress in their areas.

The overall effort will be supported by the established management structure at the three DOE laboratories (BNL, FNAL, JLab) that are major participants in the project. The project will benefit enormously from access to the SciDAC-1 clusters, the QCDOC and the computers obtained through the LQCD Computing Project, which are operated by these laboratories. This hardware is available to the entire U.S. lattice gauge theory community. The project will also benefit from the BlueGene/L computers at Boston University and MIT. All software developed under this proposal will be made publicly available, as was the software created under our SciDAC-1 grant. In addition, the large gauge configurations generated in major research projects that make use of the hardware located at BNL, FNAL and JLab will be stored in a common format, and made immediately available to the entire U.S. lattice gauge theory community in order to maximize the physics obtained from these computationally expensive data sets. This data will be made available to the international lattice gauge theory community through the ILDG after the physicists who generate it have had an opportunity to use it in initial calculations.

**Budget Narrative:** The overall budget for the five year project we propose is summarized in Table 3 of Appendix A.2. The overwhelming fraction of the budget is for support of the people who will carry out the tasks discussed in this proposal. Support is requested for a total of 17.2 FTE per year of which 15.7 FTE will go into the software effort. A total of 0.5 FTE per year is requested for the investigation of cluster components, as well as \$80,000 per year to purchase the components themselves. Support for 1.0 FTE per year is requested for the design of a specialized computer, and \$50,000 is requested in the second year of the grant, and \$125,000 in each of the third and fourth years for the procurement of hardware, design and simulation software, and non-recurring engineering costs. All funding in this project is via direct grants to participating institutions. There are no subcontracts or funded consortium arrangements.

## A Appendices

### A.1 References Cited

- [1] The senior personnel will participate in this project or have indicated that they will make use of the infrastructure it creates are listed in Appendix A.4.
- [2] The MILC Collaboration: C. Bernard *et al.*, [Phys. Rev. D](#) **64**, 054506 (2001); [Phys. Rev. D](#) **70**, 094505 (2004).
- [3] The Fermilab, HPQCD, MILC, and UKQCD Collaborations: C.T.H. Davies *et al.*, [Phys. Rev. Lett.](#) **92**, 022001 (2004).
- [4] The RBC Collaboration, Y. Aoki *et al.*, [Phys. Rev. D](#) **72**, 114505 (2005).
- [5] The MILC Collaboration: C. Bernard *et al.*, [Phys. Rev. D](#) **70**, 114501 (2004); [Nucl. Phys. \(Proc. Suppl.\)](#) **B140**, 231 (2005).
- [6] S. Gottlieb *et al.*, [Proceedings of Science \(Lattice 2005\)](#) 203 (2005).
- [7] The HPQCD and UKQCD Collaborations: A. Gray *et al.*, [Phys. Rev. D](#) **72**, 094507 (2005).
- [8] HPQCD, MILC, and UKQCD collaborations: C. Aubin, *et al.*, [Phys. Rev. D](#) **70** 031504(R) (2004).
- [9] The HPQCD Collaboration: Q. Mason, *et al.*, [arXiv:hep-lat/0511160](#).
- [10] The HPQCD and UKQCD Collaborations: C. Davies *et al.*, [Phys. Rev. Lett.](#) **95**, 052002 (2005).
- [11] W.J. Marciano, [Phys. Rev. Lett.](#) **93**, 231803 (2004).
- [12] The Fermilab Lattice, MILC and HPQCD Collaborations: C. Aubin, *et al.*, [Phys. Rev. Lett.](#) **95** 122002 (2005)
- [13] The Fermilab Lattice and MILC Collaborations: C. Aubin *et al.*, [Phys. Rev. Lett.](#) **94**, 011601 (2005).
- [14] The Fermilab Lattice and UKQCD Collaborations: I.F. Allison *et al.*, [Phys. Rev. Lett.](#) **94**, 172001 (2005).
- [15] F. Karsch, [arXiv:hep-lat/0601013](#).
- [16] V.G. Bornyakov *et al.*, [Proceedings of Science \(Lattice 2005\)](#) 157 (2005).
- [17] The MILC Collaboration: C. Bernard, *et al.*, [Phys. Rev. D](#) **71**, 034504 (2005).
- [18] Z. Fodor, S. D. Katz and K. K. Szabo, [Phys. Lett. B](#) **568**, 73 (2003); F. Csikor, G. I. Egri, Z. Fodor, S. D. Katz, K. K. Szabo and A. I. Toth, [JHEP](#) **0405**, 046 (2004).
- [19] C. R. Allton, M. Doring, S. Ejiri, S.J. Hands, O. Kaczmarek, F. Karsch, E. Laermann, K. Redlich, [Phys. Rev D](#) **71**, 054508 (2005).
- [20] P. de Forcrand and O. Philipsen, [Nucl. Phys.](#) **B642**, 290 (2002); M. D'Elia and M.-P. Lombardo, [Phys. Rev. D](#) **67**, 014505 (2003).
- [21] The MILC Collaboration: C. Bernard *et al.*, [Proceedings of Science \(Lattice 2005\)](#) 157 (2005).
- [22] C. Jung, [Proceedings of Science \(Lattice 2005\)](#) 150 (2005).
- [23] P. Petreczky, [J. Phys. Conf. Ser.](#) **16**, 169 (2005).

- [24] The LHC Collaboration: P. Hagler, *et al.*, [Phys. Rev D](#)**68**, 034505 (2003).
- [25] The LHC Collaboration: P. Hagler, *et al.*, [Phys. Rev. Lett.](#) **93**, 112001 (2004).
- [26] C. Alexandrou, P. de Forcrand, Th. Lippert, H. Neff, J. W. Negele, K. Schilling, W. Schroers and A. Tsapalis, [Phys. Rev D](#)**69**, 114506 (2004).
- [27] C. Alexandrou, P. de Forcrand, H. Neff, J. W. Negele, W. Schroers and A. Tsapalis, [Phys. Rev. Lett.](#) **94**, 021601 (2005).
- [28] The LHPC Collaboration: S. Basak, *et al.*, [Phys. Rev D](#)**72**, 094506 (2005).
- [29] The LHPC Collaboration: S. Basak, *et al.*, [Phys. Rev D](#)**72**, 074501 (2005).
- [30] O. Jahn, J. W. Negele and D. Sigaev, [Proceedings of Science \(Lattice 2005\)](#) 069 (2005).
- [31] The LHPC Collaboration: R. G. Edwards, *et al.*, [To be published in Phys. Rev. Lett..](#)
- [32] The LHPC Collaboration: F.D.R. Bonnet, *et al.*, [Phys. Rev D](#)**72**, 054506 (2005).
- [33] The NPLQCD Collaboration: S. R. Beane, P. F. Bedaque, K. Orginos and M. J. Savage, [arXiv:hep-lat/0506013](#).
- [34] The NPLQCD Collaboration: S. R. Beane, P. F. Bedaque, K. Orginos and M. J. Savage, [arXiv:hep-lat/0602010](#).
- [35] K. Jansen, [Nucl. Phys. B \(Proc. Suppl.\)](#), **129**, 3 (2004); T. DeGrand, [Int. J. Mod. Phys. A](#) **19**, 1337 (2004).
- [36] E. Follana, A. Hart and C.T.H. Davies (HPQCD Collaboration), [Phys. Rev. Lett.](#) **93**, 241601 (2004); S. Dürr, C. Hoelbling and U. Wenger, [Phys. Rev D](#)**70**, 094501 (2004); D.H. Adams, [Phys. Rev D](#)**72**, 114512 (2005); F. Maresca and M. Peardon, [arXiv:hep-lat/0411029](#).
- [37] Y. Shamir, [Phys. Rev D](#)**71**, 034509 (2005).
- [38] The RBC Collaboration: Y. Aoki *et al.*, [arXiv:hep-lat/0508011](#).
- [39] P. A. Boyle, <http://www.ph.ed.ac.uk/~paboyle/bagel/Bagel.html>, 2005.
- [40] R. Brower, R. Edwards, C. Rebbi and E. Vicari, [Nucl. Phys. B](#)**366**, 689 (1991).
- [41] A. Hulsebos, J. Smit, and J. Vink, in Proceedings for Fermion algorithms 161, (Juelich 1991).
- [42] T. Kalkreuter, [J. Comput. Appl. Math.](#) **63**, 57 (1995).
- [43] J. Brannick, *et al.*, Proceedings of the 16th International Conference on Domain Decomposition Methods (2005).
- [44] T. Chartier, *et al.*, [SIAM J. Sci. Comput.](#) **25**, 1 (2003).
- [45] <http://lqcd.jlab.org/users/RunTimeEnv.html>.

## A.2 Budget Summary

Table 3 below shows the total budget for each participating institution in each of the five years of the proposed grant. Nearly all of the funds are for the 15.7 FTE working on the software effort. The exceptions are 0.50 FTE and \$80,000 per year in hardware for the cluster component studies at FNAL and JLab; and the funds going to Columbia University for research and development on a special purpose computer. The request for the last item consist of 1.0 FTE per year, plus \$50,000 in the second year and \$125,000 in each of the third and fourth years for the procurement of hardware, design and simulation software, and non-recurring engineering costs.

Institution	FY06	FY07	FY08	FY09	FY10	Total
BNL	407	426	445	460	480	2,218
FNAL	550	569	589	610	631	2,949
JLab	568	585	603	622	641	3,019
Boston U.	176	183	191	198	206	954
Columbia U.	107	161	240	245	124	878
DePaul U.	64	66	68	70	72	340
IIT	30	30	30	30	30	150
Indiana U.	50	51	52	54	55	262
MIT	226	235	244	254	264	1,224
U. Arizona	50	51	53	54	55	263
U. North Carolina	111	113	116	119	122	581
UC Santa Barbara	30	30	30	30	30	150
U. Utah	53	55	56	58	60	282
Vanderbilt U.	74	75	76	76	77	378
<b>Total</b>	<b>2,496</b>	<b>2,630</b>	<b>2,793</b>	<b>2,880</b>	<b>2,847</b>	<b>13,648</b>

Table 3: Institution and Total Budgets in \$1,000

On the following pages of this appendix we reproduce the cover pages, detailed budgets and budget explanations of each of the participating institutions.

### A.3 Tasks and Milestones of Participating Institutions

In this appendix we briefly describe the tasks that will be carried out by each of the collaborating institutions, the FTE budgeted for them, and indicate the major milestones for the first two years of the grant. More detailed descriptions of the tasks can be found in the work statements of the individual institutions that appear below.

**BNL:** BNL will continue to optimize software and implement new algorithms for the QCDOC. It will compile, install and test SciDAC software packages on this machine. BNL will continue the evolution of the Columbia Physics System (CPS) code. It will optimize the CPS for the BlueGene/L, and work on an implementation for the successor to the QCDOC. This work will continue throughout the project, although there will be greater emphasis on QCDOC software during the first three years, and on software for the QCDOC successor in the last two years. A total of 2.5 FTE is budgeted for this work.

**FNAL:** During the first year of the grant, FNAL will port SciDAC code from the Intel 32 bit to 64 bit environment, and will optimize the code for Opteron processors. During year one, it will also explore the approach for and determine the benefit of a native implementation of QMP over Infiniband, and if warranted, create the implementation. In collaboration with JLab and university researchers, FNAL will provide code to support multi-core processors. With computer scientists at Illinois Institute of Technology it will provide software for automated workflow, and with computer scientists at Vanderbilt it will create software to enhance the reliability of large systems. It will implement and/or deploy software to support the ILDG and other grid activities, and provide software support for the evaluation of new hardware. FNAL will work with JLab throughout the project to study commodity hardware for lattice QCD. During the first year of the grant, it will evaluate AMD Opteron processors and Pathscale Infinipath. Finally, it will assist in the overall project management. A total of 3.0 FTE per year is budgeted for these tasks.

**JLab:** In each year of this project JLab will carry out research aimed at improving algorithms and producing high performance code for the study of lattice QCD. During the first year of the project, JLab will focus on implementations and optimizations for multi-core processors and for the Intel/SSE3 architecture, and on support for data analysis activities. It will also expand the existing code testing framework, and provide enhanced user support in collaboration with other institutions via workshops, phone and email. JLab will work with FNAL throughout the project to study commodity hardware for lattice QCD. During the first year of the project, JLab will study the Intel dual core “Woodcrest” processor, and double data rate Infiniband fabrics. A total of 3.1 FTE per year is budgeted for these tasks.

**Boston University:** Boston University provides significant leadership for the project as a whole with Richard Brower serving as Software Coordinator and Claudio Rebbi as chair of the Scientific Project Committee. James Osborn of BU has special responsibility to develop the C implementation of QDP and work with collaborators at Arizona, Indiana and Utah to integrate it into the MILC code. He will also work closely with Andrew Pochinsky at MIT to optimize the QCD API for the BlueGene architecture. Brower and Rebbi are leading the physics side of the collaboration with TOPS to study multigrid methods for lattice QCD. A total of 0.97 FTE per year is budgeted for these tasks.

**Columbia University:** Columbia University will lead an international effort to design and prototype a specialized computer for QCD. During the first year, different design approaches will be studied, and a detailed report prepared describing the results of the study and proposing what is judged to be the best approach. During the second year, this approach will be pursued in greater detail, and a proposal will be submitted to the Executive Committee with a specific architecture, cost and schedule for design and construction. If this proposal is accepted, then the design and prototyping work will be pursued in subsequent years. A total of 1.0 FTE per year is budgeted for this project.

**DePaul University:** DePaul University will lead the design and development of a visualization tool for lattice QCD. Work will be done in collaboration with physicists involved in the project and with computer scientists at the University of North Carolina. The goals for the first year of the project are to identify and catalog the types of datasets to be visualized, identify appropriate smoothing and visualization algorithms, and develop a prototype interface. In subsequent years, plugins will be developed to read in the various types of datasets produced in lattice QCD simulations, and tools for manipulating the data in increasingly sophisticated ways will be created. A total of 1.08 FTE per year is budgeted for this effort.

**University of Arizona, Indiana University and University of Utah:** The MILC code is an integrated package of some 150,000 lines of scientific application code and a library of generic supporting codes, that is publicly available and widely used. Arizona, Indiana and Utah will work together to carry out a major overhaul of this code to exploit the advantages of the SciDAC software. During the first year of this effort, generic code that supports multiple science-specific applications will be converted to QLA/C to take advantage of its platform-specific optimizations. During the second year, key modules will be rewritten in QDP. Optimization and tuning of the RHMC algorithm, which is currently being incorporated into the code, will be carried out. The first production version of the algorithm will be made available by the end of year one of the grant. Production versions of the code optimized for the Cray XT3 and BlueGene/L will be completed during the first year of the grant, and multi-core and enhanced compiler improvements will be incorporated during the second year. As always, upgrades to the code will be made available to the lattice community as they are completed. Finally, improved documentation for the code will be produced and published on the web by the end of the second year of the grant. A total of 1.875 FTE per year is budgeted for this effort, divided approximately equally among the three universities.

**Illinois Institute of Technology:** Computer scientists at the Illinois Institute of Technology (IIT) will build a workflow management system for planning, capturing and executing LQCD analysis campaigns. This work will be done in collaboration with FNAL. During the first two years of the grant, a workflow system will be developed and integrated into the existing LQCD computing infrastructure, allowing users to describe their analysis campaign workflow through XML files or graphical interfaces, and submit them for execution. Next a scheduling system capable of interacting with the workflow system and the system performance monitor will be deployed. The final result will be an integrated workflow environment capable of handling multiple campaigns. A total of 1.083 FTE per year is budgeted for this project.

**MIT:** Andrew Pochinsky of MIT will lead an effort to optimize the QCD API for the BlueGene series of computers. During the first two years, the effort will focus on the BlueGene/L. The gcc compiler will be modified to make efficient use of the two arithmetic units on each processor. The QLA routines will be compiled with this modified compiler, and key routines will be hand optimized as required. A level-3 inverter for domain wall fermions will be written, and in collaboration with James Osborn of Boston University, an optimized version of QMP will be developed. This work will be aided by contract commitments made by IBM as part of the MIT purchase of a BlueGene/L. It is anticipated that in subsequent years these software developments will be extended to later models in the BlueGene line. A total of 0.925 FTE is budgeted for this effort.

**University of North Carolina:** Computer scientists at the University of North Carolina will develop a performance profiling library (PQDP) to analyze the performance of the MILC and Chroma codes during the first year of the project. During the second year, the PQDP will be validated by profiling the MILC code on a variety of HPC platforms, including the QCDOC, clusters, the BlueGene/L and the Cray XT3. In subsequent years the UNC SvPablo performance analysis toolkit will be extended to support analysis of C++ codes so that the PQDP can be used to study Chroma. Performance analysis will be carried out on both codes on a wide variety of HPC platforms, and a web-based performance database will be established. The goal is to optimize the performance of MILC and Chroma based on the collected performance data. Finally, UNC will work with computer scientists at DePaul on the visualization effort discussed above. A total of 0.5325 FTE per year has been budgeted for these tasks.

**UCSB:** As chair of the Lattice QCD Executive Committee Robert Sugar provides overall leadership and coordination of the project. UCSB will administer funds for travel not covered by grants to other participating institutions. These trips will include visits of collaboration members to participating institutions for joint work, and attendance at meetings directly related to the project. UCSB will also administer travel funds for Principal Investigators S. Sharpe and R. Sugar.

**Vanderbilt University:** Computer scientists at Vanderbilt will develop an automated fault monitoring and mitigation system for the large lattice QCD clusters being built at FNAL and JLab. This work will be done in collaboration with FNAL. During the first year, an integrated monitoring and control system will be designed using existing standards and tools. Also during the first year, a tool will be developed for definition of workflows, monitoring and mitigation actions, based on Vanderbilt's Generic Modeling Environment. This task will be closely coordinated with work at IIT. During the second year, model based generators will be developed to transform the designs into components and configurations for the runtime system. In subsequent years, refined versions of these tools will be developed. A total for 1.083 FTE per year is budgeted for this work.

## A.4 Committees and Senior Personnel

In this appendix we list the membership of the committees making up the management team of this project. We also list the senior personnel who will participate in this project, or have indicated that they will make use of the infrastructure it creates. They comprise nearly all of the senior lattice gauge theorists in the United States, as well as computer scientists and engineers who have agreed to participate in the project.

### Lattice QCD Executive Committee

Richard Brower	Boston University
Norman Christ	Columbia University
Michael Creutz	Brookhaven National Laboratory
Paul Mackenzie	Fermi National Accelerator Laboratory
John Negele	Massachusetts Institute of Technology
Claudio Rebbi	Boston University
David Richards	Thomas Jefferson National Accelerator Facility
Stephen Sharpe	University of Washington
Robert Sugar (Chair)	University of California, Santa Barbara

### Scientific Program Committee

Andreas Kronfeld	Fermi National Accelerator Laboratory
Robert Mawhinney	Columbia University
Colin Morningstar	Carnegie Mellon University
John Negele	Massachusetts Institute of Technology
Claudio Rebbi (Chair)	Boston University
Stephen Sharpe	University of Washington
Doug Toussaint	University of Arizona
Frank Wilczek	Massachusetts Institute of Technology

### Software Committee

Richard Brower (Chair)	Boston University
Carleton DeTar	University of Utah
Robert Edwards	Thomas Jefferson National Accelerator Facility
Donald Holmgren	Fermi National Accelerator Laboratory
Robert Mawhinney	Columbia University
Chip Watson	Thomas Jefferson National Accelerator Facility
Ying Zhang	University of North Carolina

### Oversight Committee

Tanmoy Bhattacharya	Los Alamos National Laboratory
Steven Gottlieb (Chair)	Indiana University
Anna Hasenfratz	University of Colorado
Julius Kuti	University of California, San Diego
Robert Pennington	National Center for Supercomputer Applications
Ralph Roskies	Pittsburgh Supercomputer Center
Terry Schalk	University of California, Santa Cruz

## Senior Personnel

Theodore Bapty	Vanderbilt University
Silas Beane	University of New Hampshire
Paulo Bedaque	University of Maryland
Claude Bernard	Washington University
Tanmoy Bhattacharya	Los Alamos National Laboratory
Alan Blatecky	University of North Carolina
Thomas Blum	University of Connecticut
Richard Brower	Boston University
Matthias Burkardt	New Mexico State University
Simon Catterall	Syracuse University
Shailesh Chandrasekharan	Duke University
Jie Chen	Thomas Jefferson National Accelerator Facility
Ying Chen	Thomas Jefferson National Accelerator Facility
Norman Christ	Columbia University
Joseph Christensen	McMurray University
Michael Creutz	Brookhaven National Laboratory
Christopher Dawson	Brookhaven National Laboratory
Massimo DiPierro	DePaul University
Thomas DeGrand	University of Colorado
Carleton DeTar	University of Utah
Shao-Jing Dong	University of Kentucky
Zhihua Dong	Columbia University
Terrence Draper	University of Kentucky
Patrick Dreher	Massachusetts Institute of Technology
Anthony Duncan	University of Pittsburgh
Robert Edwards	Thomas Jefferson National Accelerator Facility
E, Efstathiadis	Brookhaven National Laboratory
Estia Eichten	Fermi National Accelerator Laboratory
Michael Engelhardt	New Mexico State University
George Fleming	Yale University
Balint Joo	Thomas Jefferson National Accelerator Facility
Chulwoo Jung	Brookhaven National Laboratory
Aida El-Khadra	University of Illinois, Urbana
Rudolf Fiebig	Florida International University
Steven Gottlieb	Indiana University
Rajan Gupta	Los Alamos National Laboratory
Anna Hasenfratz	University of Colorado
Urs Heller	Florida State University
James Hetrick	University of Pacific
Ivan Horvath	University of Kentucky
Donald Holmgren	Fermi National Accelerator Laboratory
Xiangdong Ji	University of Maryland
Frithjof Karsch	Brookhaven National Laboratory
Gregory Kilcup	Ohio State University
Joseph Kiskis	University of California, Davis
Julius Kuti	University of California, San Diego
Andreas Kronfeld	Fermi National Accelerator Laboratory
Frank Lee	George Washington University
Peter Lepage	Cornell University

Keh-Fei Liu	University of Kentucky
Paul Mackenzie	Fermi National Accelerator Laboratory
Robert Mawhinney	Columbia University
Colin Morningstar	Carnegie Mellon University
Rajamani Nayayanan	Florida International University
John Negele	Massachusetts Institute of Technology
Shigemi Ohta	KEK and Riken BNL Research Center
Kostas Orginos	William & Mary University
James Osborn	Boston University
Robert Pennington	National Center for Supercomputer Applications
Peter Petreczky	Brookhaven National Laboratory
Andrew Pochinsky	Massachusetts Institute of Technology
Michael Ramsey-Muslof	California Institute of Technology
Claudio Rebbi	Boston University
Daniel Reed	University of North Carolina
Dru Renner	University of Arizona
David Richards	Thomas Jefferson National Accelerator Facility
Martin Savage	University of Washington
Stephen Sharpe	University of Washington
Junko Shigemitsu	Ohio State University
James Simone	Fermi National Accelerator Laboratory
Donald Sinclair	Argonne National Laboratory
Amarjit Soni	Brookhaven National Laboratory
Robert Sugar	University of California, Santa Barbara
Xien-He Sun	Illinois Institute of Technology
Eric Swanson	University of Pittsburgh
Chung-I Tan	Brown University
Harry Thacker	University of Virginia
Anthony W Thomas	Thomas Jefferson National Accelerator Facility
Doug Toussaint	University of Arizona
Ruth Van de Water	Fermi National Accelerator Laboratory
Steven Wallace	University of Maryland
William Watson, III	Thomas Jefferson National Accelerator Facility
Walter Wilcox	Baylor University
Ying Zhang	University of North Carolina

---

**INFORMATION ABOUT PRINCIPAL INVESTIGATORS/PROJECT DIRECTORS(PI/PD) and  
co-PRINCIPAL INVESTIGATORS/co-PROJECT DIRECTORS**

---

Submit only ONE copy of this form for each PI/PD and co-PI/PD identified on the proposal. The form(s) should be attached to the original proposal as specified in GPG Section II.B. Submission of this information is voluntary and is not a precondition of award. This information will not be disclosed to external peer reviewers. **DO NOT INCLUDE THIS FORM WITH ANY OF THE OTHER COPIES OF YOUR PROPOSAL AS THIS MAY COMPROMISE THE CONFIDENTIALITY OF THE INFORMATION.**

---

**PI/PD Name:** Massimo DiPierro

**Gender:**  Male  Female

**Ethnicity:** (Choose one response)  Hispanic or Latino  Not Hispanic or Latino

**Race:** (Select one or more)  American Indian or Alaska Native  
 Asian

Black or African American

Native Hawaiian or Other Pacific Islander

White

Hearing Impairment

Visual Impairment

Mobility/Orthopedic Impairment

Other

None

**Disability Status:** (Select one or more)

**Citizenship:** (Choose one)  U.S. Citizen  Permanent Resident  Other non-U.S. Citizen

**Check here if you do not wish to provide any or all of the above information (excluding PI/PD name):**

**Pecase Eligibility:** N

**REQUIRED: Check here if you are currently serving (or have previously served) as a PI, co-PI or PD on any federally funded project**

---

**Ethnicity Definition:**

**Hispanic or Latino.** A person of Mexican, Puerto Rican, Cuban, South or Central American, or other Spanish culture or origin, regardless of race.

**Race Definitions:**

**American Indian or Alaska Native.** A person having origins in any of the original peoples of North and South America (including Central America), and who maintains tribal affiliation or community attachment.

**Asian.** A person having origins in any of the original peoples of the Far East, Southeast Asia, or the Indian subcontinent including, for example, Cambodia, China, India, Japan, Korea, Malaysia, Pakistan, the Philippine Islands, Thailand, and Vietnam.

**Black or African American.** A person having origins in any of the black racial groups of Africa.

**Native Hawaiian or Other Pacific Islander.** A person having origins in any of the original peoples of Hawaii, Guam, Samoa, or other Pacific Islands.

**White.** A person having origins in any of the original peoples of Europe, the Middle East, or North Africa.

---

**WHY THIS INFORMATION IS BEING REQUESTED:**

The Federal Government has a continuing commitment to monitor the operation of its review and award processes to identify and address any inequities based on gender, race, ethnicity, or disability of its proposed PIs/PDs. To gather information needed for this important task, the proposer should submit a single copy of this form for each identified PI/PD with each proposal. Submission of the requested information is voluntary and will not affect the organization's eligibility for an award. However, information not submitted will seriously undermine the statistical validity, and therefore the usefulness, of information received from others. Any individual not wishing to submit some or all the information should check the box provided for this purpose. (The exceptions are the PI/PD name and the information about prior Federal support, the last question above.)

Collection of this information is authorized by the NSF Act of 1950, as amended, 42 U.S.C. 1861, et seq. Demographic data allows NSF to gauge whether our programs and other opportunities in science and technology are fairly reaching and benefiting everyone regardless of demographic category; to ensure that those in under-represented groups have the same knowledge of and access to programs and other research and educational opportunities; and to assess involvement of international investigators in work supported by NSF. The information may be disclosed to government contractors, experts, volunteers and researchers to complete assigned work; and to other government agencies in order to coordinate and assess programs. The information may be added to the Reviewer file and used to select potential candidates to serve as peer reviewers or advisory committee members. See Systems of Records, NSF-50, "Principal Investigator/Proposal File and Associated Records", 63 Federal Register 267 (January 5, 1998), and NSF-51, "Reviewer/Proposal File and Associated Records", 63 Federal Register 268 (January 5, 1998).

---

**List of Suggested Reviewers or Reviewers Not To Include (optional)**

---

**SUGGESTED REVIEWERS:**

Not Listed

**REVIEWERS NOT TO INCLUDE:**

Not Listed

# COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 04-23  NSF 05-579                    07/21/05					FOR NSF USE ONLY  NSF PROPOSAL NUMBER
FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)  <b>PHY - EDUCATION &amp; INTERDISCIP RESEAR</b>					
DATE RECEIVED	NUMBER OF COPIES	DIVISION ASSIGNED	FUND CODE	DUNS# (Data Universal Numbering System)	FILE LOCATION
				<b>825753379</b>	
EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)  <b>362167048</b>		SHOW PREVIOUS AWARD NO. IF THIS IS <input type="checkbox"/> A RENEWAL <input type="checkbox"/> AN ACCOMPLISHMENT-BASED RENEWAL		IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES <input type="checkbox"/> NO <input checked="" type="checkbox"/> IF YES, LIST ACRONYM(S)	
NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE  <b>DePaul University</b>			ADDRESS OF AWARDEE ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE  <b>1 East Jackson Boulevard Chicago, IL. 606042218</b>		
AWARDEE ORGANIZATION CODE (IF KNOWN)  <b>0016717000</b>					
NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE			ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE		
PERFORMING ORGANIZATION CODE (IF KNOWN)					
IS AWARDEE ORGANIZATION (Check All That Apply) (See GPG II.C For Definitions)		<input type="checkbox"/> SMALL BUSINESS <input type="checkbox"/> MINORITY BUSINESS <input type="checkbox"/> FOR-PROFIT ORGANIZATION <input type="checkbox"/> WOMAN-OWNED BUSINESS		<input type="checkbox"/> IF THIS IS A PRELIMINARY PROPOSAL THEN CHECK HERE	
TITLE OF PROPOSED PROJECT <b>CAREER: Computational Future of Lattice Quantum Field Theories</b>					
REQUESTED AMOUNT <b>\$ 772,524</b>	PROPOSED DURATION (1-60 MONTHS) <b>60</b> months	REQUESTED STARTING DATE <b>10/15/05</b>	SHOW RELATED PRELIMINARY PROPOSAL NO. IF APPLICABLE		
CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW					
<input type="checkbox"/> BEGINNING INVESTIGATOR (GPG I.A) <input type="checkbox"/> HUMAN SUBJECTS (GPG II.D.6) <input type="checkbox"/> DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.C)      Exemption Subsection _____ or IRB App. Date _____ <input type="checkbox"/> PROPRIETARY & PRIVILEGED INFORMATION (GPG I.B, II.C.1.d) <input type="checkbox"/> HISTORIC PLACES (GPG II.C.2.j) <input type="checkbox"/> SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.1) <input type="checkbox"/> VERTEBRATE ANIMALS (GPG II.D.5) IACUC App. Date _____ <input type="checkbox"/> INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES INVOLVED (GPG II.C.2.j) <input type="checkbox"/> HIGH RESOLUTION GRAPHICS/OTHER GRAPHICS WHERE EXACT COLOR REPRESENTATION IS REQUIRED FOR PROPER INTERPRETATION (GPG I.E.1)					
PI/PD DEPARTMENT <b>CTI</b>	PI/PD POSTAL ADDRESS <b>243 S. Wabash Av.</b>				
PI/PD FAX NUMBER <b>312-362-6116</b>	Chicago, IL 606042218 United States				
NAMES (TYPED)	High Degree	Yr of Degree	Telephone Number	Electronic Mail Address	
PI/PD NAME <b>Massimo DiPierro</b>	<b>PhD</b>	<b>1999</b>	<b>312-362-5173</b>	<b>MDiPierro@cs.depaul.edu</b>	
CO-PI/PD					

## CERTIFICATION PAGE

### **Certification for Authorized Organizational Representative or Individual Applicant:**

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 04-23. Willful provision of false information in this application and its supporting documents or in reports required under an ensuing award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflicts which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

#### **Drug Free Work Place Certification**

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Drug Free Work Place Certification contained in Appendix C of the Grant Proposal Guide.

#### **Debarment and Suspension Certification**

(If answer "yes", please provide explanation.)

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency?

Yes

No

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Debarment and Suspension Certification contained in Appendix D of the Grant Proposal Guide.

#### **Certification Regarding Lobbying**

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding \$100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding \$150,000.

#### **Certification for Contracts, Grants, Loans and Cooperative Agreements**

The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, "Disclosure of Lobbying Activities," in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all tiers including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, Title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than \$10,000 and not more than \$100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE	SIGNATURE	DATE
NAME		
TELEPHONE NUMBER	ELECTRONIC MAIL ADDRESS	FAX NUMBER

\*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

## CAREER ELIGIBILITY CERTIFICATIONS

### A. CAREER ELIGIBILITY CERTIFICATION

To be eligible for a CAREER award, you must meet the CAREER eligibility requirements as defined in the CAREER Program Solicitation (also refer to the CAREER FAQ for further explanations). To certify your eligibility, complete each section of the CAREER checklist below. The CAREER Eligibility Certification checklist will be included as part of the proposal and will be sent to reviewers.

**I certify that by the relevant July deadline for submission of CAREER proposals, I will have met all of the following criteria.**

- I will hold a doctoral degree in a field of science or engineering supported by NSF
- I will be untenured
- I will not have received an NSF PECASE or CAREER award
- I will not have competed more than two times in previous NSF CAREER Program Competitions

**I certify that by October 1st following the relevant July deadline for submission of CAREER proposals I will**

- be employed in a tenure-track position  
**OR**  
 be employed in a tenure-track equivalent position
  
- hold the title of assistant professor  
**OR**  
 hold a title that is equivalent to assistant professor
  
- be employed at an institution in the U.S., its territories, or possessions, or the Commonwealth of Puerto Rico that awards degrees in a field of science or engineering supported by NSF  
**OR**  
 be employed at an institution in the U.S., its territories, or possessions, or the Commonwealth of Puerto Rico that is a non-profit, non-degree granting institution such as a museum, observatory, or research lab

## B. PROJECT SUMMARY

We will develop: a) a graph-based representation for the Lagrangian<sup>1</sup> of Lattice Quantum Field Theories (LQFT) and Lattice Quantum Chromodynamics (LQCD) in particular; b) A software engine that, given such representation of a Lagrangian, generates an optimized parallel computer program to perform physics computations in the world described by the Lagrangian; c) a graphical interface to build and visualize LQFT and LQCD Lagrangians.

**Intellectual merit:** LQCD is a “Department of Energy Grand Challenge Project” and it is already partially supported by various funding agencies. Our project will complement existing ones by providing scientists with a tool to automate most of their work, reduce development costs, reduce human of error, and increase the overall productivity of the LQCD community. The PI is an expert in this field. He is the author of more than 20 papers and he is author of FermiQCD ([www.fermiqcd.net](http://www.fermiqcd.net)), a state of the art software tool for parallel LQFT and LQCD computations. FermiQCD is currently used at Fermilab (Department of Energy) and by other research groups around the world. This project will be carried out in consultation with scientists at Fermilab.

**Broader Impact:** Completion of this project will provide a number of major benefits:

- Facilitate the formulation and testing of new and improved LQCD Lagrangians.
- Reduce development and debugging time of LQCD code.
- Facilitate the communication of lattice Lagrangians among scientists.
- Reduce training time for young scientists.
- Allow automated classification and storage of generated data.

Finally this project will strengthen the collaboration between Fermilab and DePaul University and will bring cutting-edge research closer to students of diverse backgrounds.

---

<sup>1</sup>A Lagrangian is a mathematical representation of the dynamical properties of a generic physical system.

## TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

	<b>Total No. of Pages</b>	<b>Page No.* (Optional)*</b>
Cover Sheet for Proposal to the National Science Foundation		
Project Summary (not to exceed 1 page)	<u>1</u>	
Table of Contents	<u>1</u>	
Project Description (Including Results from Prior NSF Support) (not to exceed 15 pages) ( <b>Exceed only if allowed by a     specific program announcement/solicitation or if approved in     advance by the appropriate NSF Assistant Director or designee</b> )	<u>14</u>	
References Cited	<u>1</u>	
Biographical Sketches (Not to exceed 2 pages each)	<u>2</u>	
Budget (Plus up to 3 pages of budget justification)	<u>7</u>	
Current and Pending Support	<u>1</u>	
Facilities, Equipment and Other Resources	<u>4</u>	
Special Information/Supplementary Documentation	<u>7</u>	
Appendix (List below.) <b>(Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)</b>		
Appendix Items:		

\*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated.  
Complete both columns only if the proposal is numbered consecutively.

## D. PROJECT DESCRIPTION

### D.1. Background and Objectives

In 2008, the Linear Hadron Collider[1] (LHC), the most ambitious and powerful experimental facility for high energy particle physics, will start producing data. LHC will allow the creation of new composite particles and precision measurement of their properties[2].

In order to understand the properties of these composite particles it is crucial to improve our computational models of the interactions between the elementary constituents of hadronic matter (quarks and gluons). It is also possible that LHC will discover new particles and new forces that will have to be described in a quantitative way.

Quantum Field Theories (QFT) provide the most general class of mathematical models to describe these particles and their interactions[3]. Quantum Chromodynamics (QCD) is that particular model that describes strong interactions between quarks and gluons[4]. Their “lattice” counterparts consist of a numerical/computational formulation of these models based on discretization of space-time combined with a Markov Chain Monte Carlo approach to the Feynman Path Integral[5].

Lattice Quantum Chromodynamics (LQCD) is the only theoretical framework available to compute, in a systematic way, the properties of composite particles from the properties of their elementary constituents. The systematic comparison between LQCD computations and experimental results may lead to major discoveries, such as the non-unitarity of the Cabibbo-Kobayashi-Maskawa matrix[6] and shed some light into the origin of masses of quarks and leptons, which is one of the fundamental problems of modern particle physics[7, 8].

The US LQCD community is a world leader in this type of computations[9]. The PI is an expert in this field, he is author of more than 20 papers, and he is author of FermiQCD[10, 11, 12].

FermiQCD is the most comprehensive software library for parallel LQFT and

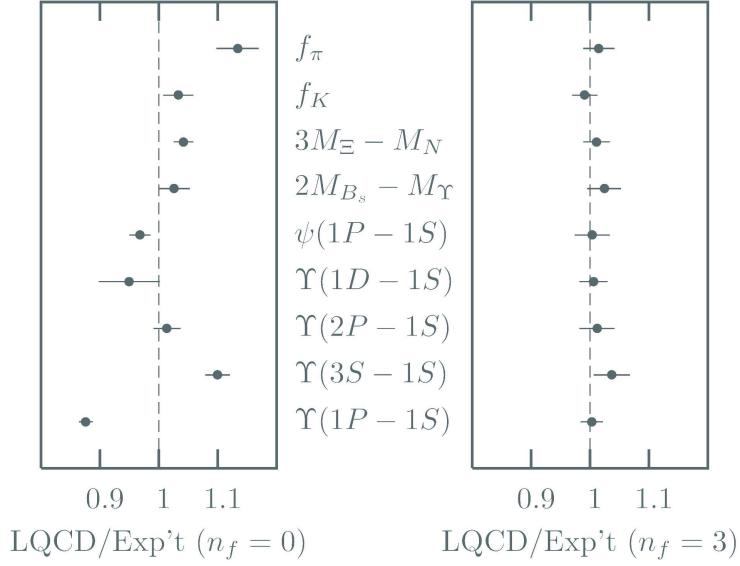


Figure D.1: Ratios of LQCD results over experimental results for various particle properties. The left plot shows old results, the right plot shows new results made possible by simulating dynamical fermions and using an improved Lagrangian for such simulation (asqtad). A significative deviation from 1 would indicate the discovery of a new physical phenomenon.

LQCD computations, is used by many physicists worldwide, and it is considered state of the art in this field.

LQCD is a “Department of Energy Grand Challenge Project”[13] and various projects are currently supported by DOE SciDAC, NSF HEP and NSF NP.

The DOE SciDAC LQCD project[14] for example has focused on the development of the software infrastructure which would allow the easy porting of existing LQCD codes like MILC to new hardware, such as QCDOC and the IBM Blue-Gene/L. The software includes libraries for communications, linear algebra, data parallel computations, and parallel I/O.

SciDAC LQCD also supported the construction of modest prototype clusters at Fermilab and Jefferson Lab based on commodity computers and high performance fabrics, such as gigE toroidal mesh, Myrinet, and Infiniband.

Nevertheless the challenges of Lattice QCD are not purely computational. One of the greatest costs of these computations are training and development costs.

This is the problem we wish to solve with this project.

We believe this project will:

- Facilitate the formulation and testing of new and improved LQCD Lagrangians.
- Facilitate the communication of these Lagrangians among scientists.
- Reduce training time for young scientists.
- Reduce development and debugging time of LQCD code.
- Allow automated classification and storage of generated data.

Our project fits into the existing effort and is complementary to it. We aim to leverage the existing software tools (both FermiQCD and SciDAC) and bring them one step further into the future by completely automating most of the software development and optimization. Upon completion of the project, our system will allow automated development of efficient bug-free parallel code for LQFT and LQCD and will allow experimentation with new and improved discretizations of the LQCD Lagrangian to a higher degree than possible with existing techniques.

## D.2. Technical Proposal

The main idea behind any LQFT in general and LQCD in particular is the following:

- A sufficiently large portion of space-time is discretized and fermionic particles live on lattice sites (represented by red spheres in fig.D.2). Interactions between particles are described by the Lagrangian. The terms of the Lagrangian represent interactions between particles that live on neighbor sites.
- The simplest type of interaction between a fermion (for example a quark) and a gauge field (for example the chromo-electro-magnetic field) on a Lattice was proposed by Wilson (fig.D.3). It can be described as a set of links connecting each lattice site (in blue) with the 8 neighbor sites (in green). The links connecting the sites represent the gauge field.

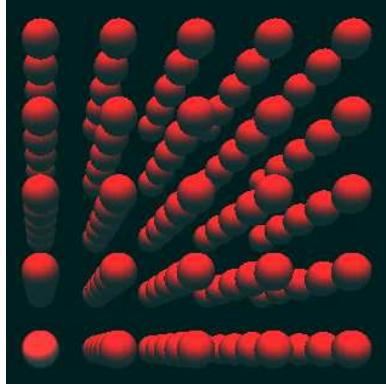


Figure D.2: A small portion of space-time is discretized and fermionic particles live on lattice sites (here represented by red spheres). Interactions between particles are described by the Lagrangian. The terms of the Lagrangian represent interactions between particles that live on neighbor sites.

- Closed paths on the lattice correspond to pure gauge interactions in the Lagrangian. They describe, for example, pure Quantum Electrodynamics (QED) and pure QCD (fig.D.4)
- Open paths correspond to interaction terms between the gauge field and the matter fields. For example the interaction between quarks and gluons in LQCD. Improved actions result in longer and more complex paths (fig.D.5)

We will develop a graph-based representation (serialized in XML[15]), and a graphical interface to describe LQFT and LQCD Langrangians, and a software engine that, given as input a Lagrangian, produces as output optimized parallel code for lattice computations (fig.D.6).

The output of our engine will be plug-ins for the existing FermiQCD library that was developed by the PI and is used by many physicists worldwide. FermiQCD is considered state of the art for parallel LQFT and LQCD computations (fig.D.7).

### D.2.1. Graph-based representation and XML Schema

A Lagrangian is a sum of terms and each term represent a different type of short distance interaction between elementary particles/fields. Because of translation

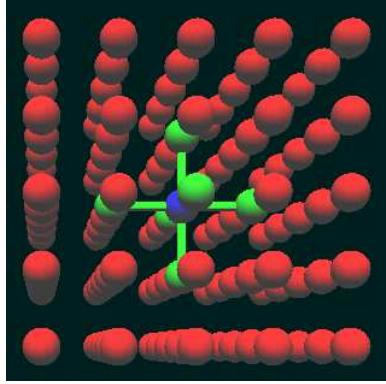


Figure D.3: The simplest type of interaction between a fermion (a quark) and a gauge field (the chromo-electro-magnetic) on a lattice was proposed by Wilson. It can be described as a set of links connecting each lattice site (in blue) with the 8 neighbor sites (in green). Here only a 3D projection is shown therefore each site has only 6 neighbors. The links connecting the sites represent the gauge field.

invariance we can think of each term as associated with a particular lattice site that we identify as the origin. Hence each term can be represented as a connected graph containing the origin.

The links of the graph starting from the origin form paths. Each path is uniquely associated with the transformation a particle would undergo if it were to move along the path. This includes a gauge transformation and a spin transformation. For example the Wilson Lagrangian (fig.D.3) can be represented as a set of 8 one-links paths starting from the origin and going in opposite and orthogonal directions:

$$\begin{aligned} U_{+0}(r + \gamma^0), & \quad U_{-0}(r - \gamma^0), \\ U_{+1}(r + \gamma^1), & \quad U_{-1}(r - \gamma^1), \\ U_{+2}(r + \gamma^2), & \quad U_{-2}(r - \gamma^2), \\ U_{+3}(r + \gamma^3), & \quad U_{-3}(r - \gamma^3); \end{aligned}$$

where  $U_{+i}$  indicates a link connecting the origin to the lattice site obtained from the origin by shifting coordinate  $i$  by  $+1$ .  $(r \pm \gamma^i)$  is a  $4 \times 4$  complex matrix that encodes the spin structure. Each of these terms can be represented using XML in the following form:

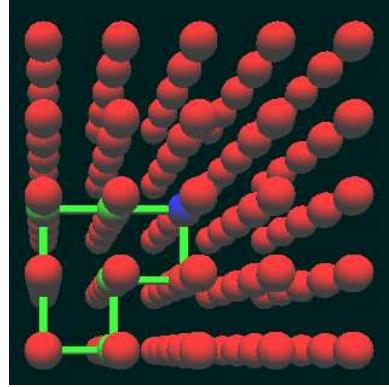


Figure D.4: Closed paths on the lattice correspond to pure gauge interactions in the Lagrangian. They describe, for example, pure quantum electrodynamics and pure quantum chromo dynamics.

```

<path>
  <shift direction=0 verse=+1/>
  <spin>
    <sum>
      <constant name='r' />
      <gamma direction=0/>
    </sum>
  </spin>
</path>
```

The sum of all these connected paths is a disconnected graph which can be also be described in XML by combining similar elementary tags. We will identify the minimal set of tags required to describe any U(1) and SU(n) gauge group and any improved Wilson and Staggered fermions. Optionally we will consider extensions to a number of space-time dimensions different than four.

Our XML schema will provide a unique identifier for each possible Lagrangian. Currently Lagrangians are identified by referencing the name of the paper where it was first proposed. This prevents any automated classification of such Lagrangians and gives rise to notational ambiguities in communication. Our system will overcome this limitation.

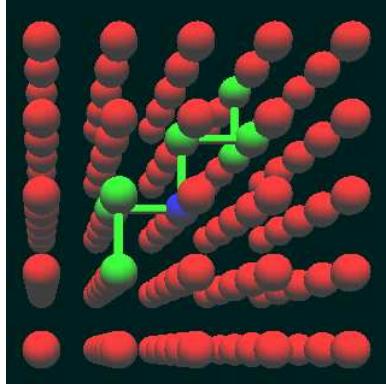


Figure D.5: Open paths correspond to interaction term in the Lagrangian between the gauge field and the matter fields. For example the interaction between quarks and gluons in LQCD. Improved actions result in longer and more complex paths.

### D.2.2. The Graphical Interface

The graphical interface will allow an easy to understand visual representation of the types of interactions between elementary particle described by a Lagrangian.

The graphical interface will look very much as fig.D.3 and will allow 3D projections of the paths in the 4 dimensional space-time. The output of the graphical interface will be the XML representation of the Lagrangian shown in the above example.

Most LQCD Lagrangians are quite complex because they contain a very large number of paths and it is not feasible to design all of them graphically. Nevertheless very few of these paths are independent and most of them are related by symmetries. The graphical interface will be able to interact with the engine (described below) and impose the desired symmetries. For example, if rotational invariance is required, each path will be rotated in all possible ways and new paths will be generated automatically. The system will be able to compare rotated paths to prevent double counting.

### D.2.3. The Engine

The engine is the most important part of our project. It will take as input the graph-based representation of a Lagrangian in our XML schema and will generate FermiQCD plug-ins for that Lagrangian.

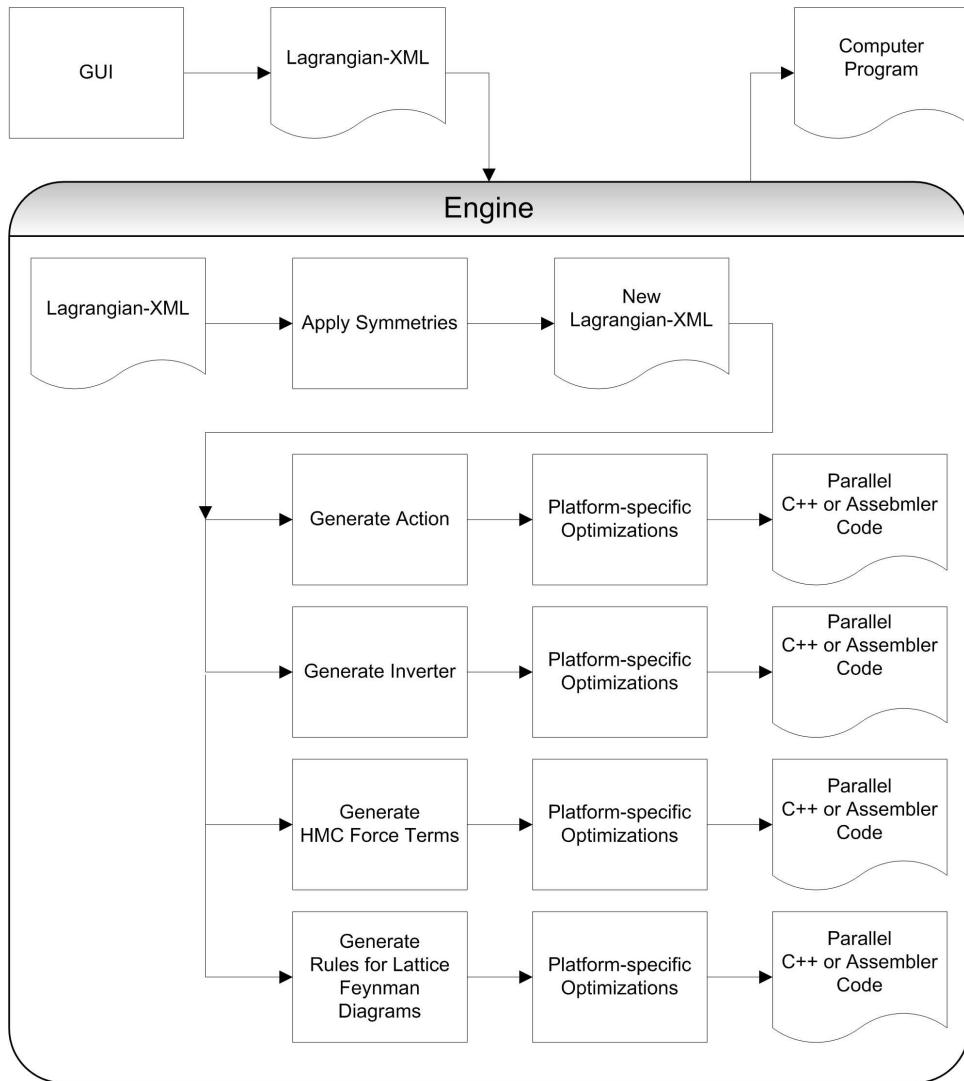


Figure D.6: We will develop a graphical interface and an XML schema to describe improved Lattice Langrangians and a software engine that, given a Lagrangian as input, produces as output an optimized parallel code for Lattice computations.

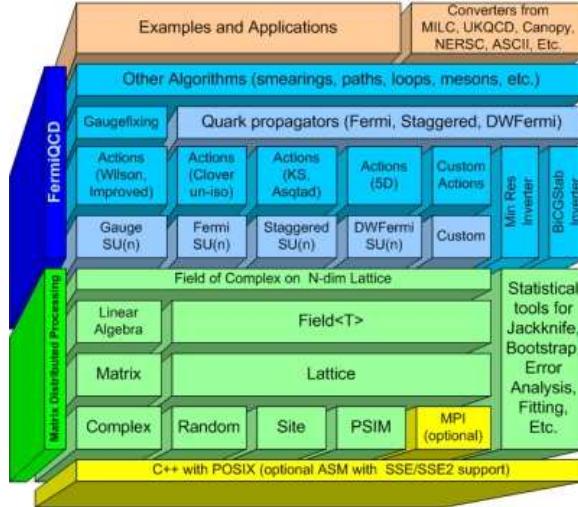


Figure D.7: The output of our engine will be plug-ins for the existing FermiQCD library that was developed by the PI and is used by many physicists worldwide. FermiQCD is considered state of the art in parallel Lattice QCD computations.

These plug-ins are C++ and/or Assembly function that extend the current computational capabilities of FermiQCD. In particular we have identified two classes of plug-ins:

- A function that, given values for the input fields, applies the Lagrangian to each lattice site. In FermiQCD this is the only plug-in required to generate “quark propagators” in the world described by that Lagrangian.
- A function that given a particular lattice link identifies and builds all paths passing through that link. This function is the crucial step of many LQCD algorithms such as the heatbath (Markov-chains based method to generate field configurations), the Hybrid Monte Carlo and the Multi Boson techniques for dynamical fermions.

Optionally, our engine will also generate functions that implement Lattice Feynman Rules for that Lagrangian. Lattice Feynman Diagrams differ from regular ones because they are regularized by the lattice discretization and they are usually computed numerically. They are an essential ingredient of every LQCD computation.

All of the functions above, in the end, can be reduced to the problem of finding the most efficient way to multiply matrices along the possible paths in the graph-based representation of the Lagrangian. This is an optimization problem that can be solved using standard dynamic programming techniques by introducing ad-hoc temporary vectors to store those products that occur more often.

The problem of parallelizing the code is completely factorized out by generating the above functions as plug-ins for FermiQCD because the FermiQCD library abstracts the algorithms from parallelization issues.

#### D.2.4. FermiQCD

FermiQCD is not the only software for these kind of computations. Two common software packages are the MILC code (developed by the MILC collaboration[16]) and SciDAC Chroma[17].

FermiQCD is the only one with a top-down design approach and completely Object Oriented. Its modularity makes it particularly suitable for the proposed project.

FermiQCD differs from MILC in the following aspects: MILC is written in C while FermiQCD is written in C++; MILC is a collection of algorithms for specific LQCD computations while FermiQCD is a collection of algorithms for generic LQFT computations (see fig.D.7); FermiQCD currently does not include any algorithm for dynamical fermions while MILC does. The current project will “complete” FermiQCD in this respect by providing multiple modules for dynamical fermions. This would make FermiQCD unique.

FermiQCD differs from Chroma in the following aspects: Chroma has bottom-up design (the authors have designed highly efficient low level communication and linear algebra functions in C, they then built Chroma as an additional layer of abstraction in C++) while FermiQCD has a top-down design (the PI has designed the interface and multiple implementations of the underlying plug-ins exist); Chroma was designed to work on a variety of machine architectures including dedicated ones while FermiQCD is designed to be most efficient on PC cluster with high network bandwidth and high network latency; the Chroma interface is based on global operations while FermiQCD interface is built around local operations which makes the latter more intuitive. Currently FermiQCD high-level algorithms are faster than Chroma ones but some of the Chroma’s low level routines are better than FermiQCD’s.

The current project will take advantage of these differences and, when appro-

priate, will use some of the low level routines used by Chroma while maintaining the easy-to-use FermiQCD interface.

#### D.2.5. Conclusions

The goals of Lattice Quantum Chromodynamics, as stated in the DOE Grand Challenge are:

- study the spectroscopy of hadrons made up of light-light, heavy-light, and heavy-heavy quarks;
- study the spectroscopy of glueballs and exotic hadrons;
- provide estimates of up, down, strange, charm, and bottom quark masses;
- calculate the strong coupling constant;
- calculate the decay constants for light and heavy-light mesons;
- calculate the form factors for the semi-leptonic decays of B and D meson ;
- calculate the form factor for the radiative decay of B to  $K^*$  gamma;
- calculate the matrix elements of the four fermion operators needed to study CP violation;
- investigate the properties of QCD at finite temperature;
- develop better algorithms and improve the lattice methodology.

Our project addresses the last issue in order to improve our ability to perform all of the above computations.

### D.3. Proposed Timetable and Merit Review Criteria

#### Year 1

We will deliver a detailed description of graph-based representation for any Lagrangian describing scalars, fermions and gauge bosons for any  $SU(n)$  gauge group. This includes naive, Wilson, Kogut-Susskind fermions and all their improved counterpart such as the D1234 action and the Asqtad action.

We will also deliver an engine that works for pure gauge theories (no spin structure).

### **Year 2**

We will deliver modules that can read and write the most general graph-based representation of a Lagrangian in the XML schema.

We will also deliver a graphical interface that works for pure gauge actions.

### **Year 3**

We will extend both the engine and the graphical interface to deal with simple spin structures such as Wilson fermions.

### **Year 4**

We will complete the engine and the graphical interface to sufficient degree to be able to generate automatically Hybrid Monte Carlo algorithms and inverters for some of the most important known improved LQCD Lagrangians.

### **Year 5**

We will perform additional tests and benchmarks that will lead to further improvements of our system. We will deliver detailed documentation.

## **D.4. Requested Support**

We estimate the completion of this project will require 5 years. We request funding for the next 5 years that will support:

- one full time Ph.D. student (tuition plus salary),
- one full time postdoc,
- travel for the PI, the graduate student, and the postdoc,
- one course reduction for the PI.

Both the Ph.D. student and the postdoc would be based at DePaul but they will work in close collaboration with the Fermilab Theory Group.

## **D.5. Educational Impact**

In the short term, this project will provide the opportunity for one graduate and one postdoc to participate in solving a practical and important problem that

encompasses many disciplines including physics, math, computer science, software engineering and distributed system.

In the longer term, it will strengthen the collaborative relations between DePaul University and Fermilab and it will increase the computational capabilities of the US LQCD community.

DePaul has a large minority student body and traditionally it has focused more on teaching than research. In the last few years research activity at DePaul has increased a great deal and this has focused the attention and motivated some of our brightest students. We believe a closer collaboration with the nearby Fermilab will be beneficial to these students and, ultimately, to the community we live in.

DePaul University has grown to be one of the largest universities in the Chicago area and, according to 2002 data, DePaul CTI (Computer Science, Telecommunications, and Information Systems) offers the largest computer science program in the country. The undergraduate program enrolls 2,118 students and offers six different degrees. More than 1039 students are enrolled in the graduate program, which offers nine different master's degrees. DePaul CTI also features a Ph.D. program in computer science that currently enrolls about 50 students. DePaul CTI employs more than 80 full-time faculty and more than 150 part-timers.

The university has shown a growing commitment to cutting edge research and it therefore provides an ideal context for implementing this research project. Research labs include Software Engineering, Human-Computer Interaction, Multimedia, Programming Languages, and Artificial Intelligence labs.

Moreover DePaul University has a highly diverse student body, and in a 1999 Princeton Review was ranked 2nd out of 331 colleges surveyed in student diversity, and 7th in interaction between students of different races and socio-economic classes. DePaul was also recognized in a 2001 issue of "Black Issues in Higher Education" as one of the top 100 universities in America for awarding bachelor's degrees to minorities. DePaul has accomplished this through its ongoing commitment to providing a quality education to all students, by offering special programs such as the NSF funded scholarships for low-income students entering the IT field, and by creating a learning environment in which interaction between students and faculty members is highly valued. The investigators will make a significant effort to provide research opportunities for a broad range of student participants, and will provide active mentoring in order to facilitate their success.

## **D.6. Results from previous NSF Support**

The PI did not receive previous NSF support.

## BIBLIOGRAPHY

- [1] <http://lhc.web.cern.ch/lhc/>
- [2] F. Gianotti *et al.*, Eur. Phys. J. C 39, 293-333 (2005)
- [3] M. Peskin, An Introduction to Quantum Field Theory, HarperCollins Publishers (June 1, 1995)
- [4] T. Muta, Foundations of Quantum Chromodynamics, World Scientific Pub Co Inc (June 1, 1984)
- [5] M. Creutz, Quarks, Gluons and Lattices, Cambridge University Press
- [6] <http://pdg.lbl.gov/>
- [7] W. J. Marciano, Nucl.Phys.Proc.Suppl.59:339-346,1997
- [8] F. Wilczek, hep-th/0201222
- [9] M. di Pierro *et al.*, Nucl. Phys. Proc. Suppl. **129**, 340 (2004) [hep-lat/0310042].
- [10] <http://www.fermiqcd.net>
- [11] M. Di Pierro, Nucl. Phys. Proc. Suppl. **106**, 1034 (2002) [hep-lat/0110116].
- [12] M. Di Pierro, hep-lat/0011083.
- [13] <http://old-www.nersc.gov/research/annrep98/kilcup.html>
- [14] <http://lqcd.fnal.gov/scidac.html>
- [15] <http://www.w3.org/XML/>
- [16] S. A. Gottlieb, Nucl. Phys. Proc. Suppl. **106**, 1031 (2002) [hep-lat/0112038].
- [17] R. G. Edwards and B. Joo [SciDAC Collaboration], hep-lat/0409003.

# F. PI: MASSIMO DI PIERRO

Address: School of Computer Science, Telecommunications and Information Systems, DePaul University, 243 S Wabash Av., Chicago, IL 60604

Phone: 1-312-375-6536; Fax: 1-312-362-6116; Email: mdipierro@cs.depaul.edu

Web Page: <http://www.metacryption.com/mdp>

## Professional Preparation

- Ph.D. in Physics, June 1999. University of Southampton, Southampton, UK
- BS-MS in Physics, June 1996. University of Pisa, Pisa, Italy

## Appointments

- Assistant Professor, DePaul University, School of Computer Science, Telecommunications and Information Systems, 2002-present
- Dostdoc, Fermilab, Theory Division (working on Lattice QCD), 1999-2002

## Synergistic Activities

- Undergrad classes thaught: *Object Oriented Programming in C++, Design and Analysis of Algorithms, Host Security.*
- Grad classes thaught: *Foundations of Computer Science II, Monte Carlo Simulations, Parallel Algorithms, Network Programming.*
- Author of **FermiQCD**, A collection of parallel programs and algorithms for Lattice Computations. [<http://www.fermiqcd.net>]
- Author of **Matrix Distributed Processing**. A C++ toolkit for fast development of parallel applications.[[http://www.phoenixcollective.org/mdp/index\\_mdp.html](http://www.phoenixcollective.org/mdp/index_mdp.html)]
- Author of the **Algorithm Animator**. A didactic software for visualizing algorithms. [[http://www.phoenixcollective.org/mdp/index\\_csc321.html](http://www.phoenixcollective.org/mdp/index_csc321.html)]

## Selected Publications

- C. Aubin *et al.* [Fermilab Lattice Collaboration], Phys. Rev. Lett. **94**, 011601 (2005) [hep-ph/0408306].
- J. N. Simone *et al.* [The Fermilab Lattice, MILC and HPQCD Collaborations], Nucl. Phys. Proc. Suppl. **140**, 443 (2005) [hep-lat/0410030].

- M. Okamoto *et al.*, Nucl. Phys. Proc. Suppl. **140**, 461 (2005) [hep-lat/0409116].
- M. Di Pierro *et al.* [FermiQCD Colaboration Collaboration], Nucl. Phys. Proc. Suppl. **129**, 832 (2004) [hep-lat/0311027].
- M. di Pierro *et al.*, Nucl. Phys. Proc. Suppl. **129**, 328 (2004) [hep-lat/0310045].
- M. di Pierro *et al.*, Nucl. Phys. Proc. Suppl. **129**, 340 (2004) [hep-lat/0310042].
- M. Okamoto *et al.*, Nucl. Phys. Proc. Suppl. **129**, 334 (2004) [hep-lat/0309107].
- C. T. H. Davies *et al.* [HPQCD Collaboration], Phys. Rev. Lett. **92**, 022001 (2004) [hep-lat/0304004].
- M. Di Pierro *et al.*, Nucl. Phys. Proc. Suppl. **119**, 586 (2003) [hep-lat/0210051].
- M. Di Pierro and P. B. Mackenzie, Nucl. Phys. Proc. Suppl. **106**, 777 (2002) [hep-lat/0110120].
- M. Di Pierro, Nucl. Phys. Proc. Suppl. **106**, 1034 (2002) [hep-lat/0110116].
- M. Di Pierro and E. Eichten, Phys. Rev. D **64**, 114004 (2001) [hep-ph/0104208].
- M. Di Pierro, Comput. Phys. Commun. **141**, 98 (2001) [hep-lat/0004007].
- M. Di Pierro *et al.*, Phys. Lett. B **468**, 143 (1999) [hep-lat/9906031].
- G. M. de Divitiis *et al.* [UKQCD Collaboration], JHEP **9810**, 010 (1998) [hep-lat/9807032].
- M. Di Pierro and C. T. Sachrajda, Nucl. Phys. B **534**, 373 (1998) [hep-lat/9805028].

### **Collaborators and & Other Affiliations**

Chris Sachrajda, University of Southampton, Southampton, UK (Ph.D. advisor); Adriano Di Giacomo, University of Pisa, Pisa, Italy (Undergraduate advisor); Kenichi Konishi, University of Pisa, Pisa, Italy (Undergraduate advisor)

Other collaborators in alphabetic order:

C.T.H. Davies, E. Follana, A. Gray, G.P. Lepage, Q. Mason, M. Nobes, J. Shigemitsu, H.D. Trottier, M. Wingate, C. Aubin, C. Bernard, T. Burch, C. DeTar, Steven A. Gottlieb, E.B. Gregory, E. Eichten, U.M. Heller, J.E. Hetrick, J. Osborn, R. Sugar, D. Toussaint, A. El-Khadra, Andreas S. Kronfeld, P.B. Mackenzie, D. Menscher, M. Okamoto, J. Simone.

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 1**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	\$ <b>9,986</b>	\$
2.						
3.						
4.						
5.						
6. ( <b>0</b> ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	<b>9,986</b>	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( <b>1</b> ) POST DOCTORAL ASSOCIATES		<b>12.00</b>	<b>0.00</b>	<b>0.00</b>	<b>50,000</b>	
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
3. ( <b>1</b> ) GRADUATE STUDENTS					<b>18,000</b>	
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS					<b>0</b>	
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					<b>0</b>	
6. ( <b>0</b> ) OTHER					<b>0</b>	
TOTAL SALARIES AND WAGES (A + B)					<b>77,986</b>	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					<b>21,295</b>	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					<b>99,281</b>	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					<b>0</b>	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					<b>4,000</b>	
2. FOREIGN					<b>4,000</b>	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS      \$ <b>0</b>						
2. TRAVEL <b>0</b>						
3. SUBSISTENCE <b>0</b>						
4. OTHER <b>0</b>						
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )					<b>0</b>	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					<b>0</b>	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					<b>0</b>	
3. CONSULTANT SERVICES					<b>0</b>	
4. COMPUTER SERVICES					<b>0</b>	
5. SUBAWARDS					<b>0</b>	
6. OTHER					<b>13,225</b>	
TOTAL OTHER DIRECT COSTS					<b>13,225</b>	
H. TOTAL DIRECT COSTS (A THROUGH G)					<b>120,506</b>	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Modified Total Direct Costs (Rate: 43.0000, Base: 107281)</b>						
TOTAL INDIRECT COSTS (F&A)					<b>46,131</b>	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					<b>166,637</b>	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					<b>0</b>	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ <b>166,637</b>	\$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

1 \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 2**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	\$ <b>10,286</b>	\$
2.						
3.						
4.						
5.						
6. ( <b>0</b> ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	<b>10,286</b>	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( <b>1</b> ) POST DOCTORAL ASSOCIATES		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>51,500</b>	
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
3. ( <b>1</b> ) GRADUATE STUDENTS					<b>18,540</b>	
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS					<b>0</b>	
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					<b>0</b>	
6. ( <b>0</b> ) OTHER					<b>0</b>	
TOTAL SALARIES AND WAGES (A + B)					<b>80,326</b>	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					<b>21,934</b>	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					<b>102,260</b>	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					<b>0</b>	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					<b>4,000</b>	
2. FOREIGN					<b>4,000</b>	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ <b>0</b>						
2. TRAVEL <b>0</b>						
3. SUBSISTENCE <b>0</b>						
4. OTHER <b>0</b>						
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )					<b>0</b>	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					<b>0</b>	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					<b>0</b>	
3. CONSULTANT SERVICES					<b>0</b>	
4. COMPUTER SERVICES					<b>0</b>	
5. SUBAWARDS					<b>0</b>	
6. OTHER					<b>13,622</b>	
TOTAL OTHER DIRECT COSTS					<b>13,622</b>	
H. TOTAL DIRECT COSTS (A THROUGH G)					<b>123,882</b>	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Modified Total Direct Costs (Rate: 43.0000, Base: 110260)</b>						
TOTAL INDIRECT COSTS (F&A)					<b>47,412</b>	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					<b>171,294</b>	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					<b>0</b>	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ <b>171,294</b>	\$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

2 \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 3**

		FOR NSF USE ONLY			
ORGANIZATION		PROPOSAL NO.		DURATION (months)	
<b>DePaul University</b>				Proposed	Granted
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR		AWARD NO.			
<b>Massimo DiPierro</b>					
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer
		CAL	ACAD	SUMR	Funds granted by NSF (if different)
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	\$ <b>10,594</b> \$
2.					
3.					
4.					
5.					
6. ( <b>0</b> ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	<b>10,594</b>
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. ( <b>1</b> ) POST DOCTORAL ASSOCIATES		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>53,045</b>
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>
3. ( <b>1</b> ) GRADUATE STUDENTS					<b>19,096</b>
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS					<b>0</b>
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					<b>0</b>
6. ( <b>0</b> ) OTHER					<b>0</b>
TOTAL SALARIES AND WAGES (A + B)					<b>82,735</b>
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					<b>22,592</b>
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					<b>105,327</b>
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)					
TOTAL EQUIPMENT					<b>0</b>
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					<b>4,000</b>
2. FOREIGN					<b>4,000</b>
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS \$ <b>0</b>					
2. TRAVEL <b>0</b>					
3. SUBSISTENCE <b>0</b>					
4. OTHER <b>0</b>					
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )					<b>0</b>
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES					<b>0</b>
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					<b>0</b>
3. CONSULTANT SERVICES					<b>0</b>
4. COMPUTER SERVICES					<b>0</b>
5. SUBAWARDS					<b>0</b>
6. OTHER					<b>14,031</b>
TOTAL OTHER DIRECT COSTS					<b>14,031</b>
H. TOTAL DIRECT COSTS (A THROUGH G)					<b>127,358</b>
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Modified Total Direct Costs (Rate: 43.0000, Base: 113327)</b>					
TOTAL INDIRECT COSTS (F&A)					<b>48,731</b>
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					<b>176,089</b>
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					<b>0</b>
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ <b>176,089</b> \$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$			
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY			
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION			
		Date Checked	Date Of Rate Sheet		Initials - ORG

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 4**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	\$ <b>10,912</b>	\$
2.						
3.						
4.						
5.						
6. ( <b>0</b> ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	<b>10,912</b>	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( <b>1</b> ) POST DOCTORAL ASSOCIATES		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>54,636</b>	
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
3. ( <b>1</b> ) GRADUATE STUDENTS					<b>19,669</b>	
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS					<b>0</b>	
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					<b>0</b>	
6. ( <b>0</b> ) OTHER					<b>0</b>	
TOTAL SALARIES AND WAGES (A + B)					<b>85,217</b>	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					<b>23,270</b>	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					<b>108,487</b>	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					<b>0</b>	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					<b>4,000</b>	
2. FOREIGN					<b>4,000</b>	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ <b>0</b>						
2. TRAVEL <b>0</b>						
3. SUBSISTENCE <b>0</b>						
4. OTHER <b>0</b>						
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )					<b>0</b>	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					<b>0</b>	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					<b>0</b>	
3. CONSULTANT SERVICES					<b>0</b>	
4. COMPUTER SERVICES					<b>0</b>	
5. SUBAWARDS					<b>0</b>	
6. OTHER					<b>14,452</b>	
TOTAL OTHER DIRECT COSTS					<b>14,452</b>	
H. TOTAL DIRECT COSTS (A THROUGH G)					<b>130,939</b>	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Modified Total Direct Costs (Rate: 43.0000, Base: 116487)</b>						
TOTAL INDIRECT COSTS (F&A)					<b>50,089</b>	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					<b>181,028</b>	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					<b>0</b>	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ <b>181,028</b>	\$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 5**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	\$ <b>11,239</b>	\$
2.						
3.						
4.						
5.						
6. ( <b>0</b> ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>1.00</b>	<b>0.00</b>	<b>11,239</b>	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( <b>0</b> ) POST DOCTORAL ASSOCIATES		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b>	<b>0</b>	
3. ( <b>1</b> ) GRADUATE STUDENTS					<b>20,259</b>	
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS					<b>0</b>	
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					<b>0</b>	
6. ( <b>0</b> ) OTHER					<b>0</b>	
TOTAL SALARIES AND WAGES (A + B)					<b>31,498</b>	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					<b>4,272</b>	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					<b>35,770</b>	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					<b>0</b>	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					<b>4,000</b>	
2. FOREIGN					<b>4,000</b>	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS      \$ <b>0</b>						
2. TRAVEL <b>0</b>						
3. SUBSISTENCE <b>0</b>						
4. OTHER <b>0</b>						
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )					<b>0</b>	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					<b>0</b>	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					<b>0</b>	
3. CONSULTANT SERVICES					<b>0</b>	
4. COMPUTER SERVICES					<b>0</b>	
5. SUBAWARDS					<b>0</b>	
6. OTHER					<b>14,885</b>	
TOTAL OTHER DIRECT COSTS					<b>14,885</b>	
H. TOTAL DIRECT COSTS (A THROUGH G)					<b>58,655</b>	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Modified Total Direct Costs (Rate: 43.0000, Base: 43770)</b>						
TOTAL INDIRECT COSTS (F&A)					<b>18,821</b>	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					<b>77,476</b>	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					<b>0</b>	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ <b>77,476</b>	\$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

**SUMMARY  
PROPOSAL BUDGET**

**Cumulative**

		FOR NSF USE ONLY		
		PROPOSAL NO.		DURATION (months)
		Proposed	Granted	
ORGANIZATION <b>DePaul University</b>				
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.		
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months		Funds Requested By proposer
		CAL	ACAD	Funds granted by NSF (if different)
1. <b>Massimo DiPierro - PI</b>		<b>0.00</b>	<b>5.00</b>	<b>0.00</b> \$ <b>53,017</b> \$
2.				
3.				
4.				
5.				
6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b> <b>0</b>
7. ( <b>1</b> ) TOTAL SENIOR PERSONNEL (1 - 6)		<b>0.00</b>	<b>5.00</b>	<b>0.00</b> <b>53,017</b>
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)				
1. ( <b>4</b> ) POST DOCTORAL ASSOCIATES		<b>12.00</b>	<b>0.00</b>	<b>0.00</b> <b>209,181</b>
2. ( <b>0</b> ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		<b>0.00</b>	<b>0.00</b>	<b>0.00</b> <b>0</b>
3. ( <b>5</b> ) GRADUATE STUDENTS				<b>95,564</b>
4. ( <b>0</b> ) UNDERGRADUATE STUDENTS				<b>0</b>
5. ( <b>0</b> ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				<b>0</b>
6. ( <b>0</b> ) OTHER				<b>0</b>
TOTAL SALARIES AND WAGES (A + B)				<b>357,762</b>
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				<b>93,363</b>
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				<b>451,125</b>
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)				
TOTAL EQUIPMENT				<b>0</b>
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				<b>20,000</b>
2. FOREIGN				<b>20,000</b>
F. PARTICIPANT SUPPORT COSTS				
1. STIPENDS \$ <b>0</b>				
2. TRAVEL <b>0</b>				
3. SUBSISTENCE <b>0</b>				
4. OTHER <b>0</b>				
TOTAL NUMBER OF PARTICIPANTS ( <b>0</b> )			TOTAL PARTICIPANT COSTS	<b>0</b>
G. OTHER DIRECT COSTS				
1. MATERIALS AND SUPPLIES				<b>0</b>
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				<b>0</b>
3. CONSULTANT SERVICES				<b>0</b>
4. COMPUTER SERVICES				<b>0</b>
5. SUBAWARDS				<b>0</b>
6. OTHER				<b>70,215</b>
TOTAL OTHER DIRECT COSTS				<b>70,215</b>
H. TOTAL DIRECT COSTS (A THROUGH G)				<b>561,340</b>
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)				
TOTAL INDIRECT COSTS (F&A)				<b>211,184</b>
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				<b>772,524</b>
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				<b>0</b>
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)			\$	<b>772,524</b> \$
M. COST SHARING PROPOSED LEVEL \$ <b>0</b>		AGREED LEVEL IF DIFFERENT \$		
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY		
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION		
		Date Checked	Date Of Rate Sheet	Initials - ORG

C \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

## **Budget Justification**

### **A. Senior Personnel**

During the funding period, Dr. Massimo Di Pierro will commit 11% academic year effort to this project, the equivalent of one classroom course at one-ninth (11%) load per course. Thus, the salary funds requested from NSF are 11% of Dr. Di Pierro's anticipated base annual salary of \$89,875 for the 2006-2007 academic year. Subsequent years include a 3% increase.

### **B. Other Personnel**

#### **1. Post Doctoral Associates**

One post doc will work on this grant during the first four years. The annual salary for the post doc will be \$50,000 in year one with a 3% increase in subsequent years.

### **3. Graduate Students**

One PhD student will work on this grant all years of the project. The PhD student will receive a salary of \$18,000 per year and full tuition reimbursement (see G6 below). Subsequent years include a 3% increase.

### **C. Benefits**

DePaul University's current fringe benefit rate is for full-time faculty and staff is 35%. For part-time employees, faculty and students during summer months the rate is 10%.

### **E. Travel**

The PI requests \$8,000 in travel funds each year of the grant to attend conferences in order to exchange ideas with colleagues and present the proposed project. Approximately, half of the funds will be used for domestic travel and half for international travel.

### **G.6 Student Tuition**

Tuition is requested for the PhD student working on this grant. Tuition is calculated at \$13,225 in Year 1. Subsequent years are calculated with a 3% increase.

### **I. Indirect Costs**

DePaul University's federally negotiated indirect cost rate is 43% of modified total direct costs.

## H. CURRENT AND PENDING SUPPORT

No current or pending support.

# I. FACILITIES, EQUIPMENT AND OTHER RESOURCES

## I.1. DePaul CTI

### I.1.1. Laboratories

The School of Computer Science, Telecommunications and Information Systems operates specialized laboratories for research and instruction in artificial intelligence, computer-supported collaborative work, distributed systems, high performance computing, human-computer interaction, information systems and electronic commerce, networked multimedia, software engineering and languages and telecommunications. The school offers a total of approximately 800 student laboratory workstations - most of which are Pentium III or Pentium IV based. These workstations run Windows 2000, Windows XP, Red Hat Linux and Sun Solaris operating systems. The School also operates seven high-performance multi-processor Xeon-based servers with large RAID drives and an IBM ES 9000/9221.

Two laboratories may particularly be relevant for this project:

- The software research lab, which has been in existence for a few years, allows users to test experimental programs without jeopardizing the rest of the system. Currently, the lab has six PCs for installation and development of software.
- The Multimedia Networking Research Laboratory (MNLAB). This laboratory houses 20 Multimedia Workstations, including 14 Sun Ultra 10 workstations running Solaris 7 and 4 PCs running Window NT. The MNLAB workstations are connected with 100 Mbps switch which connects to the MBONE. The lab is supported in part by CTI, Sun Micro Systems and IONA Technologies

The IS division also provides printers and overhead projects for labs and residence halls. In addition to 750 lab workstations, information services maintains

over 1,200 residence halls workstations and hundreds of faculty and staff computing platforms

### **I.1.2. Computers and Networks**

DePaul University maintains an extensive technological infrastructure available for students, faculty and staff. In addition, many departments maintain their own resources dedicated for use by its own constituents. The University's existing computer and information infrastructure is in an exceptionally strong position to support the proposed project.

DePaul University is connected by an OC3c ATM-based circuit to the Chicago SBC/AADS network access point (NAP) located in downtown Chicago. The Chicago NAP is one of the largest Internet exchanges in the world. DePaul University maintains approximately fifty peering sessions at the Chicago NAP with institutions of varying sizes and locations including the University's Internet2/Abilene connector MREN. DePaul was one of the first University's in the nation to deploy Juniper router technology at its Internet border, taking advantage of Juniper's award winning design and performance. DePaul University uses BGP, MBGP, PIM-SM and MSDP protocols with its external peering partners to support full line-rate IPv4 unicast and multicast routing.

DePaul University also maintains a physical and logically diverse backup Internet connection to UUNET approximately thirty-five miles north at its Barat Campus in Lake Forest, IL.

The DePaul University metropolitan area backbone network connects campuses at Chicago downtown, Chicago Lincoln Park, Barat campus, Lake Forest, Naperville and Rolling Meadows using Gigabit Ethernet service with an inter-campus speed of 1 gigabit/second.

Almost all of the wired local area networks (LANs) on each DePaul University campus run at 10 and 100 Mb/s switched Ethernet with gigabit uplinks to core backbone devices linking buildings and campuses together. There are also approximately thirty-five 802.11b wireless LAN access points deployed throughout the University network with more to come as demand increases.

Fully transparent IPv4 unicast and multicast connectivity is available by default for all end hosts using DePaul University's public 140.192.0.0/16 IP address block. There are over 250 subnets within the entire DePaul University network. Critical networks and services are protected with network firewalls, host firewalls or various intrusion detection systems.

All primary uplinks and interconnection points on the DePaul network are managed through SNMP. Monitoring of link utilization, packet drop rates, error rates, device environments, protocol usage and summarized flow statistics are collected. Links and devices are also monitored for availability with alerts sent to pagers or to monitored email accounts. DePaul University's internal Networks and Telecom Group (NTG) in Information Services is responsible for installing, managing and monitoring all internet devices and connections. NTG consists of over twenty full-time staff members performing support duties on a 24x7 on-call basis for routers, LAN switches, cabling, telephony, network security, groupware servers and other various network services.

Each staff member is provided with one desktop computer and one laptop computer.

### **I.1.3. Other resources**

DePaul University provides library services at all campuses, with over 600,000 volumes and 8,000 serial holdings. Delivery of information and materials is linked through computer databases that are all available via Web-based interfaces. The Instructional Technology Development (ITD) Group is staffed with over a dozen members whose role is to help promote, train and develop technological solutions in support of education and research at the University. Library administrators and the ITD group are instrumental in helping bring technologies to faculty throughout the University.

The majority of classrooms are equipped with the latest recording and distance learning technologies including video cameras, document cameras, video cassette recorders, workstation screen capture software, white-board input and microphones. All online class content is automatically managed through the Black Board application including prerecorded video, audio and other multimedia materials used in class. DePaul University is undertaking numerous strides in developing next generation online course content that can be made accessible to students all over the country and throughout the world.

The School of Computer Science, Information Systems and Telecommunications is part of a large liberal-arts university including a School of Education, giving us ready access to the expertise of curriculum experts. Additionally, being centrally located in the Chicago metropolitan area provides ready access to the expertise of faculty from the University of Chicago, Northwestern, the University of Illinois at Chicago, Loyola University, and the Illinois Institute of Technology.

Further, DePaul and CTI enjoy a close relationship with corporations in Chicago, and either employ many IT professionals as adjunct faculty or have them as Alumni.

## **I.2. Fermilab**

Fermilab is a DOE national laboratory and it is operated by the Universities Research Association, Inc., a consortium of 89 research universities in the U.S. and abroad. Fermilab is the largest high-energy physics laboratory in the United States, and is second in the world only to CERN, the European Laboratory for Particle Physics.

Fermilab's Tevatron is the world's highest-energy particle accelerator and collider. In the Tevatron, counter-rotating beams of protons and antiprotons produce collisions allowing scientists to examine the most basic building blocks of matter, and the forces acting on them. Particle physics research has grown into an international effort, with experiment collaborations numbering in the hundreds.

### **I.2.1. Computers and Networks**

The Fermilab Computing Division furnishes and operates a laboratory wide network and several computing facilities, including central facilities in the Feynman Computer Center. These facilities include different types of parallel machine and, in particular, large computer clusters dedicated to simulation, reconstruction and analysis of scientific data. They also include data storage capacities of more than one Petabyte (in the form of robotic tape storage) and about hundreds of Terabytes of disk storage.

The Fermilab Computing Division leads more than 250 computer professionals, engineers, technicians and physicists in the Computing Division, whose work include R&D projects to prepare computing for future of High Energy Physics programs.

Letters of support from:

- David Miller (DePaul University, Acting Dean)
- Paul Mackenzie (Fermilab, Scientist II, Theoretical Physics)
- Andreas Kronfeld (Fermilab, Scientist II, Theoretical Physics)
- Don Holmgren (Fermilab Computing Division )

---

# DEPAUL UNIVERSITY



School of Computer Science,  
Telecommunications and  
Information Systems  
243 South Wabash Avenue  
Chicago, Illinois 60604-2301  
312/362-8381  
FAX: 312/362-6116

July 12, 2005

**RE: Massimo DiPierro's NSF CAREER Proposal**

To Whom It May Concern:

Massimo DiPierro was hired as a non tenure-track Assistant Professor in the School of Computer Science, Telecommunications and Information Systems (CTI) at DePaul University in 2002. In the spring of 2005, Dr. DiPierro applied for and received a tenure-track position in CTI. His primary teaching responsibilities have been in our programming and information security curricula.

The School of CTI and the University will support Dr. DiPierro by providing additional funds as needed for travel to professional meetings and conferences as well as office space, supplies, printing, mailing, administrative support, and adequate computing equipment and support.

To confirm his eligibility, Dr. DiPierro holds a Ph.D. in computer science from the University of Southampton. He is currently employed in a tenure-track Assistant Professor position in the School of CTI at DePaul University. He has not received a CAREER or PECASE award, and has not competed more than twice in a CAREER program competition.

I have read and endorse this career-development plan. I attest that the PI's career development plan is supported by and integrated into the educational and research goals of the School of CTI and the organization. I personally commit to the support and professional development of the PI.

Sincerely,

A handwritten signature in black ink that reads "David Miller".

David Miller  
Senior Associate Dean



Fermilab

Fermi National Accelerator Laboratory  
P.O. Box 500 • Batavia, Illinois • 60510Paul B. Mackenzie  
Theoretical Physics DepartmentE-MAIL: mackenzie@fnal.gov  
PHONE: (630) 840-3347  
FAX: (630) 840-5435

July 15, 2005

Dear Colleague,

This letter is in strong support of the application of Massimo di Pierro for an NSF Career Grant. He plays an outstanding role in applying computational methods to development of software that plays an important role in real particle physics applications. He has played a key role in our physics projects at Fermilab, and can play an important role in the national program for lattice gauge theory software.

Lattice gauge theory calculations allow the prediction of the properties of quarks and gluons by means of large-scale numerical simulations of quantum chromodynamics. Because some of these calculations are essential to the current experimental program of high energy physics, the Department of Energy has been investing around \$2,000,000 per year for the last five years in creating a national infrastructure of hardware and software for lattice gauge theory. This investment is scheduled to rise to \$4,500,000 per year starting in 2006. I serve as Fermilab's representative on the Executive Committee of USQCD, which oversees the development of this infrastructure for the DoE. The computing hardware created under this initiative includes large (0.7 teraflop) computer clusters at Fermilab which are networked tightly together with a high-speed communications fabric (Infiniband). DiPierro has access to these computers as part of his work with us. The software includes components written at Fermilab and components written as part of the national effort. DiPierro played a key role in the development of this software. His proposed new project is very imaginative and has the potential to be very influential in the national effort.

15/05 FRI 13:42 FAX 630 840 5435

FERMILAB

003

Di Pierro was a post doc in our group at Fermilab before moving to DePaul, and has played an important role in our physics program. He played a key role in developing the software that is doing our calculations at this minute. The Career Grant would enable him to expand this role for us, and to play an important role in the national program for lattice QCD software infrastructure that is being overseen by Richard Brower of Boston University. I give him my strongest recommendation for this grant.

Sincerely,



Paul B. Mackenzie



Fermi National Accelerator Laboratory  
P.O. Box 500 • Batavia, Illinois • 60510

E-MAIL: [ask@fnal.gov](mailto:ask@fnal.gov)  
PHONE: +1-630-840-3753  
FAX: +1-630-840-5435

July 18, 2005

To whom it may concern:

I am writing to support Prof. **Massimo Di Pierro**, who has submitted a proposal for an NSF Career Grant. He strikes me as an ideal candidate for such a grant. He is a theoretical physicist by training and is now an Assistant Professor of Computer Science at DePaul University. He is an innovative scientist, connecting disciplines of pure and applied science. In this sense is carving out a new niche—a new sort of career—one that bridges elementary particle physics and computer science.

My first encounter with Massimo was in 1998 at the International Symposium on Lattice Field Theory in Boulder, Colorado. He presented a short talk on results from his doctoral research. From the outset it was clear that he was someone special. Most physicists are very smart, but he stood out from the crowd. His presentation was incisive, complete, and lucid: better than most old pros and outstanding for a student.

As a result of this encounter, not to mention sterling letters of reference from his professors in England and Italy, my colleagues and I offered Di Pierro a position as a postdoctoral researcher. We found that the positive first impression was well-founded. He set about his work with great gusto and brilliance. We have had many outstanding younger colleagues here at Fermilab, but few ranged so widely. He not only carried out research in elementary particle physics but also made remarkable contributions to computer software. Now, every physicist writes some computer programs. But Dr. Di Pierro created a general-purpose library for parallel computing, called Matrix Distributed Processing, or `mdp`. These days, the most cost-effective method to build a large computing facility is to connect many PCs (with the same cheap fast processors as home or business computers) into a large network. The downside is to distribute the work across the network. This is what `mdp` does; it is so good and versatile that it finds applications outside the specialized world of science.

The current proposal concerns a suite of software called **FermiQCD**, which is built on **mdp**. **FermiQCD** already plays an important role in the research of the Fermilab Lattice Collaboration, led by Paul Mackenzie and me, of which Massimo is a valued member. This research, on lattice QCD calculations essential to experiments on the flavor sector of the Standard Model, has recently attracted wide attention. There have been, for example, descriptions of our research in *Nature*, *ScienceNews*, and *The New Scientist*. In several cases, we were in a “race” with experiments that were measuring the quantities we were trying to calculate. The versatility of *FermiQCD* was essential to getting the job done in a timely way.

Although our predictions were successful, there are several ways to improve the basic calculations. The target is a precision of 1 per cent. Indeed, such precision is necessary to aid the search for new phenomena in decays of the  $B$  meson. The extensions for *FermiQCD* proposed here will be a key to the efficient use of human and computer resources: it will help us achieve more results faster. As explained in the proposal, the proposed extensions to **FermiQCD** are not part of any other software for lattice QCD, although the framework is compatible with other packages.

The other component of the proposal is support for a postdoctoral researcher and for a student. The student would come from the Computer Science Department and would, thus, focus more on that side of the program. The postdoc would be a physicist, however. One of the NSF’s aims is to increase the number of postdocs working in the US on lattice QCD (or so I am told). This postdoc would be a member of the Fermilab Lattice Collaboration, so Prof. Di Pierro will be able to attract candidates of the highest calibre.

In summary, the proposed research is a key part of an important research program. Awarding the grant to Prof. Di Pierro will foster a special and unique career. I support this proposal without reservation.

Sincerely,

Andreas S. Kronfeld  
Scientist II, Theoretical Physics

Fermilab  
Computing Division  
P.O. Box 500, M/S 120  
Batavia, IL 60510  
630-840-2745

July 15, 2005

Dear Colleague:

I am most pleased to write a letter in support of Massimo DiPierro's NSF Career proposal. I am the Project Manager for the national Lattice QCD Computing Project of the Office of Science of the Department of Energy. This project will build and operate significant supercomputing facilities for lattice QCD simulations at Fermilab, Jefferson Lab, and Brookhaven National Lab. The lattice QCD community is very much interested in the development and exploration of new algorithms and actions (Lagrangians). The software developed under Massimo's proposal would greatly increase physicist productivity in this area. Further, there is promise that the proposed optimized code generator would increase throughput on our facilities and ease code porting between computer architectures.

I have known Massimo since 1999 and worked with him for three years while he held a postdoc at Fermilab. He is an excellent communicator and at Depaul has been an outstanding educator. His FermiQCD software is heavily used by the Fermilab lattice QCD theorists. He has a particularly strong talent for computer visualization techniques.

I would be very happy to facilitate access to DOE lattice QCD hardware for this work, as well as to foster interactions between the software developers on the DOE lattice QCD project and Massimo's group.

I highly recommend Massimo to you.

Sincerely,

Don Holmgren

---

**02 INFORMATION ABOUT PRINCIPAL INVESTIGATORS/PROJECT DIRECTORS(PI/PD) and  
co-PRINCIPAL INVESTIGATORS/co-PROJECT DIRECTORS**

---

Submit only ONE copy of this form for each PI/PD and co-PI/PD identified on the proposal. The form(s) should be attached to the original proposal as specified in GPG Section II.B. Submission of this information is voluntary and is not a precondition of award. This information will not be disclosed to external peer reviewers. **DO NOT INCLUDE THIS FORM WITH ANY OF THE OTHER COPIES OF YOUR PROPOSAL AS THIS MAY COMPROMISE THE CONFIDENTIALITY OF THE INFORMATION.**

---

**PI/PD Name:** Massimo DiPierro

**Gender:**  Male  Female

**Ethnicity:** (Choose one response)  Hispanic or Latino  Not Hispanic or Latino

**Race:**  American Indian or Alaska Native

(Select one or more)  Asian

Black or African American

Native Hawaiian or Other Pacific Islander

White

**Disability Status:**  Hearing Impairment

(Select one or more)  Visual Impairment

Mobility/Orthopedic Impairment

Other

None

**Citizenship:** (Choose one)  U.S. Citizen  Permanent Resident  Other non-U.S. Citizen

**Check here if you do not wish to provide any or all of the above information (excluding PI/PD name):**

**REQUIRED: Check here if you are currently serving (or have previously served) as a PI, co-PI or PD on any federally funded project**

---

**Ethnicity Definition:**

**Hispanic or Latino.** A person of Mexican, Puerto Rican, Cuban, South or Central American, or other Spanish culture or origin, regardless of race.

**Race Definitions:**

**American Indian or Alaska Native.** A person having origins in any of the original peoples of North and South America (including Central America), and who maintains tribal affiliation or community attachment.

**Asian.** A person having origins in any of the original peoples of the Far East, Southeast Asia, or the Indian subcontinent including, for example, Cambodia, China, India, Japan, Korea, Malaysia, Pakistan, the Philippine Islands, Thailand, and Vietnam.

**Black or African American.** A person having origins in any of the black racial groups of Africa.

**Native Hawaiian or Other Pacific Islander.** A person having origins in any of the original peoples of Hawaii, Guam, Samoa, or other Pacific Islands.

**White.** A person having origins in any of the original peoples of Europe, the Middle East, or North Africa.

---

**WHY THIS INFORMATION IS BEING REQUESTED:**

The Federal Government has a continuing commitment to monitor the operation of its review and award processes to identify and address any inequities based on gender, race, ethnicity, or disability of its proposed PIs/PDs. To gather information needed for this important task, the proposer should submit a single copy of this form for each identified PI/PD with each proposal. Submission of the requested information is voluntary and will not affect the organization's eligibility for an award. However, information not submitted will seriously undermine the statistical validity, and therefore the usefulness, of information received from others. Any individual not wishing to submit some or all the information should check the box provided for this purpose. (The exceptions are the PI/PD name and the information about prior Federal support, the last question above.)

Collection of this information is authorized by the NSF Act of 1950, as amended, 42 U.S.C. 1861, et seq. Demographic data allows NSF to gauge whether our programs and other opportunities in science and technology are fairly reaching and benefiting everyone regardless of demographic category; to ensure that those in under-represented groups have the same knowledge of and access to programs and other research and educational opportunities; and to assess involvement of international investigators in work supported by NSF. The information may be disclosed to government contractors, experts, volunteers and researchers to complete assigned work; and to other government agencies in order to coordinate and assess programs. The information may be added to the Reviewer file and used to select potential candidates to serve as peer reviewers or advisory committee members. See Systems of Records, NSF-50, "Principal Investigator/Proposal File and Associated Records", 63 Federal Register 267 (January 5, 1998), and NSF-51, "Reviewer/Proposal File and Associated Records", 63 Federal Register 268 (January 5, 1998).

---

**02 INFORMATION ABOUT PRINCIPAL INVESTIGATORS/PROJECT DIRECTORS(PI/PD) and  
co-PRINCIPAL INVESTIGATORS/co-PROJECT DIRECTORS**

---

Submit only ONE copy of this form for each PI/PD and co-PI/PD identified on the proposal. The form(s) should be attached to the original proposal as specified in GPG Section II.B. Submission of this information is voluntary and is not a precondition of award. This information will not be disclosed to external peer reviewers. **DO NOT INCLUDE THIS FORM WITH ANY OF THE OTHER COPIES OF YOUR PROPOSAL AS THIS MAY COMPROMISE THE CONFIDENTIALITY OF THE INFORMATION.**

---

**PI/PD Name:** Panagiotis Spentzouris

**Gender:**  Male  Female

**Ethnicity:** (Choose one response)  Hispanic or Latino  Not Hispanic or Latino

**Race:** (Select one or more)  American Indian or Alaska Native

Asian

Black or African American

Native Hawaiian or Other Pacific Islander

White

**Disability Status:**  Hearing Impairment

(Select one or more)  Visual Impairment

Mobility/Orthopedic Impairment

Other

None

**Citizenship:** (Choose one)  U.S. Citizen  Permanent Resident  Other non-U.S. Citizen

**Check here if you do not wish to provide any or all of the above information (excluding PI/PD name):**

**REQUIRED: Check here if you are currently serving (or have previously served) as a PI, co-PI or PD on any federally funded project**

---

**Ethnicity Definition:**

**Hispanic or Latino.** A person of Mexican, Puerto Rican, Cuban, South or Central American, or other Spanish culture or origin, regardless of race.

**Race Definitions:**

**American Indian or Alaska Native.** A person having origins in any of the original peoples of North and South America (including Central America), and who maintains tribal affiliation or community attachment.

**Asian.** A person having origins in any of the original peoples of the Far East, Southeast Asia, or the Indian subcontinent including, for example, Cambodia, China, India, Japan, Korea, Malaysia, Pakistan, the Philippine Islands, Thailand, and Vietnam.

**Black or African American.** A person having origins in any of the black racial groups of Africa.

**Native Hawaiian or Other Pacific Islander.** A person having origins in any of the original peoples of Hawaii, Guam, Samoa, or other Pacific Islands.

**White.** A person having origins in any of the original peoples of Europe, the Middle East, or North Africa.

---

**WHY THIS INFORMATION IS BEING REQUESTED:**

The Federal Government has a continuing commitment to monitor the operation of its review and award processes to identify and address any inequities based on gender, race, ethnicity, or disability of its proposed PIs/PDs. To gather information needed for this important task, the proposer should submit a single copy of this form for each identified PI/PD with each proposal. Submission of the requested information is voluntary and will not affect the organization's eligibility for an award. However, information not submitted will seriously undermine the statistical validity, and therefore the usefulness, of information received from others. Any individual not wishing to submit some or all the information should check the box provided for this purpose. (The exceptions are the PI/PD name and the information about prior Federal support, the last question above.)

Collection of this information is authorized by the NSF Act of 1950, as amended, 42 U.S.C. 1861, et seq. Demographic data allows NSF to gauge whether our programs and other opportunities in science and technology are fairly reaching and benefiting everyone regardless of demographic category; to ensure that those in under-represented groups have the same knowledge of and access to programs and other research and educational opportunities; and to assess involvement of international investigators in work supported by NSF. The information may be disclosed to government contractors, experts, volunteers and researchers to complete assigned work; and to other government agencies in order to coordinate and assess programs. The information may be added to the Reviewer file and used to select potential candidates to serve as peer reviewers or advisory committee members. See Systems of Records, NSF-50, "Principal Investigator/Proposal File and Associated Records", 63 Federal Register 267 (January 5, 1998), and NSF-51, "Reviewer/Proposal File and Associated Records", 63 Federal Register 268 (January 5, 1998).

---

**List of Suggested Reviewers or Reviewers Not To Include (optional)**

---

**SUGGESTED REVIEWERS:**

Not Listed

**REVIEWERS NOT TO INCLUDE:**

Not Listed

# COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 04-2  NSF 04-012                    02/24/04					FOR NSF USE ONLY  NSF PROPOSAL NUMBER
FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)  <b>PHY - ITR FOR NATIONAL PRIORITIES (continued)</b>					
DATE RECEIVED	NUMBER OF COPIES	DIVISION ASSIGNED	FUND CODE	DUNS# (Data Universal Numbering System)	FILE LOCATION
				<b>825753379</b>	
EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)  <b>362167048</b>		SHOW PREVIOUS AWARD NO. IF THIS IS <input type="checkbox"/> A RENEWAL <input type="checkbox"/> AN ACCOMPLISHMENT-BASED RENEWAL		IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES <input type="checkbox"/> NO <input checked="" type="checkbox"/> IF YES, LIST ACRONYM(S)	
NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE  <b>DePaul University</b>			ADDRESS OF AWARDEE ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE  <b>1 East Jackson Boulevard Chicago, IL. 606042218</b>		
AWARDEE ORGANIZATION CODE (IF KNOWN)  <b>0016717000</b>					
NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE			ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE		
PERFORMING ORGANIZATION CODE (IF KNOWN)					
IS AWARDEE ORGANIZATION (Check All That Apply) (See GPG II.C For Definitions)		<input type="checkbox"/> SMALL BUSINESS <input type="checkbox"/> MINORITY BUSINESS <input type="checkbox"/> FOR-PROFIT ORGANIZATION <input type="checkbox"/> WOMAN-OWNED BUSINESS		<input type="checkbox"/> IF THIS IS A PRELIMINARY PROPOSAL THEN CHECK HERE	
TITLE OF PROPOSED PROJECT <b>Parallel Algorithms for Particle Beam Simulation</b>					
REQUESTED AMOUNT <b>\$ 341,984</b>	PROPOSED DURATION (1-60 MONTHS) <b>48</b> months		REQUESTED STARTING DATE <b>09/15/04</b>	SHOW RELATED PRELIMINARY PROPOSAL NO. IF APPLICABLE	
CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW					
<input type="checkbox"/> BEGINNING INVESTIGATOR (GPG I.A) <input type="checkbox"/> HUMAN SUBJECTS (GPG II.D.6) <input type="checkbox"/> DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.C)      Exemption Subsection _____ or IRB App. Date _____ <input type="checkbox"/> PROPRIETARY & PRIVILEGED INFORMATION (GPG I.B, II.C.1.d) <input type="checkbox"/> INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES INVOLVED <input type="checkbox"/> HISTORIC PLACES (GPG II.C.2.j)      (GPG II.C.2.j) <input type="checkbox"/> SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.1) <input type="checkbox"/> VERTEBRATE ANIMALS (GPG II.D.5) IACUC App. Date _____ <input type="checkbox"/> HIGH RESOLUTION GRAPHICS/OTHER GRAPHICS WHERE EXACT COLOR REPRESENTATION IS REQUIRED FOR PROPER INTERPRETATION (GPG I.E.1)					
PI/PD DEPARTMENT <b>CTI</b>	PI/PD POSTAL ADDRESS <b>243 S. Wabash Av.</b>				
PI/PD FAX NUMBER <b>312-362-6116</b>	Chicago, IL 606042218 United States				
NAMES (TYPED)	High Degree	Yr of Degree	Telephone Number	Electronic Mail Address	
PI/PD NAME <b>Massimo DiPierro</b>	<b>PhD</b>	<b>1999</b>	<b>312-362-5173</b>	<b>MDiPierro@cs.depaul.edu</b>	
CO-PI/PD <b>Panagiotis Spentzouris</b>	<b>PhD</b>	<b>1994</b>	<b>202-293-1382</b>	<b>spentz@fnal.gov</b>	
CO-PI/PD					
CO-PI/PD					
CO-PI/PD					

## CERTIFICATION PAGE

### **Certification for Authorized Organizational Representative or Individual Applicant:**

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 04-2. Willful provision of false information in this application and its supporting documents or in reports required under an ensuing award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflicts which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

#### **Drug Free Work Place Certification**

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Drug Free Work Place Certification contained in Appendix C of the Grant Proposal Guide.

#### **Debarment and Suspension Certification**

(If answer "yes", please provide explanation.)

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency?

Yes

No

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Debarment and Suspension Certification contained in Appendix D of the Grant Proposal Guide.

#### **Certification Regarding Lobbying**

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding \$100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding \$150,000.

#### **Certification for Contracts, Grants, Loans and Cooperative Agreements**

The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, "Disclosure of Lobbying Activities," in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all tiers including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, Title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than \$10,000 and not more than \$100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE	SIGNATURE	DATE
NAME		
TELEPHONE NUMBER	ELECTRONIC MAIL ADDRESS	FAX NUMBER

\*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

## **COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION**

FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) - continued from page 1  
(Indicate the most specific unit known, i.e. program, division, etc.)

**CNS - ITR FOR NATIONAL PRIORITIES**  
**DGE - ITR FOR NATIONAL PRIORITIES**

## B. PROJECT SUMMARY

This proposal describes a research effort whose primary objective is to develop and optimize parallel solvers for high-fidelity beam dynamics simulations which aim to model the many-body, three-dimensional, and non-linear aspects of present and future accelerators.

**Merit Review Criteria:** The proposed project is of crucial importance to the study of space-charge effects in high-intensity proton sources, such as the Fermilab Booster, and the study of beam-beam effects in hadron colliders, such as the Fermilab Tevatron. In order to accurately understand these effects, thousands of turns around the machines need to be simulated using millions of particles, thus the development of fast and efficient solvers is essential for the successful use of these models. The PI, Massimo Di Pierro, is a Professor of Computer Science at DePaul University and has been a Research Associate in the Fermilab Theory Group. He has developed FermiQCD, an open source library for parallel computations. The Co-PI, Panagiotis Spentzouris, is a leading scientist at Fermilab with extensive experience in the field of particle beam simulation. The PI and Co-PI have access to the resources required to complete this project.

**Broader Impact:** The effort described here will bring together Fermilab accelerator physicists who are developing such beam dynamics codes, and scientists from the School of Computer Science of DePaul University who have great expertise in the development of such solvers. The newly developed solvers will maximize the efficiency of the simulation, thus enabling researchers to better pursue accelerator design and performance studies while, at the same time, will provide an excellent research project for DePaul students. The project will support a graduate student and a new summer internship program for DePaul undergraduates, hosted at Fermilab.

## TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

	<b>Total No. of Pages</b>	<b>Page No.* (Optional)*</b>
Cover Sheet for Proposal to the National Science Foundation		
Project Summary (not to exceed 1 page)	<u>1</u>	
Table of Contents	<u>1</u>	
Project Description (Including Results from Prior NSF Support) (not to exceed 15 pages) <b>(Exceed only if allowed by a     specific program announcement/solicitation or if approved in     advance by the appropriate NSF Assistant Director or designee)</b>	<u>13</u>	
References Cited	<u>1</u>	
Biographical Sketches (Not to exceed 2 pages each)	<u>4</u>	
Budget (Plus up to 3 pages of budget justification)	<u>11</u>	
Current and Pending Support	<u>2</u>	
Facilities, Equipment and Other Resources	<u>4</u>	
Special Information/Supplementary Documentation	<u>10</u>	
Appendix (List below.) <b>(Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)</b>		
Appendix Items:		

\*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated.  
Complete both columns only if the proposal is numbered consecutively.

## D. PROJECT DESCRIPTION

### D.1. Background and Objectives

Particle accelerators constitute one of the most useful and complex research instruments available to scientists today. In High Energy Physics (HEP), experiments associated with accelerators have led to important discoveries about the fundamental forces of nature and to the discovery of new elementary particles. In order to keep exploring the frontier of HEP, it is necessary that accelerators provide higher energy, higher intensity, and higher luminosity beams. This leads to systems with higher complexity and more stringent performance tolerances than in the past. Simulation plays already a prominent role in the design and performance optimization of accelerators, and there is an increasing need for three-dimensional high-fidelity simulations of nonlinear collective beam effects. The performance of both existing accelerators, such as the Fermilab Tevatron and the Fermilab Booster[1] and future accelerators, some of which will come online in the next few years (such as the CERN Large Hadron Collider), will benefit from these computational models. The desirable final objective is that of understanding and controlling the collective beam effects. Some of these effects result in the development of the beam halo that eventually causes beam losses and irradiation from the accelerator complex; other effects result in the dilution of the beam phase space that causes a decrease in accelerator luminosity.

The dynamics of charged particles in accelerators, in absence of radiation, can be divided into two categories: effects due to the interaction of the beam particles with the electromagnetic fields produced by the magnets and the RF cavities of the accelerator (single-particle optics), and effects due to the interaction of beam particles with fields produced by the beam itself or by other beams in each vicinity (collective beam effects). A comprehensive accelerator simulation must include both types of effects. The problem of modeling linear and non-linear single-particle dynamics in accelerators is addressed by the field of magnetic optics, a very mature research field that has reached great levels of accuracy. On

the other hand, modeling the evolution of a distribution of particles, such as a beam, is a less mature and much more computationally intensive endeavor, especially in the case of very intense charged beams, because the fields produced by the particles themselves affect the beam dynamics. The most widely used method to simulate these effects is the particle simulation approach. In this approach, the beam distribution is represented by a set of macro-particles that evolve according to the single particle equations of motion; the additional effects due to the electromagnetic fields are calculated by solving the Poisson-Vlasov equation at every time step. The most commonly used approach for solving this equation is the Particle-In-Cell (PIC) method. In this method, three operations are performed during the simulation at each time step: first, the particles are deposited on a spatial grid; second, the fields are calculated on the grid; third and final, the fields are interpolated back to the particles and used to push the particles. Since a large number of macro-particles (order a few million or more) is needed to obtain the required simulation accuracy, the use of massively parallel computers is necessary.

In the parallel PIC approach there are three main bottlenecks that affect parallel performances: a) inter-processor communication required by the fact that each particle affects grids points that may reside on a processing node other than the one where the particle resides; b) the distribution of particles among the processing nodes may become unbalanced; c) global communication or irregular communication may be required by the solver. A complete simulation, especially of circular accelerators, typically involves many thousands of steps. For this reason, the development of fast and efficient algorithms for the field solvers and the manipulation of particles is essential for an effective simulation of collective beam effects.

## D.2. Technical Proposal

The Co-PI of this proposal is the leader of the Fermilab team which has developed the Synergia accelerator modeling package[2].

Synergia is an accelerator physics simulation framework that implements a fully three-dimensional space-charge model, with both circular and linear machine simulation capabilities. The implementation is fully parallelized. Synergia is a hybrid code and the primary accelerator physics components are taken from existing, although modified, codes.

In particular, Synergia uses IMPACT[3] for its parallel simulation of the prop-

agation of particles, the modeling of RF cavities and, most importantly, parallel calculations of space-charge effects. IMPACT contains a fully three-dimensional space charge model utilizing the split-operator technique. The split-operator technique takes a Hamiltonian of the form  $H = H_{ext} + H_{sc}$ , where  $H_{ext}$  is the Hamiltonian for the magnetic optics part of the problem and  $H_{sc}$  is the Hamiltonian for the space-charge part of the problem. If  $M_{ext}$  is the transfer map corresponding to  $H_{ext}$  and  $M_{sc}$  is the transfer map corresponding to  $H_{sc}$ , then  $M(t) = M_{ext}(t/2)M_{sc}(t)M_{ext}(t/2) + O(t^2)$  is the transfer map for  $H$  to leading order in  $t$ . IMPACT is a FORTRAN90 code.

For the modeling of single-particle optics, Synergia uses the mxyzptlk/beamline libraries[4], which can perform a wide range of accelerator physics computations. This package is written in a modern style and its design principles are based on objects, encapsulation and well-considered application program interfaces. The libraries include a set of useful utility classes such as Vector, Matrix, etc., objects for modeling elements of a beamline including a full parser for the Methodic Accelerator Design (MAD) language, and a module for automatic differentiation and differential algebra. The Synergia code integrates the above packages using a framework written in C++ and Python. Synergia also provides easy-to-use users interface (Fig.D.1), analysis and visualization tools (Fig.D.2).

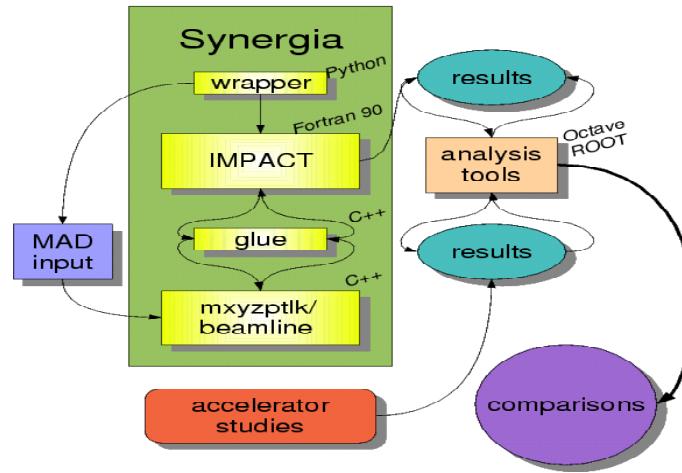


Figure D.1: Overview of the Synergia framework.

Synergia has been used, for example, to model the first few hundred turns

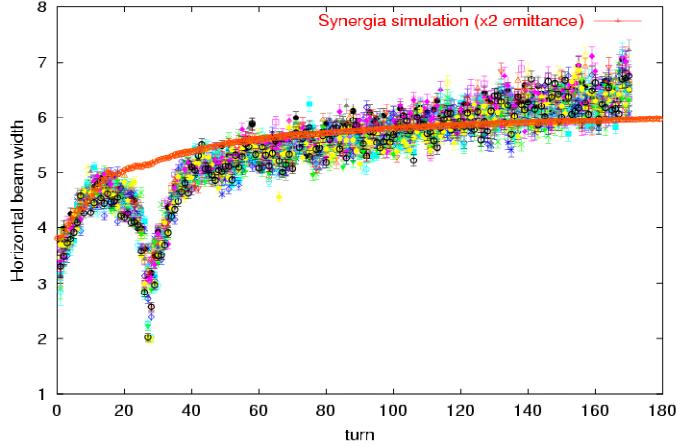


Figure D.2: The spatial beam distribution from a Booster simulation using the Synergia framework.

of beam circulation in the Fermilab Booster, after injection. There are many factors affecting the behavior of the Booster beam, including energy and remittance of the incoming beam, nonlinear field errors and space charge effects. The space-charge effects are believed to be responsible for a significant fraction of the observed losses in the Booster during the first 2ms of each cycle (injection, capture, and bunching phase). Synergia predictions, including space-charge, have been compared to measurements of the beam width as function of time and good qualitative agreement has been found (Fig.D.3).

The above simulations of the Fermilab Booster only cover 1% of the full Booster cycle (the beam performs a total of 20000 turns from injection to extraction). Although space-charge effects are most important at injection, it is desirable to be able to model the complete Booster. Using 128 processing nodes on the NERSC IBM SP super-computer, the IMPACT simulation of the entire Booster would take roughly 24 days.

As shown in Fig.D.4, the performance of the current code plateaus at around 128 processors, so it is clear that the current version of Synergia is not able to take advantage of a significant fraction of the full power of the IBM SP architecture. In addition, plans are underway to extend the Synergia framework to include beam-beam and electron cloud effects which are relevant to the ongoing Fermilab program and also will be very important for commissioning and oper-

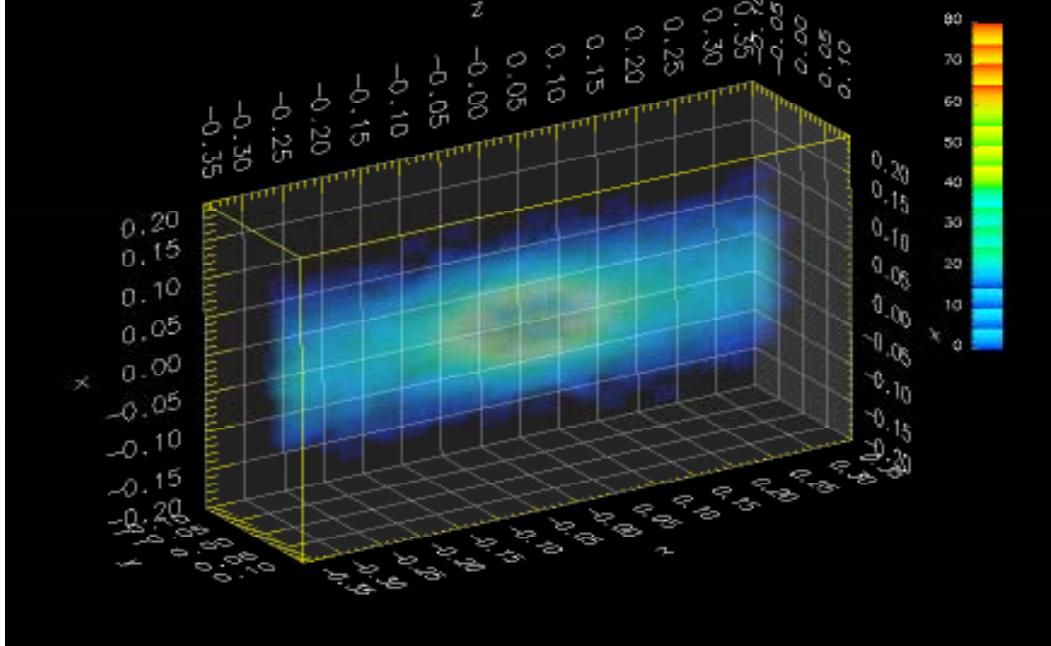


Figure D.3: Booster beam profile as a function of time (turn number) compared to the Synergia prediction. The "notch" close to injection is due to injection magnet effects not included in the simulation.

ating the LHC. These additional physics topics have very similar computational requirements. Thus, it is imperative to develop more scalable and efficient PIC implementations and solvers.

The Principal Investigator is the author of a software library called FermiQCD [5][6]. This library is written in C++ and was originally developed to perform Lattice Quantum Chromo Dynamics simulations. FermiQCD is based on Matrix Distributed Processing [7][8][9] also written and developed by the PI. The main task of the library is that of abstracting the description of high level algorithms from low level parallelization issues. Algorithms written in FermiQCD are automatically parallel. The parallelization is based on a distributed memory model and communications algorithms are optimized for distributed memory machines such as PC clusters. The actual communication functions are implemented in MPI[10] (Message Passing Interface). The basic classes that constitute the library are: the lattice, the site and the field. A lattice object (`lattice`) represents

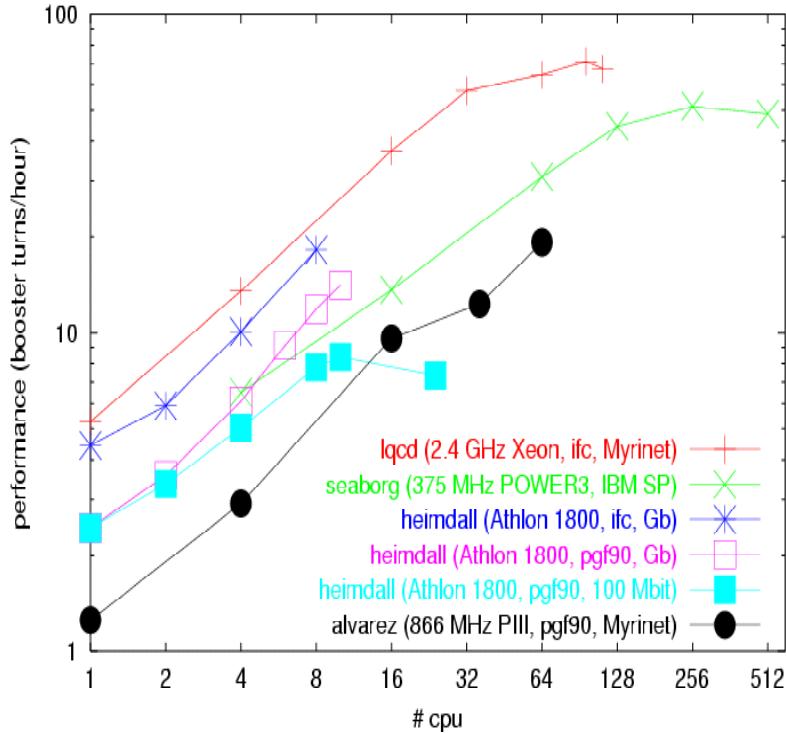


Figure D.4: Synergia performance modeling space charge effects in the FNAL Booster, on various platforms . The simulation used 2.7M particles on a 653 grid, with 100 space-charge kicks per cycle.

the discretized space over which the physical problem is defined. A lattice object can be represented as a graph; i.e. as a collection of points connected by some topology. A typical lattice has a mesh topology with periodic boundary conditions. A site object (`x`) is a variable representing a point on the lattice. Since any lattice is embeddable in a multidimensional space, any site object has methods to access its coordinates and to move along each space direction. A field object (`phi`) is a data structure that lives on the lattice. Each point of the lattice is associated to an instance of the data structure. For example a field of integers would correspond to allocating an integer variable for each point of the lattice. Each field variable can be accessed by referencing a field object and a site object on the lattice (`phi(x)`).

When a lattice object is declared it is automatically partitioned in parallel over the distributed memory architecture and each computing node determines its optimal communication patterns according to a heuristic algorithm based on some empirical rules. These rule require that a) node  $A$  and  $B$  communicate only if  $A$  and  $B$  contain sites that are neighbors according to the lattice topology; b) each node  $A$  is involved at most in one send and one receive at the same time; c) two nodes,  $A$  and  $B$ , never try to communicate with the same node  $C$  at the same time; d) if node  $A$  needs to communicate to node  $B$ , the data is packed and communication is done with a single send instruction; e) no temporary buffers are used when receiving data.

When a field object is defined each computing node allocates field variables of each site that is local to the node plus additional buffers for those neighbors of the local sites that reside on different nodes. The field variables stored in the buffers are updated only when required and communication is performed according to the communication patters determined by the underlying lattice object.

In the case of a computer simulation of the beam dynamics in a beam pipe the following objects can be identified: a) a lattice object describing the discretized space in the beam pipe, having periodic (or open, depending on the application) boundary conditions along the pipe direction; b) a field object representing the electromagnetic field in the pipe; c) a field object representing the particles of the beam in each cell (or box) of the discretized space. Additional auxiliary fields will be required to implement the solver algorithms and perform additional optimizations.

Once a logical map between the physical objects that represent the problem to be studied and the C++ objects implemented in FermiQCD is realized, the parallelization aspect of the problem is taken care of by the underlying library.

Our project consists of the following steps:

- Identify the optimal data structures to represent the particles and fields of the problem.
- Implement the relevant algorithms in FermiQCD.
- Implement a system for dynamically changing the parameters of the simulation in real time.
- Implement a crash/recovery system that allows a simulation program to resume automatically in case of crash.

- Benchmark and optimize the code.
- Develop analysis tools to extract real-time information from the simulation.

The solving algorithm can be summarized as follows:

- 1) compute E and B fields generated by the beam by solving the Poisson-Vlasov equation at fixed time (in parallel)
- 2) for each particle in the beam
  - determine the electro-magnetic force acting on the particle
  - interact with existing code that applies the transfer matrix
- 3) for each particle in the beam
  - if the particle has moved outside its original cell
    - transfer the particle to the appropriate cell
    - if the appropriate cell resides on a different node
      - perform an MPI communication
- 4) interact in real time with client programs,  
 perform analysis of data,  
 perform checks and IO to provide crash recovery

The first step involves solving the Poisson-Vlasov equation in parallel. This is a standard procedure and can be achieved by implementing a parallel PDE solver. FermiQCD already includes a parallel minimum residue and a parallel stabilized bi-conjugate inverter that can be extended for this scope (Fig.D.5).

The second step involves changing the momenta of particles and applying the external map, as prescribed by the split operator technique.

The third step of the simulation involves the following check. If a particle has moved outside the lattice cell where it was in the preceding iterations (Fig.D.6), the particle has to be moved into a new cell. In our simulation a particle will be an object allocated (within a field) on a lattice site. Moving a particle on a different cell means that a particle is de-allocated from one lattice site and is re-allocated on a different lattice site. If the old site and the new site reside on different processing nodes an explicit call to communication functions will be required.

The fourth step involves communication through a pipe or socket. In fact we envision a client-server architecture. The server program is the one just described. It runs in parallel on a cluster or other distributed memory machine and performs

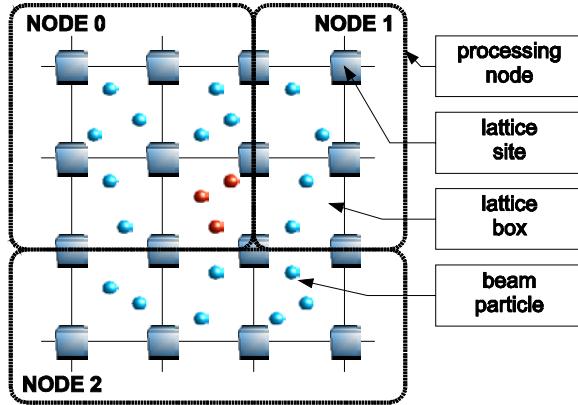


Figure D.5: Lattice schematics

the simulation. The client program provides a remote interface to the server program and communicates with it through a pipe or socket. Simulation parameters will be inserted through the client program. At the same time the server program will return information about the simulation (for example charge distributions, convergence precision, elapsed time) so that they can be displayed and analyzed in real time.

The server program will also perform memory checks and input/output in order to guarantee crash recovery capability. In a parallel environment there are many circumstances that may cause the computation to abort, for example a network failure or a node failure. In the event this occurs the client program should be able to resume the computation at (almost) the same point where it was interrupted without data loss and without requiring the intervention of the user.

The server program will be written in C++ on top of FermiQCD; the client program will be written in Java (in order to run on a web browser) and/or Python (to ensure compatibility with the present system).

### D.3. Proposed Timetable and Merit Review Criteria

#### Year 1

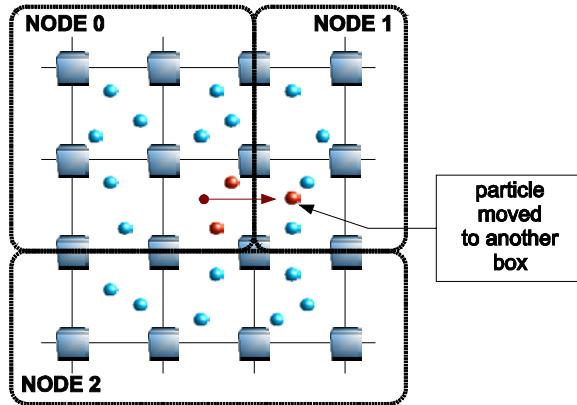


Figure D.6: Particle moves on a different cell

- Analysis of existing algorithms and software implementation, their strengths and weaknesses
- Development of an optimal parallel data-structure, appropriate for the description of the variables of the problem (particles and fields).
- Begin the development of an optimal parallel algorithms to solve the Poisson-Vlasov equation.

## Year 2

- Finalize the development of an optimal parallel algorithms to solve the Poisson-Vlasov equation
- Implementation of the data-structure and the algorithms using C++ and FermiQCD
- Implementation of serialization algorithms for the data
- Begin integration with existing software.

## Year 3

- Finalize integration with existing software

- Analysis of the possible crash cases
- Development of an optimal crash recovery algorithm
- Implementation of the crash recovery algorithm on top of the solver
- Intensive testing and benchmarks
- Write basic documentation.

#### **Year 4**

- Begin utilization of the software in real simulation applications
- Developing an appropriate client-server communication protocol
- Transform the solver program into a server program
- Develop a GUI-based client program to manage the solver
- Write more extensive documentation.

#### **D.4. Requested Support**

We estimate the completion of this project will require 4 years. In order to carry on this project we request funding for the 4 years that will support:

- one full time graduate student (tuitions plus salary)
- travel for the PI and the graduate student
- summer salary for the PI
- one summer internship at Fermilab (sub-awarded to Fermilab)
- travel for the Co-PI (sub-awarded to Fermilab)
- associated fringe benefits and overhead.

## **D.5. Educational Impact**

The project will provide the opportunity for one graduate and some undergraduate students of diverse backgrounds to participate in solving a very practical computational expensive problem that encompasses many disciplines including physics, math, computer science, software engineering and distributed system.

DePaul University has grown to be one of the largest universities in the Chicago area and, according to 2002 data, DePaul CTI (Computer Science, Telecommunications, and Information Systems) offers the largest computer science program in the country. The undergraduate program enrolls 2,118 students and offers six different degrees. More than 1039 students are enrolled in the graduate program, which offers nine different master's degrees. DePaul CTI also features a Ph.D. program in computer science that currently enrolls about 50 students. DePaul CTI employs more than 80 full-time faculty and more than 150 part-timers.

The university has shown a growing commitment to cutting edge research and it therefore provides an ideal context for implementing this research project. Research labs include Software Engineering, Human-Computer Interaction, Multi-media, Programming Languages, and Artificial Intelligence labs.

Moreover DePaul University has a highly diverse student body, and in a 1999 Princeton Review was ranked 2nd out of 331 colleges surveyed in student diversity, and 7th in interaction between students of different races and socio-economic classes. DePaul was also recognized in a 2001 issue of "Black Issues in Higher Education" as one of the top 100 universities in America for awarding bachelor's degrees to minorities. DePaul has accomplished this through its ongoing commitment to providing a quality education to all students, by offering special programs such as the NSF funded scholarships for low-income students entering the IT field, and by creating a learning environment in which interaction between students and faculty members is highly valued. The investigators will make a significant effort to provide research opportunities for a broad range of student participants, and will provide active mentoring in order to facilitate their success.

The project will impact curriculum at DePaul in the Computer Science area. For example students interested in the Fermilab internship program (supported by this grant) will be trained at DePaul and will contribute to research activities while working for their degree.

This project may also have the effect to broaden access to education. In fact one undergraduate per year will have access to the internship and he/she may decide to use the salary to pay for his/her further studies. This opportunity will

be particularly appealing to low income students with no other sources of funding.

#### **D.6. Results from previous NSF Support**

The PI and Co-PI did not receive previous NSF support.

## E. REFERENCES CITED

- [1] DoE Review 2003. [http://www-bd.fnal.gov/doereview03/  
docs/DOE\\_closeout\\_23july03.pdf](http://www-bd.fnal.gov/doereview03/docs/DOE_closeout_23july03.pdf)
- [2] Synergia: a hybrid, parallel beam dynamics code with 3D space charge, by J. Amundson, P. Spentzouris (Fermilab). FERMILAB-CONF-03-126-E, Jul 2003. Presented at Particle Accelerator Conference (PAC 03), Portland, Oregon, 12-16 May 2003.
- [3] J.Qiang, R.D.Ryne, S.Habib and V.Decyk, J. Comput. Phys. **163**, 434 (2000)
- [4] L.Michelotti, FERMILAB-CONF-91-159 and FERMILAB-FN-535-REV
- [5] www.fermiqcd.net by Massimo Di Pierro. Aug 2003. Proceedings of the Lattice 2003 conference, Tsukuba, Japan.
- [6] FermiQCD: a Toolkit for Parallel Lattice QCD Applications. 19th International Symposium on Lattice Field Theory (Lattice 2001), Berlin, Germany, 19-24 Aug 2001. Published in Nuclear Physics Proc.Suppl.106:1034-1036,2002. e-Print Archive: hep-lat/0110116
- [7] Matrix Distributed Processing and FermiQCD. 7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2000), Batavia, Illinois, 16-20 Oct 2000. e-Print Archive: hep-lat/0011083
- [8] A bird's eye view of Matrix Distributed Processing, ICCSA Conference Proceedings, Springer Ed. e-Print Archive: cs.dc/0303031
- [9] Matrix Distributed Processing. Computer Physics Communications 141, 2001. e-Print Archive: hep-lat/0004007
- [10] P.S. Pacheco, Parallel Programming with MPI, San Francisco CA, Morgan Kaufmann, 1997

## **F.1. PI: Massimo Di Pierro**

Address: School of Computer Science, Telecommunications and Information Systems, DePaul University, 243 S Wabash Av., Chicago, IL 60604 -  
Phone: 1-312-375-6536 - Fax: 1-312-362-6116 - Email: mdipierro@cs.depaul.edu  
- Web Page: <http://www.phoenixcollective.org/mdp/index.html>

### **F.1.1. Professional Preparation**

- Ph.D. in Physics, June 1999. University of Southampton, Southampton, UK
- BS-MS in Physics, June 1996. University of Pisa, Pisa, Italy

### **F.1.2. Appointments**

- Assistant Professor, DePaul University, School of Computer Science, Telecommunications and Information Systems, 2002-present
- Postdoctoral fellow, Fermilab, Theory Division (working on Lattice QCD), 1999-2002
- Graduate student, University of Southampton (Southampton, UK), 1996-1999

### **F.1.3. Related Publications**

- [www.fermiqcd.net](http://www.fermiqcd.net) by Massimo Di Pierro. Aug 2003. Proceedings of the Lattice 2003 conference, Tsukuba, Japan.
- A bird's eye view of Matrix Distributed Processing ICCSA Conference Proceedings, Springer Ed. e-Print Archive: cs.dc/0303031
- FermiQCD: a Toolkit for Parallel Lattice QCD Applications. 19th International Symposium on Lattice Field Theory (Lattice 2001), Berlin, Germany, 19-24 Aug 2001. Published in Nuclear Physics Proc.Suppl.106:1034-1036,2002. e-Print Archive: hep-lat/0110116
- Matrix Distributed Processing and FermiQCD. 7th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT 2000), Batavia, Illinois, 16-20 Oct 2000. e-Print Archive: hep-lat/0011083
- Matrix Distributed Processing. Computer Physics Communications 141, 2001. e-Print Archive: hep-lat/0004007

### **F.1.4. Other Publications**

- High Precision Lattice QCD Confront Experiment by HPQCD Collaboration and UKQCD Collaboration and MILC Collaboration and Fermilab Lattice Collaboration

- (C.T.H. Davies et al.). Apr 2003. 4pp. e-Print Archive: hep-lat/0304004
- Excited Heavy-Light Systems and Hadronic Transitions by Massimo Di Pierro, Estia Eichten (Fermilab). FERMILAB-PUB-01-033-T, Apr 2001. 42pp. Published in Physical Review D64:114004,2001. e-Print Archive: hep-ph/0104208
  - An Exploratory Lattice Study of Spectator Effects in Inclusive Decays of  $\Lambda_b$  Baryon by UKQCD collaboration (Massimo Di Pierro et al.). 12pp. Published in Physics Letters B468:143,1999. e-Print Archive: hep-lat/9906031
  - Toward a Lattice Determination of the  $g_{B^*B\pi}$  Coupling by UKQCD Collaboration (G.M. de Divitiis et al.). 20pp. Published in Journal of High Energy Physics 9810:010,1998. e-Print Archive: hep-lat/9807032
  - Mass, Confinement and CP Invariance in the Seiberg-Witten Model by Massimo Di Pierro and Kenichi Konishi. 11pp. Published in Physics Letters B388:90-96,1996. e-Print Archive: hep-th/9605178

### **F.1.5. Synergistic Activities**

Massimo Di Pierro is author of the following computer programs/libraries:

- Matrix Distributed Processing. A C++ toolkit for fast development of parallel applications. Project web page: [http://www.phoenixcollective.org/mdp/index\\_mdp.html](http://www.phoenixcollective.org/mdp/index_mdp.html)
- FermiQCD. A collection of parallel programs and algorithms for Lattice Computations. The library is currently used at Fermilab and other research centers around the world. Project web page: <http://www.fermiqcd.net>
- Algorithm Animator. A didactic software distributed with the book “Algorithms” written by R. Johnsonbaugh and M. Schaefer, published by Prentice Hall, 2003. Project web page: [http://www.phoenixcollective.org/mdp/index\\_csc321.html](http://www.phoenixcollective.org/mdp/index_csc321.html)

Massimo Di Pierro has thought Object Oriented Programming in C++, Design and Analysis of Algorithms, Foundations of Computer Science II and has developed a class on Monte Carlo Simulations.

### **F.1.6. Collaborators & Other Affiliations**

Chris Sachrajda, University of Southampton, Southampton, UK (Ph.D. advisor); Adriano Di Giacomo, University of Pisa, Pisa, Italy (Undergraduate advisor); Kenichi Konishi, University of Pisa, Pisa, Italy (Undergraduate thesis advisor); Other collaborators in alphabetic order: Christine Davies, Luigi Del Debbio, Giulia De Divitiis, Alex Dougall, Aida El-Khadra, Estia Eichten, Jonathan Flynn, Steven Gottlieb, Andreas Kronfeld, Peter Lepage, Paul Mackenzie, Masataka Okamoto, Mehmet Oktay, James Simone.

## **F.2. Co-PI: Panagiotis Spentzouris**

Address: Fermilab CD/CEPA, MS 234, PO Box 500, Batavia, IL 60510 -  
Phone: 1-630-840-4342 - Email: spentz@fnal.gov

### **F.2.1. Professional Preparation**

- Ph.D. in Physics, 1994. Northwestern University, Evanston, IL. Experiment: E665, Fermilab. Thesis: Measurement of the cross-section ratio  $\sigma_n/\sigma_p$ . Advisor: Prof. Heidi Schellman
- Physics Diploma, 1987. University of Athens, Greece. Experiment: DELPHI, CERN. Thesis: Fast Monte Carlo for simulating events at the BARREL RICH detector of DELPHI Advisor: Prof. Christine Kourkoumelis

### **F.2.2. Appointments**

- Group Leader, Simulation Group of the FNAL Computational Physics Department, 2000-present
- Scientist I, Fermilab. 2003-present
- Associate Scientist, Fermilab. 1998-2003
- Research projects: Neutrino Factory, E815 (NuTeV), E898 (MiniBooNE) and study of collective effects in particle accelerators.
- Associate Research Scientist, Columbia University. 1997-1998
- Main Research Project: E815 (NuTeV) at Fermilab
- Postdoctoral Research Associate, Columbia University 1994-1997.
- Main Research Project: E815 (NuTeV) at Fermilab

### **F.2.3. Awards**

- DOE SciDAC grant recipient, 2001-2003

### **F.2.4. Selected Publications**

- Synergia: a hybrid, parallel beam dynamics code with 3D space charge, by J. Amundson, P. Spentzouris (Fermilab). FERMILAB-CONF-03-126-E, Jul 2003. Presented at Particle Accelerator Conference (PAC 03), Portland, Oregon, 12-16 May 2003.
- FNAL Booster: experiment and modeling, by P. Spentzouris, J. Amundson (Fermilab). FERMILAB-CONF-03-127, Jun 2003. Presented at Particle Accelerator Conference (PAC 03), Portland, Oregon, 12-16 May 2003.

- Beam modeling tools for Geant4 (and neutrino source applications), by V. Daniel Elvira, P. Lebrun, P. Spentzouris (Fermilab). FERMILAB-PUB-03-133-E, May 2003. 22pp. Submitted to JCTPA
- Report of the Snowmass T7 working group on high performance computing, By K. Ko (SLAC), R. Ryne (LBL, Berkeley), P. Spentzouris (Fermilab). SLAC-PUB-9477, SNOWMASS-2001-T7001, Jun 2001.
- Design and simulation of muon ionization cooling channels for the Fermilab neutrino factory feasibility study, by J. Monroe, P. Spentzouris, V. Balbekov, P. Lebrun (Fermilab), G. Penn, C. Kim, E.S. Kim (LBL, Berkeley), D.M. Kaplan (IIT, Chicago), Phys.Rev. ST Accel. Beams 4:041301,2001
- Calibration of the Fermilab Booster ionization profile monitor, by J. Amundson, J. Lackey, P. Spentzouris, G. Jungman, and L. Spentzouris, Phys. Rev. ST Accel. Beams 6, 102801 (2003)
- Precision Electroweak Measurements From Nutev, by P. Spentzouris [NuTeV Collaboration], Acta Phys. Polon. B 33, 3843 (2002).
- A precise determination of electroweak parameters in neutrino nucleon scattering, By NuTeV Collaboration (G.P. Zeller et al.), Phys.Rev.Lett. 88:091802, 2002
- Experiments at Fermilab after MINOS, P. Spentzouris, Nucl.Phys. Proc.Suppl. 100:204-206, 2001.
- Precise measurement of dimuon production cross-sections in  $\nu\mu$ -Fe AND  $\bar{\nu}\mu$ -Fe deep inelastic scattering at the Tevatron, by NuTeV Collaboration (M. Goncharov, P. Spentzouris et al.), Phys.Rev.D64:112006,2001

### **F.2.5. Professional Service**

- Fermilab Computational Science Fellowship Selection Committee, Aug 2002
- Conference on Underground Service, Oct 2001. Co-convener of the Long Baseline Oscillation Working Group.
- Snowmass 2001. July 2001. Co-convener of the High Performance Computing Accelerator Simulation Working Group.
- Reviewer for the Kansas DOE-EPSCoR program, Oct 1999-Jan 2000
- DOE Strategic Simulation Initiative meeting, March 1999, and proposal writing.

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 1**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - none</b>		0.00	0.00	2.00	\$ 19,227	\$
2. <b>Panagiotis Spentzouris - none</b>		0.00	0.00	0.00	0	
3.						
4.						
5.						
6. ( 0 ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		0.00	0.00	0.00	0	
7. ( 2 ) TOTAL SENIOR PERSONNEL (1 - 6)		0.00	0.00	2.00	19,227	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00	0	
2. ( 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00	0	
3. ( 1 ) GRADUATE STUDENTS					13,500	
4. ( 0 ) UNDERGRADUATE STUDENTS					0	
5. ( 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					0	
6. ( 0 ) OTHER					0	
TOTAL SALARIES AND WAGES (A + B)					32,727	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					1,471	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					34,198	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					0	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					1,500	
2. FOREIGN					1,500	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ 0						
2. TRAVEL 0						
3. SUBSISTENCE 0						
4. OTHER 0						
TOTAL NUMBER OF PARTICIPANTS ( 0 )				TOTAL PARTICIPANT COSTS	0	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					0	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					0	
3. CONSULTANT SERVICES					0	
4. COMPUTER SERVICES					0	
5. SUBAWARDS					12,682	
6. OTHER					13,104	
TOTAL OTHER DIRECT COSTS					25,786	
H. TOTAL DIRECT COSTS (A THROUGH G)					62,984	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Fringe Benefits (Rate: 52.0000, Base: 1471) (Cont. on Comments Page)</b>						
TOTAL INDIRECT COSTS (F&A)					17,783	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					80,767	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ 80,767	\$
M. COST SHARING PROPOSED LEVEL \$ 0		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

1 \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

## **SUMMARY PROPOSAL BUDGET COMMENTS - Year 1**

---

**\*\* I- Indirect Costs**  
**Salaries (Rate: 52.0000, Base 32727)**

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 2**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - none</b>		0.00	0.00	2.00	\$ 19,996	\$
2. <b>Panagiotis Spentzouris - none</b>		0.00	0.00	0.00	0	
3.						
4.						
5.						
6. ( 0 ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		0.00	0.00	0.00	0	
7. ( 2 ) TOTAL SENIOR PERSONNEL (1 - 6)		0.00	0.00	2.00	19,996	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00	0	
2. ( 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00	0	
3. ( 1 ) GRADUATE STUDENTS					14,040	
4. ( 0 ) UNDERGRADUATE STUDENTS					0	
5. ( 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					0	
6. ( 0 ) OTHER					0	
TOTAL SALARIES AND WAGES (A + B)					34,036	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					1,529	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					35,565	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					0	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					1,500	
2. FOREIGN					1,500	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ 0						
2. TRAVEL 0						
3. SUBSISTENCE 0						
4. OTHER 0						
TOTAL NUMBER OF PARTICIPANTS ( 0 )				TOTAL PARTICIPANT COSTS	0	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					0	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					0	
3. CONSULTANT SERVICES					0	
4. COMPUTER SERVICES					0	
5. SUBAWARDS					13,147	
6. OTHER					13,628	
TOTAL OTHER DIRECT COSTS					26,775	
H. TOTAL DIRECT COSTS (A THROUGH G)					65,340	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Fringe Benefits (Rate: 52.0000, Base: 1529) (Cont. on Comments Page)</b>						
TOTAL INDIRECT COSTS (F&A)					18,494	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					83,834	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ 83,834	\$
M. COST SHARING PROPOSED LEVEL \$ 0		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

2 \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

## **SUMMARY PROPOSAL BUDGET COMMENTS - Year 2**

---

**\*\* I- Indirect Costs**  
**Salaries (Rate: 52.0000, Base 34036)**

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 3**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - none</b>		0.00	0.00	0.00	\$ 20,795	\$
2. <b>Panagiotis Spentzouris - none</b>		0.00	0.00	0.00	0	
3.						
4.						
5.						
6. ( 0 ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		0.00	0.00	0.00	0	
7. ( 2 ) TOTAL SENIOR PERSONNEL (1 - 6)		0.00	0.00	0.00	20,795	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00	0	
2. ( 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00	0	
3. ( 1 ) GRADUATE STUDENTS					14,601	
4. ( 0 ) UNDERGRADUATE STUDENTS					0	
5. ( 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					0	
6. ( 0 ) OTHER					0	
TOTAL SALARIES AND WAGES (A + B)					35,396	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					1,591	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					36,987	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					0	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					1,500	
2. FOREIGN					1,500	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ 0						
2. TRAVEL 0						
3. SUBSISTENCE 0						
4. OTHER 0						
TOTAL NUMBER OF PARTICIPANTS ( 0 )				TOTAL PARTICIPANT COSTS	0	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					0	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					0	
3. CONSULTANT SERVICES					0	
4. COMPUTER SERVICES					0	
5. SUBAWARDS					13,634	
6. OTHER					14,173	
TOTAL OTHER DIRECT COSTS					27,807	
H. TOTAL DIRECT COSTS (A THROUGH G)					67,794	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Fringe Benefits (Rate: 52.0000, Base: 1591) (Cont. on Comments Page)</b>						
TOTAL INDIRECT COSTS (F&A)					19,233	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					87,027	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ 87,027	\$
M. COST SHARING PROPOSED LEVEL \$ 0		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

## **SUMMARY PROPOSAL BUDGET COMMENTS - Year 3**

---

**\*\* I- Indirect Costs**  
**Salaries (Rate: 52.0000, Base 35397)**

**SUMMARY  
PROPOSAL BUDGET**

**YEAR 4**

ORGANIZATION <b>DePaul University</b>		FOR NSF USE ONLY				
		PROPOSAL NO.		DURATION (months)		
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.				
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months			Funds Requested By proposer	Funds granted by NSF (if different)
		CAL	ACAD	SUMR		
1. <b>Massimo DiPierro - none</b>		0.00	0.00	0.00	\$ 21,627	\$
2. <b>Panagiotis Spentzouris - none</b>		0.00	0.00	0.00	0	
3.						
4.						
5.						
6. ( 0 ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		0.00	0.00	0.00	0	
7. ( 2 ) TOTAL SENIOR PERSONNEL (1 - 6)		0.00	0.00	0.00	21,627	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)						
1. ( 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00	0	
2. ( 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00	0	
3. ( 1 ) GRADUATE STUDENTS					15,185	
4. ( 0 ) UNDERGRADUATE STUDENTS					0	
5. ( 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					0	
6. ( 0 ) OTHER					0	
TOTAL SALARIES AND WAGES (A + B)					36,812	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					1,654	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					38,466	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)						
TOTAL EQUIPMENT					0	
E. TRAVEL      1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					1,500	
2. FOREIGN					1,500	
F. PARTICIPANT SUPPORT COSTS						
1. STIPENDS \$ 0						
2. TRAVEL 0						
3. SUBSISTENCE 0						
4. OTHER 0						
TOTAL NUMBER OF PARTICIPANTS ( 0 )				TOTAL PARTICIPANT COSTS	0	
G. OTHER DIRECT COSTS						
1. MATERIALS AND SUPPLIES					0	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					0	
3. CONSULTANT SERVICES					0	
4. COMPUTER SERVICES					0	
5. SUBAWARDS					14,147	
6. OTHER					14,740	
TOTAL OTHER DIRECT COSTS					28,887	
H. TOTAL DIRECT COSTS (A THROUGH G)					70,353	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) <b>Fringe Benefits (Rate: 52.0000, Base: 1654) (Cont. on Comments Page)</b>						
TOTAL INDIRECT COSTS (F&A)					20,003	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					90,356	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)					0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)					\$ 90,356	\$
M. COST SHARING PROPOSED LEVEL \$ 0		AGREED LEVEL IF DIFFERENT \$				
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY				
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION				
		Date Checked	Date Of Rate Sheet		Initials - ORG	

## **SUMMARY PROPOSAL BUDGET COMMENTS - Year 4**

---

**\*\* I- Indirect Costs**  
**Salaries (Rate: 52.0000, Base 36813)**

**SUMMARY  
PROPOSAL BUDGET**

**Cumulative**

		FOR NSF USE ONLY		
		PROPOSAL NO.		DURATION (months)
		Proposed	Granted	
ORGANIZATION <b>DePaul University</b>				
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR <b>Massimo DiPierro</b>		AWARD NO.		
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months		Funds Requested By proposer
		CAL	ACAD	Funds granted by NSF (if different)
1. <b>Massimo DiPierro - none</b>		0.00	0.00	4.00 \$ 81,645 \$
2. <b>Panagiotis Spentzouris - none</b>		0.00	0.00	0.00 0
3.				
4.				
5.				
6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)		0.00	0.00	0.00 0
7. ( 2 ) TOTAL SENIOR PERSONNEL (1 - 6)		0.00	0.00	4.00 81,645
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)				
1. ( 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00 0
2. ( 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00 0
3. ( 4 ) GRADUATE STUDENTS				57,326
4. ( 0 ) UNDERGRADUATE STUDENTS				0
5. ( 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0
6. ( 0 ) OTHER				0
TOTAL SALARIES AND WAGES (A + B)				138,971
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				6,245
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				145,216
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)				
TOTAL EQUIPMENT				0
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				6,000
2. FOREIGN				6,000
F. PARTICIPANT SUPPORT COSTS				
1. STIPENDS \$ 0				
2. TRAVEL 0				
3. SUBSISTENCE 0				
4. OTHER 0				
TOTAL NUMBER OF PARTICIPANTS ( 0 )			TOTAL PARTICIPANT COSTS	0
G. OTHER DIRECT COSTS				
1. MATERIALS AND SUPPLIES				0
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0
3. CONSULTANT SERVICES				0
4. COMPUTER SERVICES				0
5. SUBAWARDS				53,610
6. OTHER				55,645
TOTAL OTHER DIRECT COSTS				109,255
H. TOTAL DIRECT COSTS (A THROUGH G)				266,471
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)				
TOTAL INDIRECT COSTS (F&A)				75,513
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				341,984
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)			\$ 341,984	\$
M. COST SHARING PROPOSED LEVEL \$ 0		AGREED LEVEL IF DIFFERENT \$		
PI/PD NAME <b>Massimo DiPierro</b>		FOR NSF USE ONLY		
ORG. REP. NAME*		INDIRECT COST RATE VERIFICATION		
		Date Checked	Date Of Rate Sheet	Initials - ORG

C \*ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET

## F. BUDGET

### **F.1. Budget Description**

#### **F.1.1. Senior Personnel**

Dr. Massimo Di Pierro from DePaul University will direct the project. Dr. Massimo Di Pierro will be paid 2 month summer salary each year of the grant to supervise all aspects of the project. Through out the academic year, he will also devote time to the supervision of the graduate student and the undergraduate students. The summer salary is 2/9th of his current salary increased 4% each year to account for salary raises. In year 1 he will be paid \$19,227; in year 2 he will be paid \$19,996; in year 3 he will be paid \$20,795 and in year 4 he will be paid \$21,627

#### **F.1.2. Graduate student**

This project will require support for a graduate student pursuing a Ph.D. degree in Computer Science and working on the project. The duration of funding will be for 4 years. The student will be reimbursed course tuitions for 6 courses per year, a stipend for the nine months of the academic year. During the year 1 the student will receive \$13,104 for tuition (budget item G.6) and \$13,500 (budget item B.3) for total stipend (including summer salary). The support will be increased 4% each year. In year 2 the student will receive \$13,628 for tuitions and \$14,040 for stipend; in year 3 will he/she receive \$14,173 and \$14,601; and in year 4 he/she will receive \$14,740 and \$15,185.

#### **F.1.3. Benefits**

PI work will be completed over the summer. Fringe benefits rate for the summer is 7.65%

#### **F.1.4. Travel**

Each year the PI and Ph.D. student will travel to attend conferences in order to exchange ideas with colleagues and present the proposed project. We believe that other national laboratories that host particle accelerators may have an interest in the present project. \$1500 per person per year is budgeted to cover all travel expenses, half will be national travel and half will be international travel.

#### **F.1.5. Indirect costs**

DePaul University's federally negotiated indirect cost rate is 52% of salaries, wages and benefits.

#### **F.1.6. Sub-awards**

This project will be carried out in collaboration with Fermilab. Panagiotis Spentzouris from Fermilab will be Co-PI for the project. Fermilab will receive a subaward of \$12,682 (budget item G.5) during year 1, \$13,147 for during year 2, \$13,634 for year 3, and \$14,147 for year 4. The sub-award will cover travel expenses for the Co-PI, a three month summer internships for a DePaul undergraduate student and associate fringe benefits and indirect costs.

## **Current and Pending Support**

(See GPG Section II.D.8 for guidance on information to include on this form.)

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Investigator: <b>Massimo DiPierro</b>		Other agencies (including NSF) to which this proposal has been/will be submitted.
Support: <input type="checkbox"/> Current <input checked="" type="checkbox"/> Pending <input type="checkbox"/> Submission Planned in Near Future <input type="checkbox"/> *Transfer of Support Project/Proposal Title: <b>Parallel Algorithms for Particle Beam Simulation</b>		
Source of Support: <b>NSF 04-146</b> Total Award Amount: \$ <b>346,132</b> Total Award Period Covered: <b>09/15/04 - 09/14/08</b> Location of Project: <b>DePaul University CTI</b> Person-Months Per Year Committed to the Project.    Cal: <b>0.00</b> Acad: <b>0.00</b> Sumr: <b>2.00</b>		
Support: <input type="checkbox"/> Current <input checked="" type="checkbox"/> Pending <input type="checkbox"/> Submission Planned in Near Future <input type="checkbox"/> *Transfer of Support Project/Proposal Title: <b>High Performance Computing in the Classroom</b>		
Source of Support: <b>NSF 04-001</b> Total Award Amount: \$ <b>540,030</b> Total Award Period Covered: <b>08/15/04 - 08/14/07</b> Location of Project: <b>DePaul University</b> Person-Months Per Year Committed to the Project.    Cal: <b>0.00</b> Acad: <b>3.00</b> Sumr: <b>0.00</b>		
Support: <input type="checkbox"/> Current <input type="checkbox"/> Pending <input type="checkbox"/> Submission Planned in Near Future <input type="checkbox"/> *Transfer of Support Project/Proposal Title:		
Source of Support: Total Award Amount: \$                          Total Award Period Covered: Location of Project: Person-Months Per Year Committed to the Project.    Cal:                          Acad:                          Sumr:		
Support: <input type="checkbox"/> Current <input type="checkbox"/> Pending <input type="checkbox"/> Submission Planned in Near Future <input type="checkbox"/> *Transfer of Support Project/Proposal Title:		
Source of Support: Total Award Amount: \$                          Total Award Period Covered: Location of Project: Person-Months Per Year Committed to the Project.    Cal:                          Acad:                          Sumr:		
Support: <input type="checkbox"/> Current <input type="checkbox"/> Pending <input type="checkbox"/> Submission Planned in Near Future <input type="checkbox"/> *Transfer of Support Project/Proposal Title:		
Source of Support: Total Award Amount: \$                          Total Award Period Covered: Location of Project: Person-Months Per Year Committed to the Project.    Cal:                          Acad:                          Summ:		

\*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

## Current and Pending Support

**(See GPG Section II.D.8 for guidance on information to include on this form.)**

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Other agencies (including NSF) to which this proposal has been/will be submitted.

**Investigator:** **Panagiotis Spentzouris**

Support:  Current  Pending  Submission Planned in Near Future  \*Transfer of Support

Project/Proposal Title: **Parallel Algorithms for Particle Beam Simulation**

Source of Support: **NSF 04-146**

Total Award Amount: \$ **346,132** Total Award Period Covered: **09/15/04 - 09/14/08**

Location of Project: **DePaul University - Fermilab**

Person-Months Per Year Committed to the Project. Cal:**0.00** Acad:**0.00** Sumr: **0.00**

Support:  Current  Pending  Submission Planned in Near Future  \*Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support:  Current  Pending  Submission Planned in Near Future  \*Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support:  Current  Pending  Submission Planned in Near Future  \*Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support:  Current  Pending  Submission Planned in Near Future  \*Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Summ:

\*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

# I. FACILITIES, EQUIPMENT AND OTHER RESOURCES

## I.1. DePaul CTI

### I.1.1. Laboratories

The School of Computer Science, Telecommunications and Information Systems operates specialized laboratories for research and instruction in artificial intelligence, computer-supported collaborative work, distributed systems, high performance computing, human-computer interaction, information systems and electronic commerce, networked multimedia, software engineering and languages and telecommunications. The school offers a total of approximately 800 student laboratory workstations - most of which are Pentium III or Pentium IV based. These workstations run Windows 2000, Windows XP, Red Hat Linux and Sun Solaris operating systems. The School also operates seven high-performance multi-processor Xeon-based servers with large RAID drives and an IBM ES 9000/9221.

Two laboratories may particularly be relevant for this project:

- The software research lab, which has been in existence for a few years, allows users to test experimental programs without jeopardizing the rest of the system. Currently, the lab has six PCs for installation and development of software.
- The Multimedia Networking Research Laboratory (MNLAB). This laboratory houses 20 Multimedia Workstations, including 14 Sun Ultra 10 workstations running Solaris 7 and 4 PCs running Window NT. The MNLAB workstations are connected with 100 Mbps switch which connects to the MBONE. The lab is supported in part by CTI, Sun Micro Systems and IONA Technologies

The IS division also provides printers and overhead projects for labs and residence halls. In addition to 750 lab workstations, information services maintains

over 1,200 residence halls workstations and hundreds of faculty and staff computing platforms

### **I.1.2. Computers and Networks**

DePaul University maintains an extensive technological infrastructure available for students, faculty and staff. In addition, many departments maintain their own resources dedicated for use by its own constituents. The University's existing computer and information infrastructure is in an exceptionally strong position to support the proposed project.

DePaul University is connected by an OC3c ATM-based circuit to the Chicago SBC/AADS network access point (NAP) located in downtown Chicago. The Chicago NAP is one of the largest Internet exchanges in the world. DePaul University maintains approximately fifty peering sessions at the Chicago NAP with institutions of varying sizes and locations including the University's Internet2/Abilene connector MREN. DePaul was one of the first University's in the nation to deploy Juniper router technology at its Internet border, taking advantage of Juniper's award winning design and performance. DePaul University uses BGP, MBGP, PIM-SM and MSDP protocols with its external peering partners to support full line-rate IPv4 unicast and multicast routing.

DePaul University also maintains a physical and logically diverse backup Internet connection to UUNET approximately thirty-five miles north at its Barat Campus in Lake Forest, IL.

The DePaul University metropolitan area backbone network connects campuses at Chicago downtown, Chicago Lincoln Park, Barat campus, Lake Forest, Naperville and Rolling Meadows using Gigabit Ethernet service with an inter-campus speed of 1 gigabit/second.

Almost all of the wired local area networks (LANs) on each DePaul University campus run at 10 and 100 Mb/s switched Ethernet with gigabit uplinks to core backbone devices linking buildings and campuses together. There are also approximately thirty-five 802.11b wireless LAN access points deployed throughout the University network with more to come as demand increases.

Fully transparent IPv4 unicast and multicast connectivity is available by default for all end hosts using DePaul University's public 140.192.0.0/16 IP address block. There are over 250 subnets within the entire DePaul University network. Critical networks and services are protected with network firewalls, host firewalls or various intrusion detection systems.

All primary uplinks and interconnection points on the DePaul network are managed through SNMP. Monitoring of link utilization, packet drop rates, error rates, device environments, protocol usage and summarized flow statistics are collected. Links and devices are also monitored for availability with alerts sent to pagers or to monitored email accounts. DePaul University's internal Networks and Telecom Group (NTG) in Information Services is responsible for installing, managing and monitoring all internet devices and connections. NTG consists of over twenty full-time staff members performing support duties on a 24x7 on-call basis for routers, LAN switches, cabling, telephony, network security, groupware servers and other various network services.

Each staff member is provided with one desktop computer and one laptop computer.

### **I.1.3. Other resources**

DePaul University provides library services at all campuses, with over 600,000 volumes and 8,000 serial holdings. Delivery of information and materials is linked through computer databases that are all available via Web-based interfaces. The Instructional Technology Development (ITD) Group is staffed with over a dozen members whose role is to help promote, train and develop technological solutions in support of education and research at the University. Library administrators and the ITD group are instrumental in helping bring technologies to faculty throughout the University.

The majority of classrooms are equipped with the latest recording and distance learning technologies including video cameras, document cameras, video cassette recorders, workstation screen capture software, white-board input and microphones. All online class content is automatically managed through the Black Board application including prerecorded video, audio and other multimedia materials used in class. DePaul University is undertaking numerous strides in developing next generation online course content that can be made accessible to students all over the country and throughout the world.

The School of Computer Science, Information Systems and Telecommunications is part of a large liberal-arts university including a School of Education, giving us ready access to the expertise of curriculum experts. Additionally, being centrally located in the Chicago metropolitan area provides ready access to the expertise of faculty from the University of Chicago, Northwestern, the University of Illinois at Chicago, Loyola University, and the Illinois Institute of Technology.

Further, DePaul and CTI enjoy a close relationship with corporations in Chicago, and either employ many IT professionals as adjunct faculty or have them as Alumni.

## **I.2. Fermilab**

Fermilab is a DOE national laboratory and it is operated by the Universities Research Association, Inc., a consortium of 89 research universities in the U.S. and abroad. Fermilab is the largest high-energy physics laboratory in the United States, and is second in the world only to CERN, the European Laboratory for Particle Physics.

Fermilab's Tevatron is the world's highest-energy particle accelerator and collider. In the Tevatron, counter-rotating beams of protons and antiprotons produce collisions allowing scientists to examine the most basic building blocks of matter, and the forces acting on them. Particle physics research has grown into an international effort, with experiment collaborations numbering in the hundreds.

### **I.2.1. Computers and Networks**

The Fermilab Computing Division furnishes and operates a laboratory wide network and several computing facilities, including central facilities in the Feynman Computer Center. These facilities include different types of parallel machine and, in particular, large computer clusters dedicated to simulation, reconstruction and analysis of scientific data. They also include data storage capacities of more than one Petabyte (in the form of robotic tape storage) and about hundreds of Terabytes of disk storage.

The Fermilab Computing Division leads more than 250 computer professionals, engineers, technicians and physicists in the Computing Division, whose work include R&D projects to prepare computing for future of High Energy Physics programs.

# DEPAUL UNIVERSITY



November 10, 2003

School of Computer Science,  
Telecommunications and  
Information Systems  
Helmut P. Epp, Ph.D.,  
Dean  
243 South Wabash Avenue  
Chicago, Illinois 60604-2301  
312/362-8760  
FAX: 312/362-5185

Dr. Massimo DiPierro  
DePaul University  
School of Computer Science, Telecommunications and Information Systems  
243 South Wabash Avenue  
Chicago, IL 60604

Dr. Panagiotis Spentzouris  
Fermilab  
P.O. Box 500  
Batavia, IL 60510-0500

Dear Massimo and Panagiotis,

I write in support of your proposal "Parallel Algorithms for Particle Beam Simulation". DePaul CTI has a growing commitment to cutting edge research and strongly encourages cooperation between our faculty and national laboratories such as Fermilab. This project provides a concrete example of such cooperation that, by bringing together the different expertise of scientists from DePaul and from Fermilab, will contribute to solve scientific problems of national and international interests. This and similar projects will affect DePaul's curricula in a positive way by giving students greater access to frontier research and will confront them with new challenging problems.

I trust that the PI and Co-PI have the required expertise to complete the proposed project and deliver more than satisfactory results. DePaul CTI strongly endorses this project and will provide all required support to those DePaul faculty and students which are involved.

Sincerely,

A handwritten signature in black ink that reads "Helmut Epp".

Helmut Epp



Fermilab

Fermi National Accelerator Laboratory  
Victoria A. White, • Head, Computing Division  
MS370 • P.O. Box 500 • Batavia, IL 60510  
Office: 630/840-3936 • Fax: 630/840-3785  
Email: white@fnal.gov • Cell: 630/774-9552

[computing.fnal.gov](http://computing.fnal.gov)

November 7, 2003

Professor Massimo DiPierro  
School of Computer Science  
DePaul University  
1 East Jackson Boulevard  
Chicago, IL 606042218

Dear Massimo,

I am really very pleased to write in support of your proposal “Parallel Algorithms for Particle Beam Simulation” and to affirm strong support from Fermilab for this collaborative work between Fermilab and DePaul. From the Fermilab side the work will be led by one of our brightest and best scientists, Panagiotis Spentzouris.

Fermi National Accelerator Laboratory, also known as Fermilab, is a US Department of Energy National Laboratory dedicated to basic scientific research. Its mission is to advance the understanding of the fundamental nature of matter and energy by providing leadership and resources for conducting research on the frontiers of high energy physics and related disciplines. The Laboratory’s Tevatron is the world’s highest-energy particle accelerator. The collaborations utilizing our facilities involve research programs at over 150 top tier research universities and institutions throughout the world, and support the work of 2000+ researchers. Fermilab is also participating in the construction of the Large Hadron Collider, a particle accelerator which will turn on at CERN in 2007.

Simulation of an entire accelerator complex in order to understand and improve performance and, eventually, to design new and innovative accelerators more cost effectively is an exciting and ambitious goal that we all look forward to realizing in the future. Today, some extremely interesting work and insights are coming out of work on simulation of parts of accelerator complexes, both at Fermilab, SLAC and elsewhere. The simulations are complex and the ability to parallelize the computations effectively and perform calculations with sufficient accuracy to simulate, and thereby provide insights into, the current performance of parts of our accelerator complex requires a tightly coupled effort between theoretical accelerator science, computer science and the experimental accelerator science that instruments and measures accelerator performance. I believe that considerable strides have been made in understanding accelerator behavior

by getting computer scientists and accelerator scientists to “team up” and work together. During my three and one half years working as a program manager and computing advisor in the Department of Energy’s Office Science I actively worked to encourage and fund such programs of work, under the SciDAC (Scientific Discovery through Advanced Computing) program. Since taking over as head of the Computing Division at Fermilab I have continued to support and encourage such work and hope to be able to provide more support from Fermilab in the coming years. There is a huge amount of work that could be done to develop and refine accurate simulations of parts of the Fermilab Accelerator complex. Much of this work, if done correctly, would provide a structure and tools for simulations of other complex accelerator systems. There are a large number of extremely challenging computer science issues and problems that must be solved in each of these potential programs of work.

This proposal focuses on one aspect of this work, namely the development and optimization of parallel solvers for high-fidelity beam dynamics simulations. The work will be immediately applicable to modeling the many-body, three-dimensional, and non-linear aspects of parts of our current accelerator complex. It will be applied to study space-charge effects in the Fermilab Booster and to developing a better understanding of beam-beam effects in the Fermilab Tevatron. However, the work will clearly have wider applicability and impact. I believe this to be an extremely worthwhile research area, with challenging problems from a computer science perspective.

I strongly support this work. I particularly encourage and support the interdisciplinary aspects of this work. I believe the educational opportunities for students to become involved in working at Fermilab on a real accelerator, solving problems that people really care about are exciting. At Fermilab we look forward to such involvement, especially from our neighboring universities and educational institutions and strive to provide opportunities for students to participate and learn.

Yours sincerely

A handwritten signature in black ink, appearing to read "Victoria A. White".

Victoria A. White  
Head, Computing Division  
Fermilab

This is a notification of an ITR letter of intent submission. The information below was collected:

Letter of Intent ID: 834

Title: Parallel Algorithms for Particle Beam Simulation

Primary Division: PHY

Secondary Division: CNS

Tertiary Division: DGE

PI Name: Massimo Di Pierro

PI Affiliation: DePaul University - School of Computer Science,  
Telecommunications and Information Systems

PI Affiliation Type: Academic

PI Email: mdipierro@cs.depaul.edu

Co-Pi Name: Panagiotis Spentzouris

Co-Pi Affiliation: Fermi National Accelerator Laboratory

Co-Pi Affiliation Type: Government Lab

Co-Pi Email: spentz@fnal.gov

#### Project Description:

The primary objective of our project is to develop and optimize parallel solvers for high-fidelity beam dynamics simulations which aim to model the many-body, three-dimensional, and non-linear aspects of present and future particle accelerators.

The proposed project is of crucial importance to the study of space-charge effects in high-intensity proton sources, such as the Fermilab Booster, and the study of beam-beam effects in hadron colliders, such as the Fermilab Tevatron. In order to accurately understand these effects, thousands of turns around the machines need to be simulated using millions of particles, thus the development of fast and efficient solvers is essential for the successful use of these models. The PI, Massimo Di Pierro, is a Professor of Computer Science at DePaul University and has been a Research Associate in the Fermilab Theory Group. He has developed FermiQCD, an open source library for parallel computations. The Co-PI, Panagiotis Spentzouris, is a leading scientist at Fermilab with extensive experience in the field of particle beam simulation. The PI and Co-PI have access to the resources required to complete this project.

This project will bring together Fermilab accelerator physicists who are developing such beam dynamics codes, and scientists from the School of Computer Science of DePaul University who have great expertise in the development of such solvers. The newly developed solvers will maximize the efficiency of the simulation, thus enabling researchers to better pursue accelerator design and performance studies while, at the same time, will provide an excellent research project for DePaul students.

The total requested funding, spread over a period of 3 years, amounts to \$346,132. This will support the PI (summer salary and travel only) and one full-time Ph.D. student dedicated 100% to the project. Part of the funding, \$53,610, will be sub-awarded to Fermilab and will support a 3 years summer internship

program for a DePaul undergraduate who will work on integrating the new parallel solver with existing Fermilab software.

This project is endorsed by Dr. Vicky White, Director of the Fermilab Computing Division, and Dr. Helmut Epp, Dean of the School of Computer Science of DePaul University.

Primary Priority Area: ASE

Primary Technical Focus Area: SIM

## Fermilab sub-award budget

11/10/2003 13:54 FAX 630 840 2900

FERMILAB DIRECTOR'S OFF.

002

NSF 96-115

Smith

ORGANIZATION <b>Fermilab</b> PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR	Total of 4 Years			
	FOR NSF USE ONLY		DURATION (MONTHS)	
	PROPOSAL NO.	AWARD NO.	Proposed	Granted
A. SENIOR PERSONNEL: PI/PD, Co-PI'S, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)				
0. First Name M Last Name Title Panagiotis Spentzouris Co-PI	CAL	ACAD	SUMR	Funds Requested By Proposer
4.80	0.00	0.00	\$0	
( <b>1</b> ) TOTAL SENIOR PERSONNEL (1-6)				
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)				
1. 0 ) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	\$0
2. 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	\$0
3. 0 ) GRADUATE STUDENTS	\$0			
4. 1 ) UNDERGRADUATE STUDENTS	\$26,447			
5. 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)	\$0			
6. 0 ) OTHER (ASSOCIATE RESEARCH SCIENTIST)	\$0			
TOTAL SALARIES AND WAGES (A+B)	\$2,276			
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)				
D. PERMANENT EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000)	\$0			
TOTAL EQUIPMENT	\$0			
E. TRAVEL	\$6,000			
1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)	\$0			
2. FOREIGN	\$0			
F. PARTICIPANT SUPPORT COSTS				
1. STIPENDS	\$0			
2. TRAVEL	\$0			
3. SUBSISTENCE	\$0			
4. OTHER	\$0			
( <b>1</b> ) TOTAL NUMBER OF PARTICIPANTS				
G. OTHER DIRECT COSTS	\$0			
1. MATERIALS AND SUPPLIES	\$0			
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION	\$6,000			
3. CONSULTANT SERVICES	\$0			
4. COMPUTERS SERVICES	\$0			
5. SUBAWARDS	\$0			
6. OTHER	\$0			
TOTAL OTHER DIRECT COSTS	\$0			
H. TOTAL DIRECT COSTS (A THROUGH G)				
I. INDIRECT COSTS (SPECIFY RATE AND BASE)				
Name of indirect cost item	Amount	Rate		
Salaries	\$30,723	30.35%		
Equipment & Materials & Supplies	\$6,000	16.05%		
Travel	\$6,000	10.00%		
TOTAL INDIRECT COSTS	\$0			
J. TOTAL DIRECT AND INDIRECT COSTS (H+I)	\$0			
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.I.)	\$0			
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)	\$53,610			
M. COST SHARING: PROPOSED LEVEL	AGREED LEVEL IF DIFFERENT \$			
P/I/PD TYPED NAME & SIGNATURE* <i>Panagiotis Spentzouris</i>	DATE	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION		
INST. REP. TYPED NAME & SIGNATURE* <i>Michael Witherell</i>	DATE	Date Checked	Date Rate of Sheet	Initials-ORG
*SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG II.B)				

NSF Form 1030 (10/97) Supersedes All Previous Editions

11/10/2003 13:55 FAX 630 840 2900

FERMILAB DIRECTOR'S OFF.

003

NSF 96-115

Smith

		YEAR 1	
		FOR NSF USE ONLY	
ORGANIZATION Fermilab		PROPOSAL NO.	DURATION (MONTHS)
PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR		AWARD NO.	Proposed      Granted
A. SENIOR PERSONNEL: PI/PD, Co-PI'S, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months	
0.	First Name M Last Name Title	CAL.	ACAD
	Panagiotis Spentzouris Co-PI	1.20	0.00
(1) TOTAL SENIOR PERSONNEL (1-6)		\$0	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)		\$0	
1.	0 ) POST DOCTORAL ASSOCIATES	0.00	0.00
2.	0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00
3.	0 ) GRADUATE STUDENTS	\$0	
4.	1 ) UNDERGRADUATE STUDENTS	\$6,600	
5.	0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)	\$0	
6.	0 ) OTHER (ASSOCIATE RESEARCH SCIENTIST)	\$0	
TOTAL SALARIES AND WAGES (A+B)		\$528	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)		\$528	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)		\$528	
D. PERMANENT EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000)		\$0	
TOTAL EQUIPMENT		\$0	
E. TRAVEL      1. DOMESTIC (INCL CANADA, MEXICO AND U.S. POSSESSIONS)		\$1,500	
2. FOREIGN		\$0	
F. PARTICIPANT SUPPORT COSTS		\$0	
1.	STIPENDS	\$0	
2.	TRAVEL	\$0	
3.	SUBSISTENCE	\$0	
4.	OTHER	\$0	
(1) TOTAL NUMBER OF PARTICIPANTS		\$0	
G. OTHER DIRECT COSTS		\$0	
1.	MATERIALS AND SUPPLIES	\$0	
2.	PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION	\$1,500	
3.	CONSULTANT SERVICES	\$0	
4.	COMPUTERS SERVICES	\$0	
5.	SUBAWARDS	\$0	
6.	OTHER	\$0	
TOTAL OTHER DIRECT COSTS		\$0	
H. TOTAL DIRECT COSTS (A THROUGH G)		\$12,682	
I. INDIRECT COSTS (SPECIFY RATE AND BASE)		\$0	
Name of indirect cost item		Amount	Rate
Salaries		\$7,128	30.35%
Equipment & Materials & Supplies		\$1,500	16.05%
Travel		\$1,500	10.00%
TOTAL INDIRECT COSTS		\$0	
J. TOTAL DIRECT AND INDIRECT COSTS (H+I)		\$0	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.J.)		\$0	
L. AMOUNT OF THIS REQUEST (J OR I MINUS K)		\$12,682	
M. COST SHARING: PROPOSED LEVEL		AGREED LEVEL IF DIFFERENT \$0	
PI/PD TYPED NAME & SIGNATURE Panagiotis Spentzouris		DATE 11/6/2003	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION
INST. REP. TYPED NAME & SIGNATURE Michael Witherell		DATE 11/10/03	Date Checked      Date Rate of Sheet      Initials-ORG

\*SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG III.B)

11/10/2003 13:56 FAX 630 840 2900

FERMILAB DIRECTOR'S OFF.

004

NSF 96-115

Smith

ORGANIZATION Fermilab	PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR	YEAR <u>2</u>		PROPOSAL NO. [REDACTED]	DURATION (MONTHS) Proposed [REDACTED] Granted [REDACTED]	FOR NSF USE ONLY			
		AWARD NO. [REDACTED]				NSF Funded Person-months			Funds Requested By Proposer
		CAL	ACAD			SUMR	\$0		
A. SENIOR PERSONNEL: PI/PD, Co-PI'S, Faculty and Other Senior Associates (List each separately with the A.7. show number in brackets)									
0. First Name M Last Name Title	Panagiotis Spentouris Co-PI	1.20	0.00	0.00	\$0				
(1) TOTAL SENIOR PERSONNEL (1-6)									
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)									
1. ( ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00	\$0				
2. ( ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)			0.00	0.00	\$0				
3. ( ) GRADUATE STUDENTS					\$0				
4. ( ) UNDERGRADUATE STUDENTS					\$6,930				
5. ( ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					\$0				
6. ( ) OTHER (ASSOCIATE RESEARCH SCIENTIST)					\$0				
TOTAL SALARIES AND WAGES (A+B)									
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)						\$554			
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)									
D. PERMANENT EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000)									
TOTAL EQUIPMENT									
E. TRAVEL	1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				\$1,500				
	2. FOREIGN				\$0				
F. PARTICIPANT SUPPORT COSTS									
1. STIPENDS		\$0							
2. TRAVEL		\$0							
3. SUBSISTENCE		\$0							
4. OTHER		\$0							
( ) TOTAL NUMBER OF PARTICIPANTS									
G. OTHER DIRECT COSTS									
1. MATERIALS AND SUPPLIES					\$0				
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					\$1,500				
3. CONSULTANT SERVICES					\$0				
4. COMPUTERS SERVICES					\$0				
5. SUBAWARDS					\$0				
6. OTHER					\$0				
TOTAL OTHER DIRECT COSTS									
H. TOTAL DIRECT COSTS (A THROUGH G)									
I. INDIRECT COSTS (SPECIFY RATE AND BASE)									
Name of indirect cost item	Amount	Rate							
Salaries	\$7,484	30.35%							
Equipment & Materials & Supplies	\$1,500	16.05%							
Travel	\$1,500	10.00%							
TOTAL INDIRECT COSTS									
J. TOTAL DIRECT AND INDIRECT COSTS (H+I)									
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.J.)						\$0			
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)						\$13,147			
M. COST SHARING: PROPOSED LEVEL			AGREED LEVEL IF DIFFERENT \$			\$0			
PI/PD TYPED NAME & SIGNATURE: <i>Panagiota Spentouris</i>		DATE	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION						
INST. REP. TYPED NAME & SIGNATURE: <i>Michael Witherell</i>		DATE 11/10/03	Date Checked	Date Rate of Sheet	Initiate-ORG				

NSF Form 1030 (10/97) Supersedes All Previous Editions

\*SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG III.B)

11/10/2003 13:57 FAX 630 840 2900

FERMILAB DIRECTOR'S OFF.

005

NSF 96-115

Smith

ORGANIZATION <b>Fermilab</b>	FOR NSF USE ONLY			YEAR      3
	PROPOSAL NO.		DURATION (MONTHS)	
	Proposed	Granted		
PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR	AWARD NO.			
A. SENIOR PERSONNEL: PI/PD, Co-PI'S, Faculty and Other Senior Associates (List each separately with title. A.7. show number in brackets)				
O. First Name M. Last Name Title	CAL	ACAD	SUMR	Funds Requested By Proposer
Panagiotis Spentzouris Co-PI	1.20	0.00	0.00	\$0
( ) TOTAL SENIOR PERSONNEL (1-6)				
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)				
1. ( ) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	\$0
2. ( ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	\$0
3. ( ) GRADUATE STUDENTS				\$0
4. ( ) UNDERGRADUATE STUDENTS				\$7,277
5. ( ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				\$0
6. ( ) OTHER (ASSOCIATE RESEARCH SCIENTIST)				\$0
TOTAL SALARIES AND WAGES (A+B)				
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				\$582
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)				
D. PERMANENT EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000)				
TOTAL EQUIPMENT				
E. TRAVEL	1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)			\$1,500
	2. FOREIGN			\$0
F. PARTICIPANT SUPPORT COSTS				
1. STIPENDS	\$0			
2. TRAVEL	\$0			
3. SUBSISTENCE	\$0			
4. OTHER	\$0			
( ) TOTAL NUMBER OF PARTICIPANTS				
G. OTHER DIRECT COSTS				
1. MATERIALS AND SUPPLIES	\$0			
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION	\$1,500			
3. CONSULTANT SERVICES	\$0			
4. COMPUTERS SERVICES	\$0			
5. SUBAWARDS	\$0			
6. OTHER	\$0			
TOTAL OTHER DIRECT COSTS				
H. TOTAL DIRECT COSTS (A THROUGH G)				
I. INDIRECT COSTS (SPECIFY RATE AND BASE)				
Name of Indirect cost item	Amount	Rate		
Salaries	\$7,859	30.35%		
Equipment & Materials & Supplies	\$1,500	16.05%		
Travel	\$1,500	10.00%		
TOTAL INDIRECT COSTS				
J. TOTAL DIRECT AND INDIRECT COSTS (H+I)				
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.J.)				
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)	\$13,634			
M. COST SHARING: PROPOSED LEVEL	AGREED LEVEL IF DIFFERENT \$			\$0
P/PD TYPED NAME & SIGNATURE* Panagiotis Spentzouris		DATE	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION	
INST. REP. TYPED NAME & SIGNATURE* Michael Witherell		DATE 11/10/03	Date Checked	Date Rate of Sheet
INITIALS-ORG				

NSF Form 1090 (10/87) Supersedes All Previous Editions

\*SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG III.B)

NSF 96-115

Smith

		YEAR 4		
		FOR NSF USE ONLY		
ORGANIZATION Fermilab	PRINCIPAL INVESTIGATOR/PROJECT DIRECTOR	PROPOSAL NO.		DURATION (MONTHS)
		Proposed	Granted	AWARD NO.
<b>A. SENIOR PERSONNEL: PI/PD, Co-PI'S, Faculty and Other Senior Associates</b> (List each separately with title, A.7. show number in brackets)		NSF Funded Person-months		Funds Requested By
0. First Name M Last Name Title		CAL	ACAD	SUMR
Panagiotis Spentzouris Co-PI		1.20	0.00	0.00
(1) TOTAL SENIOR PERSONNEL (1-6)				\$0
<b>B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)</b>				
1. 0 ) POST DOCTORAL ASSOCIATES		0.00	0.00	0.00
2. 0 ) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)		0.00	0.00	0.00
3. 0 ) GRADUATE STUDENTS				\$0
4. 1 ) UNDERGRADUATE STUDENTS				\$7,640
5. 0 ) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				\$0
6. 0 ) OTHER (ASSOCIATE RESEARCH SCIENTIST)				\$0
<b>TOTAL SALARIES AND WAGES (A+B)</b>				
<b>C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)</b>				\$611
<b>TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A+B+C)</b>				
<b>D. PERMANENT EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000)</b>				
<b>TOTAL EQUIPMENT</b>				
E. TRAVEL	1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)			\$1,500
	2. FOREIGN			\$0
<b>F. PARTICIPANT SUPPORT COSTS</b>				
1. STIPENDS		\$0		
2. TRAVEL		\$0		
3. SUBSISTENCE		\$0		
4. OTHER		\$0		
(0) TOTAL NUMBER OF PARTICIPANTS				
<b>G. OTHER DIRECT COSTS</b>				
1. MATERIALS AND SUPPLIES				\$0
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				\$1,500
3. CONSULTANT SERVICES				\$0
4. COMPUTERS SERVICES				\$0
5. SUBAWARDS				\$0
6. OTHER				\$0
<b>TOTAL OTHER DIRECT COSTS</b>				
<b>H. TOTAL DIRECT COSTS (A THROUGH G)</b>				
<b>I. INDIRECT COSTS (SPECIFY RATE AND BASE)</b>				
Name of Indirect cost item	Amount	Rate		
Salaries	\$8,252	30.35%		
Equipment & Materials & Supplies	\$1,500	18.05%		
Travel	\$1,500	10.00%		
<b>TOTAL INDIRECT COSTS</b>				
<b>J. TOTAL DIRECT AND INDIRECT COSTS (H+I)</b>				
<b>K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.I.)</b>				\$0
<b>L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)</b>				\$14,147
<b>M. COST SHARING: PROPOSED LEVEL</b>		AGREED LEVEL IF DIFFERENT \$		
PI/PD TYPED NAME & SIGNATURE* Panagiotis Spentzouris		DATE 11/6/2003	FOR NSF USE ONLY INDIRECT COST RATE VERIFICATION	
INST. REP. TYPED NAME & SIGNATURE Michael Witherell		DATE 11/10/03	Date Checked	Date Rate of Sheet
				Initials-ORG

NSF Form 1030 (10/97) Supersedes All Previous Editions

\*SIGNATURES REQUIRED ONLY FOR REVISED BUDGET (GPG III.B)



## Part V

# Sample Class Notes (not including books)



MASSIMO DI PIERRO

WEB AUTOMATION WITH PYTHON

EXPERTS4SOLUTIONS

Copyright 2008-2013 by Massimo Di Pierro. All rights reserved.

THE CONTENT OF THIS BOOK IS PROVIDED UNDER THE TERMS OF THE CREATIVE COMMONS PUBLIC LICENSE BY-NC-ND 3.0.

<http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode>

THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For more information about appropriate use of this material contact:

Massimo Di Pierro  
School of Computing  
DePaul University  
243 S Wabash Ave  
Chicago, IL 60604 (USA)  
Email: [massimo.dipierro@gmail.com](mailto:massimo.dipierro@gmail.com)

Library of Congress Cataloging-in-Publication Data:

ISBN: XXX-XXX-XXXX

Build Date: October 9, 2014

XXXXXXXXXXXX



# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
<b>2</b>	<b>Linux</b>	<b>13</b>
2.1	Access . . . . .	13
2.2	Bash Shell . . . . .	14
2.3	Piping stdin and stdout . . . . .	16
2.4	Installing prepackaged apps . . . . .	16
2.5	Emacs in a nutshell . . . . .	17
2.6	Installing binary packages . . . . .	18
2.7	Git Tutorial . . . . .	18
2.7.1	Importing a new project . . . . .	19
2.7.2	Viewing project history . . . . .	21
2.7.3	Review of GIT commands . . . . .	21
<b>3</b>	<b>Python</b>	<b>23</b>
3.0.4	Python vs Java and C++ syntax . . . . .	25
3.0.5	help, dir . . . . .	25
3.1	Python Types . . . . .	26
3.1.1	int and long . . . . .	27
3.1.2	float and decimal . . . . .	27
3.1.3	str . . . . .	30
3.1.4	list and array . . . . .	31
3.1.5	tuple . . . . .	33
3.1.6	dict . . . . .	35
3.1.7	set . . . . .	36

3.2	Python control flow statements . . . . .	38
3.2.1	for...in . . . . .	38
3.2.2	while . . . . .	40
3.2.3	if...elif...else . . . . .	41
3.2.4	try...except...else...finally . . . . .	41
3.2.5	def...return . . . . .	44
3.2.6	lambda . . . . .	46
3.3	Classes . . . . .	47
3.3.1	Special methods and operator overloading . . . . .	49
3.4	File input/output . . . . .	50
3.5	import modules . . . . .	51
3.5.1	math and cmath . . . . .	51
3.5.2	os . . . . .	52
3.5.3	sys . . . . .	53
3.5.4	datetime . . . . .	53
3.5.5	time . . . . .	54
3.5.6	urllib and json . . . . .	54
3.6	Regular Expressions . . . . .	55
4	OS funtions from Python	57
5	<b>BeautifulSoup</b>	61
5.1	Cached Downloads . . . . .	63
5.2	A simple web crawler . . . . .	63
6	<b>Mechanize</b>	67
7	<b>Fabric</b>	71
8	<b>User Management</b>	75
9	<b>EC2 and OpenStack</b>	77
9.1	EC2 Commad Line API Tools . . . . .	78
	<b>Bibliography</b>	81





# Preface

Some of the text in these notes has been collected from Wikipedia and other online sources.



1

# Introduction

Your account:

```
1 > ssh user<yourid>@140.192.30.237
```



## 2

# Linux

### 2.1 Access

In this course you will be using a Linux OS. There are many ways to use Linux. You can install it on your machine, with or without dual boot (dual boot is only recommended for experts), you can run it within a virtual machine (VMWare or VirtualBox for example), or you can access a remote Linux server.

Assuming you have a Linux account on a remote server you can connect to it using one of the following ways:

- From another Linux/Unix/Mac terminal:

```
1 > ssh <username>@<hostname>
```

for example:

```
1 > ssh thomas.anderson@the.matrix.net
```

- From Mac you can follow the same directions as Linux, but first you have to open Terminal. Search for “Terminal” in “Spotlight”.
- From Windows you need to download PuTTY from <http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe> and use the PuTTY GUI to insert the username and hostname.
- If you have a PythonAnywhere.com account, login there, then click on

## 14 WEB AUTOMATION WITH PYTHON

[Consoles] and [Start a new console][Bash]. PythonAnywhere provides a free Linux account in the cloud and they offer a Bash console shell you can access via the web browser.

### 2.2 Bash Shell

The “Bash shell” is a command line processor. It allows you to interactively type and execute commands and runs inside a text window. There are many different shells, a.k.a. command line processors and they have a different syntax. For example the Bash shell understand “bash” commands; the Python shell understand “Python” commands. Even Windows has a shell (the “Command Prompt”) which understand Windows commands. For now you will be using the Bash shell.

Example of commands:

What is my name?

```
1 > whoami
```

Make a file called “test.py” and write “print 1” in it:

```
1 > echo "print 1" > test.py
```

List all files in the current folder

```
1 > ls
```

(\* means everything and it is referred to as a wildcard). List all files ending in .py

```
1 > ls *.py
```

List all files ending in .py with more details

```
1 > ls -l *.py
```

Run the Python program test.py

```
1 > python test.py
```

Delete all files ending in

```
1 > rm *~
```

List the name of the current folder

```
1 > pwd
```

Navigate to the parent of the current folder

```
1 > cd ..
```

Navigate to the top folder (also known as file system root)

```
1 > cd /
```

Navigate back to your home folder

```
1 > cd ~
```

Make a new subfolder

```
1 > mkdir work
```

Enter the subfolder

```
1 > cd work
```

Create an empty file called “test.txt”

```
1 > touch test.txt
```

Step out of a folder

```
1 > cd ..
```

Remove a folder and all its content

```
1 > rm -r work
```

Print a list of all programs I am running

```
1 > ps
```

Print a list of all programs running on this machine by or other user

```
1 > ps -aux
```

List all running programs and sort them by resource consumption

```
1 > top
```

(press “q” to quit) Start a program and send it in background

```
1 > python &
```

(the program will be running but will not IO to console) Kill a program currently running

```
1 > kill -9 <pid>
```

(here <pid> is the process id of the program as listed by top and or ps).

Practice with the above commands.

`pwd`, `ls`, `cd`, `mkdir`, `echo`, `touch`, `rm`, `ps`, `top`.

## 16 WEB AUTOMATION WITH PYTHON

Use them to explore the file system.

Also notice that you can type an incomplete command and press [tab], the Bash shell will complete the command for you or give you a list of options. This is called autocomplete.

Some operations require superuser access and the system will refuse to perform the action otherwise. If you are an administrator of a system, you can perform an action with administrator by prepending `sudo` to any shell command.

For example - DO NOT DO THIS, EVER! - delete everything in the file system:

```
1 > cd /
2 > sudo rm -r *
```

At this point you have the ability to create and browse files, folders, and running processes. A running program may be constituted of one or more processes (parallel programs may do more than one thing at once, thus more than one process).

### 2.3 Piping stdin and stdout

When using the `echo` command we used the following example:

```
1 > echo "print 1" > test.py
```

Here `>` is a redirection operator. It takes the output of the `echo` command which would normally print to console, and send it to the `test.py` file. Similarly, one can redirect standard input which is normally received from the keyboard, and read it instead from a file using the `<` operator.

### 2.4 Installing prepackaged apps

At this point we need to learn how to install new programs and run them. How you do this depends on the version of Linux. I will assume Ubuntu. I will also assume you have superuser access else you will not be able to install programs.

The easiest way to install prepackaged apps for Ubuntu is:

```
1 > sudo apt-get install <appname>
```

For example

```
1 > sudo apt-get install mc
```

Once installed we can start it with

```
1 > mc
```

`mc` is a useful extension to the Bash shell which allows us to type Bash commands and, at the same time, browse the file system in a visual way.

We can also re-install existing apps to make sure we have the latest version, for example:

```
1 > sudo apt-get install python
```

Other useful apps are the following

```
1 > sudo apt-get install zip      # for compressing files
2 > sudo apt-get install unzip    # for uncompressing
3 > sudo apt-get install tar      # for packing files
4 > sudo apt-get install emacs    # a file editor
5 > sudo apt-get install python   # the python interpreter
6 > sudo apt-get install g++       # the C++ compiler
7 > sudo apt-get install wget     # to download stuff from web
8 > sudo apt-get install make     # to install source packages
9 > sudo apt-get install build-essential
10 > sudo apt-get install python2.7 # to make sure we have it
11 > sudo apt-get install python-dev # useful development tools
12 > sudo apt-get install git      # version control system
```

## 2.5 Emacs in a nutshell

Emacs is a text editor. It understand the syntax of many programming languages and can help with development. It does not have a GUI but only a text based UI and you can perform various operations using command line shortcuts. Remember the most important of them all `[ctrl]+g`. if you get into trouble it brings you back to the edit interface.

You can open a file with:

```
1 > emacs test.py
```

[crtl]+x+c	Save and Exit
[crtl]+s	Search
[meta]+<	scroll to beginning
[meta]+>	scroll to end
[meta]+X	command prompt
[tab]	automatically re-indent the current line

## 2.6 Installing binary packages

Some time you may need to install binary packages. They come in the form of tar g-zipped files. They look like: <appname>.tar.gz

They installed using the following process:

```
1 > gunzip <appname>.tar.gz
2 > tar xvf <appname>.tar
3 > cd <appname-folder>
4 > ./configure
5 > make
6 > sudo make install
```

Here <appname> is the name of the application being installed. <appname-folder> is the folder created by tar xvf .... The tar command extracts the content of the .tar file into a folder. The ./configure step can take additional command line options and one should consult the documentation of the specific package for more details. make compiles the application. make install moves the compiled files in the proper place. If everything worked fine you can delete the <appname>-folder>.

## 2.7 Git Tutorial

(from the Git man pages)

This tutorial explains how to import a new project into Git, make changes to it, and share changes with other developers.

If you are instead primarily interested in using Git to fetch a project, for

example, to test the latest version, you may prefer to start with the first two chapters of The Git User’s Manual.

It is a good idea to introduce yourself to Git with your name and public email address before doing any operation. The easiest way to do so is:

```
1 > git config --global user.name "Your Name Comes Here"
2 > git config --global user.email you@yourdomain.example.com
```

### 2.7.1 Importing a new project

Assume you have a tar-ball project.tar.gz with your initial work. You can place it under Git revision control as follows.

```
1 > tar xzf project.tar.gz
2 > cd project
3 > git init
```

Git will reply with:

```
1 Initialized empty Git repository in .git/
```

You’ve now initialized the working directory—you may notice a new directory created, named “.git”.

Next, tell Git to take a snapshot of the contents of all files under the current directory (note the .), with git add:

```
1 > git add .
```

This snapshot is now stored in a temporary staging area which Git calls the “index”. You can permanently store the contents of the index in the repository with git commit:

```
1 > git commit
```

This will prompt you for a commit message. You’ve now stored the first version of your project in Git.

Making changes Modify some files, then add their updated contents to the index:

```
1 > git add file1 file2 file3
```

You are now ready to commit. You can see what is about to be committed using git diff with the –cached option:

## 20 WEB AUTOMATION WITH PYTHON

```
1 > git diff --cached
```

(Without `-cached`, `git diff` will show you any changes that you've made but not yet added to the index.) You can also get a brief summary of the situation with `git status`:

```
1 > git status
2 # On branch master
3 # Changes to be committed:
4 #   (use `git reset HEAD <file>...` to unstage)
5 #
6 #modified:   file1
7 #modified:   file2
8 #modified:   file3
9 #
```

If you need to make any further adjustments, do so now, and then add any newly modified content to the index. Finally, commit your changes with:

```
1 > git commit
```

This will again prompt you for a message describing the change, and then record a new version of the project.

Alternatively, instead of running `git add` beforehand, you can use

```
1 > git commit -a
```

which will automatically notice any modified (but not new) files, add them to the index, and commit, all in one step.

A note on commit messages: Though not required, it's a good idea to begin the commit message with a single short (less than 50 character) line summarizing the change, followed by a blank line and then a more thorough description. The text up to the first blank line in a commit message is treated as the commit title, and that title is used throughout Git. For example, `git-format-patch(1)` turns a commit into email, and it uses the title on the Subject line and the rest of the commit in the body.

Git tracks content not files Many revision control systems provide an `add` command that tells the system to start tracking changes to a new file. Git's `add` command does something simpler and more powerful: `git add` is used both for new and newly modified files, and in both cases it takes a snapshot of the given files and stages that content in the index, ready for inclusion in the next commit.

### 2.7.2 Viewing project history

At any point you can view the history of your changes using

```
1 > git log
```

If you also want to see complete diffs at each step, use

```
1 > git log -p
```

Often the overview of the change is useful to get a feel of each step

```
1 > git log --stat --summary
```

Git is great for collaboration and allows creating and merging branches. When git projects are hosted on cloud services such as GitHub, you can send pull requests to other users which they can accept and reject. For more details consult the official documentation.

### 2.7.3 Review of GIT commands



3

Python

## 24 WEB AUTOMATION WITH PYTHON

Python is a general-purpose high-level programming language. Its design philosophy emphasizes programmer productivity and code readability. It has a minimalist core syntax with very few basic commands and simple semantics. It also has a large and comprehensive standard library, including an Application Programming Interface (API) to many of the underlying operating system (OS) functions. Python provides built-in objects such as linked lists (`list`), tuples (`tuple`), hash tables (`dict`), arbitrarily long integers (`long`), complex numbers, and arbitrary precision decimal numbers.

Python supports multiple programming paradigms, including object-oriented (`class`), imperative (`def`), and functional (`lambda`) programming. Python has a dynamic type system and automatic memory management using reference counting (similar to Perl, Ruby, and Scheme).

Python was first released by Guido van Rossum in 1991[? ]. The language has an open, community-based development model managed by the non-profit Python Software Foundation. There are many interpreters and compilers that implement the Python language, including one in Java (Jython), one built on .Net (IronPython) and one built in Python itself (PyPy). In this brief review, we refer to the reference C implementation created by Guido.

You can find many tutorials, the official documentation, and library references of the language on the official Python website. [1]

For additional Python references, we can recommend the books in ref. [? ] and ref. [? ].

You may skip this chapter if you are already familiar with the Python language.

### 3.0.4 Python vs Java and C++ syntax

	Java/C++	Python
assignment	$a = b;$	$a = b$
comparison	<code>if (a == b)</code>	<code>if a == b:</code>
loops	<code>for(a = 0; a &lt; n; a++)</code>	<code>for a in range(0, n):</code>
block	Braces {...}	indentation
function	<code>float f(float a) {</code>	<code>def f(a):</code>
function call	<code>f(a)</code>	<code>f(a)</code>
arrays/lists	<code>a[i]</code>	<code>a[i]</code>
member	<code>a.member</code>	<code>a.member</code>
nothing	<code>null / void*</code>	<code>None</code>

As in Java, variables which are primitive types (bool, int, float) are passed by copy but more complex types, unlike C++, are passed by reference.

### 3.0.5 help, dir

The Python language provides two commands to obtain documentation about objects defined in the current scope, whether the object is built-in or user-defined.

We can ask for `help` about an object, for example "1":

```

1  >>> help(1)
2  Help on int object:
3
4  class int(object)
5  |  int(x[, base]) -> integer
6  |
7  |  Convert a string or number to an integer, if possible. A floating point
8  |  argument will be truncated towards zero (this does not include a string
9  |  representation of a floating point number!) When converting a string, use
10 |  the optional base. It is an error to supply a base when converting a
11 |  non-string. If the argument is outside the integer range a long object
12 |  will be returned instead.
13 |
14 |  Methods defined here:
15 |
16 |  __abs__(...)
17 |      x.__abs__() <==> abs(x)

```

## 26 WEB AUTOMATION WITH PYTHON

18 ...

and, since "1" is an integer, we get a description about the `int` class and all its methods. Here the output has been truncated because it is very long and detailed.

Similarly, we can obtain a list of methods of the object "1" with the command `dir`:

```
1 >>> dir(1)
2 ['__abs__', '__add__', '__and__', '__class__', '__cmp__', '__coerce__',
3 '__delattr__', '__div__', '__divmod__', '__doc__', '__float__',
4 '__floordiv__', '__getattribute__', '__getnewargs__', '__hash__', '__hex__',
5 '__index__', '__init__', '__int__', '__invert__', '__long__', '__lshift__',
6 '__mod__', '__mul__', '__neg__', '__new__', '__nonzero__', '__oct__',
7 '__or__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdiv__',
8 '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__',
9 '__rshift__', '__rmod__', '__rmul__', '__ror__', '__rpow__', '__rrshift__',
10 '__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__',
11 '__str__', '__sub__', '__truediv__', '__xor__']
```

### 3.1 Python Types

Python is a dynamically typed language, meaning that variables do not have a type and therefore do not have to be declared. Variables may also change the type of value they hold through its life. Values, on the other hand, do have a type. You can query a variable for the type of value it contains:

```
1 >>> a = 3
2 >>> print(type(a))
3 <type 'int'>
4 >>> a = 3.14
5 >>> print(type(a))
6 <type 'float'>
7 >>> a = 'hello python'
8 >>> print(type(a))
9 <type 'str'>
```

Python also includes, natively, data structures such as lists and dictionaries.

### 3.1.1 int and long

There are two types representing integer numbers, they are `int` and `long`. The difference is `int` corresponds to the microprocessor's native bit length. Typically this is 32 bits and can hold signed integers in range  $[-2^{31}, +2^{31})$  while the `long` type can hold almost any arbitrary integer. It is important that Python automatically converts one into the other as necessary and you can mix and match the two types in computations. Here is an example:

```

1 >>> a = 1024
2 >>> type(a)
3 <type 'int'>
4 >>> b = a**128
5 >>> print(b)
6 20815864389328798163850480654728171077230524494533409610638224700807216119346720
7 5960244788834646483696848432279085620158276713249664692981627981321135464152584
8 82590187784406915463666993231671009459188410953796224233873542950969577339250027
9 68876520583464697770622321657076833170056511209332449663781837603694136444406281
10 042053396870977465916057756101739472373801429441421111406337458176
11 >>> print(type(b))
12 <type 'long'>
```

Computers represent 32 bit integer numbers by converting them to base 2. The conversion works in the following way:

```

1 def int2binary(n, nbits=32):
2     if n<0:
3         return [1 if bit==0 else 0 for bit in int2binary(-n-1,nbits)]
4     bits = [0]*nbits
5     for i in range(nbits):
6         n, bits[i] = divmod(n,2)
7     if n: raise OverflowError
8     return bits
```

The case  $n < 0$  is called *two's complement* and is defined as the value obtained by subtracting the number from the largest power of two ( $2^{32}$  for 32 bits). Just by looking at the most significative bit one can determine the sign of the binary number (1 for negative and 0 for zero or positive).

### 3.1.2 float and decimal

There are two ways to represent decimal numbers in Python. Using the native double precision (64bits) representation, `float`, or using the `decimal` module.

## 28 WEB AUTOMATION WITH PYTHON

Most numerical problems are dealt with simply using `float`:

```
1 >>> pi = 3.141592653589793
2 >>> two_pi = 2.0 * pi
```

The `cmath` module, described later, contains many math functions that use with the `float` type.

Floating point numbers are internally represented as follows:

$$x = \pm m2^e \quad (3.1)$$

where  $x$  is the number,  $m$  is called mantissa and it is zero or a number in range [1,2),  $e$  is called exponent. The sign,  $m$  and  $e$  can be computed using the following algorithm which also writes their representation in binary:

```
1 def float2binary(x,nm=52,ne=11):
2     if x==0:
3         return 0, [0]*nm, [0]*ne
4     sign,mantissa, exponent = (1 if x<0 else 0),abs(x),0
5     while abs(mantissa)>=2:
6         mantissa,exponent = 0.5*mantissa,exponent+1
7     while 0<abs(mantissa)<1:
8         mantissa,exponent = 2.0*mantissa,exponent-1
9     return sign,int2binary(int(2**nm*(mantissa-1)),nm),int2binary(exponent,ne)
```

Because the mantissa is stored in a fixed number of bits (11 for a 64 bits floating point number) then exponents smaller than  $-1022$  and larger than  $1023$  cannot be represented. An arithmetic operation that returns a number smaller than  $2^{-1022} \simeq 10^{-308}$  cannot be represented and results in an Underflow error. An operation that returns a number larger than  $2^{1023} \simeq 10^{308}$  also cannot be represented and results in an Overflow error.

Here is an example of Overflow:

```
1 >>> a = 10.0**200
2 >>> a*a
3 inf
```

And here is an example of Underflow:

```
1 >>> a = 10.0**-200
2 >>> a*a
3 0.0
```

Another problem with finite precision arithmetic is the loss of precision for

computation. Consider the case of the difference between two numbers with very different orders of magnitude. In order to compute the difference the CPU reduces them to the same exponent (the largest of the two) and then computes the difference in the two mantissas. If two numbers differ for a factor  $2^k$  than the mantissa of the smallest number, in binary, needs to be shifted by  $k$  position thus resulting in a loss of information because the  $k$  least significant bits in the mantissa are ignored. If the difference between the two numbers is greater than a factor  $2^{52}$  all bits in the mantissa of the smallest number are ignored and the smallest number becomes completely invisible.

Below is a practical example that produces a wrong result:

```
1 >>> a = 1.0
2 >>> b = 2.0**53
3 >>> a+b-b
4 0.0
```

Simple example of what occurs internally in a processor to add two floating point numbers together. The IEEE 754 standard states for 32 bit floating point numbers the exponent has a range of -126 to +127.

```
1 262 in IEEE 754: 0 10000111 000001100000000000000000000000 (+ e:8 m:1.0234375)
2     3 in IEEE 754: 0 10000000 100000000000000000000000000000 (+ e:1 m:1.5)
3 265 in IEEE 754: 0 10000111 000010010000000000000000000000
```

To add 262.0 to 3.0 the Exponents must be the same. The exponent of the lesser number is increased to the exponent of the greater number. In this case 3's exponent must be increased by 7. Increasing the exponent by 7 means the mantissa must be shifted 7 binary digits to the right.

```
1 0 100000111 00000011000000000000000000000000
2 0 100000111 00000011000000000000000000000000 (The implied '1' is also pushed 7 places to the right)
3 -----
4 0 100000111 00001001000000000000000000000000 which is the IEEE 754 format for 265.0
```

In the case of two numbers where the exponent is greater than the number of digits in the mantissa the smaller number is shifted right off the end. The effect is a zero is added to the larger number.

In some cases only some of the bits of the smaller number's mantissa are lost an a partial addition occurs.

## 30 WEB AUTOMATION WITH PYTHON

This precision issue is always present but not always obvious. It may consist of a small discrepancy between the true value and the computed value. This difference may increase during the computation, in particular in iterative algorithms, and may be sizable in the result of a complex algorithm.

Python also has a module for decimal floating point arithmetic which allows decimal numbers to be represented exactly. The class Decimal incorporates a notion of significant places (unlike hardware based binary floating point, the decimal module has a user alterable precision):

```
1 >>> from decimal import Decimal, getcontext
2 >>> getcontext().prec = 28 # set precision
3 >>> Decimal(1) / Decimal(7)
4 Decimal('0.1428571428571428571428571429')
```

Decimal numbers can be used almost everywhere in place of floating point number arithmetics but are slower and should be used only where arbitrary precision arithmetics is required. It does not suffer from the overflow, underflow, and precision issues described above.

```
1 >>> from decimal import Decimal
2 >>> a = Decimal(10.0)**300
3 >>> a*a
4 Decimal('1.000000000000000000000000000000E+600')
```

### 3.1.3 str

Python supports the use of two different types of strings: ASCII strings and Unicode strings. ASCII strings are delimited by '...', "...", "'..."', or """...""". Triple quotes delimit multiline strings. Unicode strings start with a `u` followed by the string containing Unicode characters. A Unicode string can be converted into an ASCII string by choosing an encoding (for example UTF8):

```
1 >>> a = 'this is an ASCII string'
2 >>> b = u'This is a Unicode string'
3 >>> a = b.encode('utf8')
```

After executing these three commands, the resulting `a` is an ASCII string storing UTF8 encoded characters.

It is also possible to write variables into strings in various ways:

```

1 >>> print('number is ' + str(3))
2 number is 3
3 >>> print('number is %s' % (3))
4 number is 3
5 >>> print('number is %(number)s' % dict(number=3))
6 number is 3

```

The final notation is more explicit and less error prone, and is to be preferred.

Many Python objects, for example numbers, can be serialized into strings using `str` or `repr`. These two commands are very similar but produce slightly different output. For example:

```

1 >>> for i in [3, 'hello']:
2     ...     print(str(i), repr(i))
3 3 3
4 hello 'hello'

```

For user-defined classes, `str` and `repr` can be defined/redefined using the special operators `__str__` and `__repr__`. These are briefly described later in this chapter. For more information on the topic, refer to the official Python documentation [? ].

Another important characteristic of a Python string is that it is an iterable object, similar to a list:

```

1 >>> for i in 'hello':
2     ...     print(i)
3 h
4 e
5 l
6 l
7 o

```

### 3.1.4 list and array

A list is a native mutable Python object made of a sequence of arbitrary objects. Arrays are not native to Python but unlike lists each element must be of the same type but are computationally faster.

The main methods of a Python lists and arrays are `append`, `insert`, and `delete`. Other useful methods include `count`, `index`, `reverse` and `sort`. Note: arrays do not have a `sort` method.

```

1 >>> b = [1, 2, 3]

```

## 32 WEB AUTOMATION WITH PYTHON

```
2 >>> print(type(b))
3 <type 'list'>
4 >>> b.append(8)
5 >>> b.insert(2, 7) # insert 7 at index 2 (3rd element)
6 >>> del b[0]
7 >>> print(b)
8 [2, 7, 3, 8]
9 >>> print(len(b))
10 4
11 >>> b.append(3)
12 >>> b.reverse()
13 print(b, " 3 appears ", b.count(3), " times.  The number 7 appears at index ", b.
     index(7))
14 [3, 8, 3, 7, 2] 3 appears 2 times.  The number 7 appears at index 3
```

Lists can be sliced:

```
1 >>> a= [2, 7, 3, 8]
2 >>> print(a[:3])
3 [2, 7, 3]
4 >>> print(a[1:])
5 [7, 3, 8]
6 >>> print(a[-2:])
7 [3, 8]
```

and concatenated/joined:

```
1 >>> a = [2, 7, 3, 8]
2 >>> a = [2, 3]
3 >>> b = [5, 6]
4 >>> print(a + b)
5 [2, 3, 5, 6]
```

A list is iterable; you can loop over it:

```
1 >>> a = [1, 2, 3]
2 >>> for i in a:
3 ...     print(i)
4 1
5 2
6 3
```

There is a very common situation for which a *list comprehension* can be used.  
Consider the following code:

```
1 >>> a = [1,2,3,4,5]
2 >>> b = []
3 >>> for x in a:
4 ...     if x % 2 == 0:
5 ...         b.append(x * 3)
6 >>> print(b)
```

```
7 [6, 12]
```

This code clearly processes a list of items, selects and modifies a subset of the input list, and creates a new result list. This code can be entirely replaced with the following list comprehension:

```
1 >>> a = [1,2,3,4,5]
2 >>> b = [x * 3 for x in a if x % 2 == 0]
3 >>> print(b)
4 [6, 12]
```

Python has a module called `array`. It provides an efficient array implementation. Unlike lists, array elements must all be of the same type and the type must be either a char, short, int, long, float or double. A type of char, short, int or long may be either signed or unsigned.

```
1 >>> from array import array
2 >>> a = array('d',[1,2,3,4,5])
3 array('d',[1.0, 2.0, 3.0, 4.0, 5.0])
```

An array object can be used in the same way as a list but its elements must all be of the same type, specified by the first argument of the constructor ("d" for double, "l" for signed long, "f" for float, and "c" for character). For a complete list of available options, refer to the official Python documentation.

Using "array" over "list" can be faster but, more importantly, the "array" storage is more compact for large arrays.

### 3.1.5 tuple

A tuple is similar to a list, but its size and elements are immutable. If a tuple element is an object, the object itself is mutable but the reference to the object is fixed. A tuple is defined by elements separated by a comma and optionally delimited by round brackets:

```
1 >>> a = 1, 2, 3
2 >>> a = (1, 2, 3)
```

The round brackets are required for a tuple of zero elements such as

```
1 >>> a = () # this is an empty tuple
```

A trailing comma is required for a one element tuple but not for two or more elements.

## 34 WEB AUTOMATION WITH PYTHON

```
1 >>> a = (1)    # not a tuple
2 >>> a = (1,)   # this is a tuple of one element
3 >>> b = (1,2)  # this is a tuple of two elements
```

Since lists are mutable, this works:

```
1 >>> a = [1, 2, 3]
2 >>> a[1] = 5
3 >>> print(a)
4 [1, 5, 3]
```

the element assignment does not work for a tuple:

```
1 >>> a = (1, 2, 3)
2 >>> print(a[1])
3 2
4 >>> a[1] = 5
5 Traceback (most recent call last):
6   File "<stdin>", line 1, in <module>
7 TypeError: 'tuple' object does not support item assignment
```

A tuple, like a list, is an iterable object. Notice that a tuple consisting of a single element must include a trailing comma, as shown below:

```
1 >>> a = (1)
2 >>> print(type(a))
3 <type 'int'>
4 >>> a = (1,)
5 >>> print(type(a))
6 <type 'tuple'>
```

Tuples are very useful for efficient packing of objects because of their immutability. The brackets are often optional. You may easily get each element of a tuple by assigning multiple variables to a tuple at one time:

```
1 >>> a = (2, 3, 'hello')
2 >>> (x, y, z) = a
3 >>> print(x)
4 2
5 >>> print(z)
6 hello
7 >>> a = 'alpha', 35, 'sigma' # notice the rounded brackets are optional
8 >>> p, r, q = a
9 >>> print(r)
10 35
```

### 3.1.6 dict

A Python dictionary is a hash table that maps a key object to a value object. For example:

```

1 >>> a = {'k': 'v', 'k2': 3}
2 >>> print(a['k'])
3 v
4 >>> print(a['k2'])
5 3
6 >>> 'k' in a
7 True
8 >>> 'v' in a
9 False

```

You will notice that the format to define a dictionary is the same as JavaScript Object Notation [JSON]. Dictionaries may be nested:

```

1 >>> a = {'x': 3, 'y': 54, 'z': {'a': 1, 'b': 2}}
2 >>> print(a['z'])
3 {'a': 1, 'b': 2}
4 >>> print(a['z']['a'])
5 1

```

Keys can be of any hashable type (int, string, or any object whose class implements the `__hash__` method). Values can be of any type. Different keys and values in the same dictionary do not have to be of the same type. If the keys are alphanumeric characters, a dictionary can also be declared with the alternative syntax:

```

1 >>> a = dict(k='v', h2=3)
2 >>> print(a['k'])
3 v
4 >>> print(a)
5 {'h2': 3, 'k': 'v'}

```

Useful methods are `has_key`, `keys`, `values`, `items` and `update`:

```

1 >>> a = dict(k='v', k2=3)
2 >>> print(a.keys())
3 ['k2', 'k']
4 >>> print(a.values())
5 [3, 'v']
6 >>> a.update({'n1': 'new item'})      # adding a new item
7 >>> a.update(dict(n2='newer item')) # alternate method to add a new item
8 >>> a['n3'] = 'newest item'        # another method to add a new item
9 >>> print(a.items())

```

## 36 WEB AUTOMATION WITH PYTHON

```
10 [(‘k2’, 3), (‘k’, ‘v’), (‘n3’, ‘newest item’), (‘n2’, ‘newer item’), (‘n1’, ‘new
     item’)]
```

The `items` method produces a list of tuples, each containing a key and its associated value.

Dictionary elements and list elements can be deleted with the command `del`:

```
1 >>> a = [1, 2, 3]
2 >>> del a[1]
3 >>> print(a)
4 [1, 3]
5 >>> a = dict(k='v', h2=3)
6 >>> del a['h2']
7 >>> print(a)
8 {'k': 'v'}
```

Internally, Python uses the `hash` operator to convert objects into integers, and uses that integer to determine where to store the value. Using a key that is not hashable will cause an `unhashable type` error

```
1 >>> hash("hello world")
2 -1500746465
3 >>> k = [1,2,3]
4 >>> a = {k:'4'}
5 Traceback (most recent call last):
6   File "<stdin>", line 1, in <module>
7 TypeError: unhashable type: 'list'
```

### 3.1.7 set

A set is something between a list and dictionary. It represents a non-ordered list of unique elements. Elements in a set cannot be repeated. Internally it is implemented as a hash-table, similar to a set of keys in a dictionary. A set is created using the `set` constructor. Its argument can be a list, a tuple, or an iterator:

```
1 >>> s = set([1,2,3,4,5,5,5]) # notice duplicate elements are removed
2 >>> print(s)
3 set([1,2,3,4,5])
4 >>> s = set((1,2,3,4,5))
5 >>> print(s)
6 set([1,2,3,4,5])
7 >>> s = set(i for i in range(1,6))
8 >>> print(s)
9 set([1, 2, 3, 4, 5])
```

Since sets are non-ordered list, appending to the end is not applicable. Instead of `append`, add elements to a set using the `add` method:

```

1 >>> s = set()
2 >>> s.add(2)
3 >>> s.add(3)
4 >>> s.add(2)
5 >>> print(s)
6 set([2, 3])

```

Notice that the same element can not be added twice (2 in the example). There is no exception/error thrown when trying to add the same element more than once.

Since sets are non-ordered, the order you add items is not necessarily the order they will be returned.

```

1 >>> s = set([6, 'b', 'beta', -3.4, 'a', 3, 5.3])
2 >>> print(s)
3 set(['a', 3, 6, 5.3, 'beta', 'b', -3.4])

```

The set object supports normal set operations like union, intersection, and difference:

```

1 >>> a = set([1,2,3])
2 >>> b = set([2,3,4])
3 >>> c = set([2,3])
4 >>> print(a.union(b))
5 set([1, 2, 3, 4])
6 >>> print(a.intersection(b))
7 set([2, 3])
8 >>> print(a.difference(b))
9 set([1])
10 >>> if len(c) == len(a.intersection(c)):
11 ...     print("c is a subset of a")
12 ... else:
13 ...     print("c is not a subset of a")
14 ...
15 c is a subset of a

```

to check for membership:

```

1 >>> 2 in a
2 True

```

## 3.2 Python control flow statements

Python uses indentation to delimit blocks of code. A block starts with a line ending with colon and continues for all lines that have a similar or higher indentation as the next line. For example:

```

1 >>> i = 0
2 >>> while i < 3:
3 ...     print(i)
4 ...     i = i + 1
5 0
6 1
7 2

```

It is common to use four spaces for each level of indentation. It is a good policy not to mix tabs with spaces, which can result in (invisible) confusion.

### 3.2.1 for...in

In Python, you can loop over iterable objects:

```

1 >>> a = [0, 1, 'hello', 'python']
2 >>> for i in a:
3 ...     print(i)
4 0
5 1
6 hello
7 python

```

In the example above you will notice that the loop index, 'i', takes on the values of each element in the list [0, 1, 'hello', 'python'] sequentially. Python range keyword creates a list of integers automatically, that may be used in a 'for' loop without manually creating a long list of numbers.

```

1 >>> a = range(0,5)
2 >>> print(a)
3 [0, 1, 2, 3, 4]
4 >>> for i in a:
5 ...     print(i)
6 0
7 1
8 2
9 3
10 4

```

The parameters for `range(a,b,c)` are first parameter is the starting value of the list. The second parameter is next value if the list contained one more element. The third parameter is the increment value.

`range` can also be called with one parameter. It is matched to "b" above with the first parameter defaulting to 0 and the third to 1.

```

1 >>> print(range(5))
2 [0, 1, 2, 3, 4]
3 >>> print(range(53,57))
4 [53,54,55,56]
5 >>> print(range(102,200,10))
6 [102, 112, 122, 132, 142, 152, 162, 172, 182, 192]
7 >>> print(range(0,-10,-1))
8 [0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

`range` is very convenient for creating a list of numbers, however as the list grows in length, the memory required to store the list also grows. A more efficient option is to use the keyword is `xrange`. It generates an iterable range instead of generating the entire list of elements.

```

1 >>> for i in xrange(0, 4):
2     ...     print(i)
3 0
4 1
5 2
6 3
```

This is equivalent to the C/C++/C#/Java syntax:

```

1 for(int i=0; i<4; i=i+1) { print(i); }
```

Another useful command is `enumerate`, which counts while looping and returns a tuple consisting of (index, value):

```

1 >>> a = [0, 1, 'hello', 'python']
2 >>> for (i, j) in enumerate(a): # the ( ) around i, j are optional
3     ...     print(i, j)
4 0 0
5 1 1
6 2 hello
7 3 python
```

There is also a keyword `range(a, b, c)` that returns a list of integers starting with the value `a`, incrementing by `c`, and ending with the last value smaller than `b`, `a` defaults to 0 and `c` defaults to 1. `xrange` is similar but does not actually generate the list, only an iterator over the list; thus it is better for

## 40 WEB AUTOMATION WITH PYTHON

looping.

You can jump out of a loop using `break`

```
1 >>> for i in [1, 2, 3]:  
2 ...     print(i)  
3 ...     break  
4 1
```

You can jump to the next loop iteration without executing the entire code block with `continue`

```
1 >>> for i in [1, 2, 3]:  
2 ...     print(i)  
3 ...     continue  
4 ...     print('test')  
5 1  
6 2  
7 3
```

Python also supports list comprehensions and you can build lists using using the following syntax:

```
1 >>> a = [i*i for i in [0, 1, 2, 3]:  
2 >>> print(a)  
3 [0, 1, 4, 9]
```

Sometimes you may need a counter to “count” the elements of a list while looping:

```
1 >>> a = [e*(i+1) for (i,e) in ['a','b','c','d']]  
2 >>> print(a)  
3 ['a', 'bb', 'ccc', 'ddddd']
```

### 3.2.2 while

Comparison operators in Python follow the C/C++/Java operators of `==`, `!=`, ... etc. However Python also accepts the `<>` operator as not equal to and is equivalent to `!=`. Logical operators are `and`, `or` and `not`.

The `while` loop in Python works much as it does in many other programming languages, by looping an indefinite number of times and testing a condition before each iteration. If the condition is `False`, the loop ends.

```
1 >>> i = 0  
2 >>> while i < 10:  
3 ...     i = i + 1
```

```

4 >>> print(i)
5 10

```

The `for` loop was introduced earlier in this chapter.

There is no `loop...until` or `do...while` construct in Python.

### 3.2.3 if...elif...else

The use of conditionals in Python is intuitive:

```

1 >>> for i in range(3):
2 ...     if i == 0:
3 ...         print('zero')
4 ...     elif i == 1:
5 ...         print('one')
6 ...     else:
7 ...         print('other')
8 zero
9 one
10 other

```

`elif` means "else if". Both `elif` and `else` clauses are optional. There can be more than one `elif` but only one `else` statement. Complex conditions can be created using the `not`, `and` and `or` logical operators.

```

1 >>> for i in range(3):
2 ...     if i == 0 or (i == 1 and i + 1 == 2):
3 ...         print('0 or 1')

```

### 3.2.4 try...except...else...finally

Python can throw - pardon, raise - Exceptions:

```

1 >>> try:
2 ...     a = 1 / 0
3 ... except Exception, e:
4 ...     print('oops: %s' % e)
5 ... else:
6 ...     print('no problem here')
7 ... finally:
8 ...     print('done')
9 oops: integer division or modulo by zero
10 done

```

## 42 WEB AUTOMATION WITH PYTHON

If an exception is raised, it is caught by the `except` clause and the `else` clause is not executed. The `finally` clause is always executed.

There can be multiple `except` clauses for different possible exceptions:

```
1 >>> try:
2 ...     raise SyntaxError
3 ... except ValueError:
4 ...     print('value error')
5 ... except SyntaxError:
6 ...     print('syntax error')
7 syntax error
```

The `finally` clause is guaranteed to be executed while the `except` and `else` are not. In the example below the function returns within a `try` block. This is bad practice, but it shows that the `finally` will execute regardless of the reason the `try` block is exited.

```
1 >>> def f(x):
2 ...     try:
3 ...         r = x*x
4 ...         return r # bad practice
5 ...     except:
6 ...         print("exception occurred %s" % e)
7 ...     else:
8 ...         print("nothing else to do")
9 ...     finally:
10 ...         print("Finally we get here")
11 ...
12 >>> y = f(3)
13 Finally we get here
14 >>> print "result is ", y
15 result is 9
```

For every `try` you must have either a `except` or `finally` while the `else` is optional

Here is a list of built-in Python exceptions

```
1 BaseException
2   +-+ SystemExit
3   +-+ KeyboardInterrupt
4   +-+ Exception
5     +-+ GeneratorExit
6     +-+ StopIteration
7     +-+ StandardError
8       |     +-+ ArithmeticError
9       |     |     +-+ FloatingPointError
```

```

10      |      +-+ OverflowError
11      |      +-+ ZeroDivisionError
12      +-+ AssertionError
13      +-+ AttributeError
14      +-+ EnvironmentError
15      |      +-+ IOError
16      |      +-+ OSError
17      |      |      +-+ WindowsError (Windows)
18      |      |      +-+ VMSError (VMS)
19      +-+ EOFError
20      +-+ ImportError
21      +-+ LookupError
22      |      +-+ IndexError
23      |      +-+ KeyError
24      +-+ MemoryError
25      +-+ NameError
26      |      +-+ UnboundLocalError
27      +-+ ReferenceError
28      +-+ RuntimeError
29      |      +-+ NotImplementedError
30      +-+ SyntaxError
31      |      +-+ IndentationError
32      |      +-+ TabError
33      +-+ SystemError
34      +-+ TypeError
35      +-+ ValueError
36      |      +-+ UnicodeError
37      |      |      +-+ UnicodeDecodeError
38      |      |      +-+ UnicodeEncodeError
39      |      |      +-+ UnicodeTranslateError
40      +-+ Warning
41      |      +-+ DeprecationWarning
42      |      +-+ PendingDeprecationWarning
43      |      +-+ RuntimeWarning
44      |      +-+ SyntaxWarning
45      |      +-+ UserWarning
46      |      +-+ FutureWarning
47      |      +-+ ImportWarning
48      |      +-+ UnicodeWarning

```

For a detailed description of each of them, refer to the official Python documentation.

Any object can be raised as an exception, but it is good practice to raise objects that extend one of the built-in exception classes.

## 44 WEB AUTOMATION WITH PYTHON

### 3.2.5 def...return

Functions are declared using `def`. Here is a typical Python function:

```
1 >>> def f(a, b):
2 ...     return a + b
3 >>> print(f(4, 2))
4 6
```

There is no need (or way) to specify the type of an argument(s) or the return value(s). In this example, a function `f` is defined that can take two arguments.

Functions are the first code syntax feature described in this chapter to introduce the concept of *scope*, or *namespace*. In the above example, the identifiers `a` and `b` are undefined outside of the scope of function `f`:

```
1 >>> def f(a):
2 ...     return a + 1
3 >>> print(f(1))
4 2
5 >>> print(a)
6 Traceback (most recent call last):
7   File "<pyshell#22>", line 1, in <module>
8     print(a)
9 NameError: name 'a' is not defined
```

Identifiers defined outside of function scope are accessible within the function; observe how the identifier `a` is handled in the following code:

```
1 >>> a = 1
2 >>> def f(b):
3 ...     return a + b
4 >>> print(f(1))
5 2
6 >>> a = 2
7 >>> print(f(1) # new value of a is used)
8 3
9 >>> a = 1 # reset a
10 >>> def g(b):
11 ...     a = 2 # creates a new local a
12 ...     return a + b
13 >>> print(g(2))
14 4
15 >>> print(a # global a is unchanged)
16 1
```

If `a` is modified, subsequent function calls will use the new value of the global `a` because the function definition binds the storage location of the identifier

a, not the value of a itself at the time of function declaration; however, if a is assigned-to inside function g, the global a is unaffected because the new local a hides the global value. The external-scope reference can be used in the creation of *closures*:

```

1 >>> def f(x):
2 ...     def g(y):
3 ...         return x * y
4 ...     return g
5 >>> doubler = f(2) # doubler is a new function
6 >>> tripler = f(3) # tripler is a new function
7 >>> quadrupler = f(4) # quadrupler is a new function
8 >>> print(doubler(5))
9 10
10 >>> print(tripler(5))
11 15
12 >>> print(quadrupler(5))
13 20

```

Function f creates new functions; and note that the scope of the name g is entirely internal to f. Closures are extremely powerful.

Function arguments can have default values and can return multiple results as a tuple: (Notice the parentheses are optional and are omitted in the example.)

```

1 >>> def f(a, b=2):
2 ...     return a + b, a - b
3 >>> x, y = f(5)
4 >>> print(x)
5 7
6 >>> print(y)
7 3

```

Function arguments can be passed explicitly by name, therefore the order of arguments specified in the caller can be different than the order of arguments with which the function was defined:

```

1 >>> def f(a, b=2):
2 ...     return a + b, a - b
3 >>> x, y = f(b=5, a=2)
4 >>> print(x)
5 7
6 >>> print(y)
7 -3

```

## 46 WEB AUTOMATION WITH PYTHON

Functions can also take a runtime-variable number of arguments. Parameters that start with \* and \*\* must be the last two parameters. If the \*\* parameter is used it must be last in the list. Extra values passed in will be placed in the \*identifier parameter while named values will be placed into the ft \*\*identifier. Notice that when passing values into the function the unnamed values must be before any and all named values.

```
1 >>> def f(a, b, *extra, **extraNamed):
2 ...     print "a = ", a
3 ...     print "b = ", b
4 ...     print "extra = ", extra
5 ...     print "extranamed = ", extraNamed
6 >>> f(1, 2, 5, 6, x=3, y=2, z=6)
7 a = 1
8 b = 2
9 extra = (5, 6)
10 extranamed = {'y': 2, 'x': 3, 'z': 6}
```

Here the first two parameters (1 and 2) are matched with the parameters a and b while the tuple 5, 6 is placed into extra and the remaining items (which are in a dictionary format) are placed into extraNamed

In the opposite case, a list or tuple can be passed to a function that requires individual positional arguments by unpacking them:

```
1 >>> def f(a, b):
2 ...     return a + b
3 >>> c = (1, 2)
4 >>> print(f(*c))
5 3
```

and a dictionary can be unpacked to deliver keyword arguments:

```
1 >>> def f(a, b):
2 ...     return a + b
3 >>> c = {'a':1, 'b':2}
4 >>> print(f(**c))
5 3
```

### 3.2.6 lambda

lambda provides a way to define a short unnamed function:

```
1 >>> a = lambda b: b + 2
2 >>> print(a(3))
3 5
```

The expression "`lambda [a]:[b]`" literally reads as "a function with arguments `[a]` that returns `[b]`". The `lambda` expression is itself unnamed, but the function acquires a name by being assigned to identifier `a`. The scoping rules for `def` apply to `lambda` equally, and in fact the code above, with respect to `a`, is identical to the function declaration using `def`:

```

1 >>> def a(b):
2 ...     return b + 2
3 >>> print(a(3))
4 5

```

The only benefit of `lambda` is brevity; however, brevity can be very convenient in certain situations. Consider a function called `map` that applies a function to all items in a list, creating a new list:

```

1 >>> a = [1, 7, 2, 5, 4, 8]
2 >>> map(lambda x: x + 2, a)
3 [3, 9, 4, 7, 6, 10]

```

This code would have doubled in size had `def` been used instead of `lambda`. The main drawback of `lambda` is that (in the Python implementation) the syntax allows only for a single expression; however, for longer functions, `def` can be used and the extra cost of providing a function name decreases as the length of the function grows. Just like `def`, `lambda` can be used to *curry* functions: new functions can be created by wrapping existing functions such that the new function carries a different set of arguments:

```

1 >>> def f(a, b): return a + b
2 >>> g = lambda a: f(a, 3)
3 >>> g(2)
4 5

```

Python functions, created with either `def` or `lambda` allow re-factoring of existing functions in terms of a different set of arguments.

### 3.3 Classes

Because Python is dynamically typed, Python classes and objects may seem odd. In fact, member variables (attributes) do not need to be specifically defined when declaring a class and different instances of the same class can have different attributes. Attributes are generally associated with the

## 48 WEB AUTOMATION WITH PYTHON

instance, not the class (except when declared as "class attributes", which is the same as "static member variables" in C++/Java).

Here is an example:

```
1 >>> class MyClass(object): pass
2 >>> myinstance = MyClass()
3 >>> myinstance.myvariable = 3
4 >>> print(myinstance.myvariable)
5 3
```

Notice that `pass` is a do-nothing command. In this case it is used to define a class `MyClass` that contains nothing. `MyClass()` calls the constructor of the class (in this case the default constructor) and returns an object, an instance of the class. The `(object)` in the class definition indicates that our class extends the built-in `object` class. This is not required, but it is good practice.

Here is a more involved class with multiple methods:

```
1 >>> class Complex(object):
2 ...     z = 2
3 ...     def __init__(self, real=0.0, imag=0.0):
4 ...         self.real, self.imag = real, imag
5 ...     def magnitude(self):
6 ...         return (self.real**2 + self.imag**2)**0.5
7 ...     def __add__(self,other):
8 ...         return Complex(self.real+other.real, self.imag+other.imag)
9 >>> a = Complex(1,3)
10 >>> b = Complex(2,1)
11 >>> c = a + b
12 >>> print(c.magnitude())
13 5
```

Functions declared inside the class are methods. Some methods have special reserved names. For example, `__init__` is the constructor. In the example we created a class to store the `real` and the `imag` part of a complex number. The constructor takes these two variables and stores them into `self` (not a keyword but a variable that plays the same role as `this` in Java and `(*this)` in C++. (This syntax is necessary to avoid ambiguity when declaring nested classes, such as a class that is local to a method inside another class, something the Python allows but Java and C++ do not).

The `self` variable is defined by the first argument of each method. They all must have it but they can use another variable name. Even if we use another

name, the first argument of a method always refers to the object calling the method. It plays the same role as the `this` keyword in the Java and the C++ languages.

`__add__` is also a special method (all special methods start and end in double underscore) and it overloads the `+` operator between `self` and `other`. In the example, `a+b` is equivalent to a call to `a.__add__(b)` and the `__add__` method receives `self=a` and `other=b`.

All variables are local variables of the method except variables declared outside methods which are called *class variables*, equivalent to C++ *static member variables* that hold the same value for all instances of the class.

### 3.3.1 Special methods and operator overloading

Class attributes, methods, and operators starting with a double underscore are usually intended to be private (*for example* to be used internally but not exposed outside the class) although this is a convention that is not enforced by the interpreter.

Some of them are reserved keywords and have a special meaning.

For example:

- `__len__`
- `__getitem__`
- `__setitem__`

They can be used, for example, to create a container object that acts like a list:

```

1 >>> class MyList(object):
2 >>>     def __init__(self, *a): self.a = list(a)
3 >>>     def __len__(self): return len(self.a)
4 >>>     def __getitem__(self, key): return self.a[key]
5 >>>     def __setitem__(self, key, value): self.a[key] = value
6 >>> b = MyList(3, 4, 5)
7 >>> print(b[1])
8 4
9 >>> b.a[1] = 7
10 >>> print(b.a)
11 [3, 7, 5]
```

## 50 WEB AUTOMATION WITH PYTHON

Other special operators include `__getattr__` and `__setattr__`, which define the get and set methods (getters and setters) for the class, and `__add__`, `__sub__`, `__mul__`, `__div__` which overload arithmetic operators. For the use of these operators we refer the reader to the chapter on linear algebra where they will be used to implement algebra for matrices.

### 3.4 File input/output

In Python you can open and write in a file with:

```
1 >>> file = open('myfile.txt', 'w')
2 >>> file.write('hello world')
3 >>> file.close()
```

Similarly, you can read back from the file with:

```
1 >>> file = open('myfile.txt', 'r')
2 >>> print(file.read())
3 hello world
```

Alternatively, you can read in binary mode with "rb", write in binary mode with "wb", and open the file in append mode "a", using standard C notation.

The `read` command takes an optional argument, which is the number of bytes. You can also jump to any location in a file using `seek`.

You can read back from the file with `read`

```
1 >>> print(file.seek(6))
2 >>> print(file.read())
3 world
```

and you can close the file with:

```
1 >>> file.close()
```

In the standard distribution of Python, which is known as CPython, variables are reference-counted, including those holding file handles, so CPython knows that when the reference count of an open file handle decreases to zero, the file may be closed and the variable disposed. However, in other implementations of Python such as PyPy, garbage collection is used instead of reference counting, and this means that it is possible that there may accumulate too many open file handles at one time, resulting in an error before the `gc` has a chance to close and dispose of them all. Therefore it is

best to explicitly close file handles when they are no longer needed.

### 3.5 import modules

The real power of Python is in its library modules. They provide a large and consistent set of Application Programming Interfaces (APIs) to many system libraries (often in a way independent of the operating system).

For example, if you need to use a random number generator, you can do:

```
1 >>> import random
2 >>> print(random.randint(0, 9))
3 5
```

This prints a random integer in the range of (0,9], 5 in the example. The function `randint` is defined in the module `random`. It is also possible to import an object from a module into the current namespace:

```
1 >>> from random import randint
2 >>> print(randint(0, 9))
```

or import all objects from a module into the current namespace:

```
1 >>> from random import *
2 >>> print(randint(0, 9))
```

or import everything in a newly defined namespace:

```
1 >>> import random as myrand
2 >>> print(myrand.randint(0, 9))
```

In the rest of this book, we will mainly use objects defined in modules `math`, `cmath`, `os`, `sys`, `datetime`, `time` and `cPickle`. We will also use the `random` module but we will describe it in a later chapter.

In the following subsections we consider those modules that are most useful.

#### 3.5.1 math and cmath

Here is a sampling of some of the methods available in the `math` and `cmath` packages.

## 52 WEB AUTOMATION WITH PYTHON

- `math.isinf(x)` returns true if the floating point number `x` is positive or negative infinity
- `math.isnan(x)` returns true if the floating point number `x` is NaN. See Python documentation or IEEE 754 standards for more information.
- `math.exp(x)` returns `e**x`
- `math.log(x[, base])` returns the logarithm of `x` to the optional `base`. If `base` is not supplied `e` is assumed.
- `math.cos(x),math.sin(x),math.tan(x)` returns the cos, sin, tan of the value of `x`. `x` is in radians.
- `math.pi, math.e` the constants for `pi` and `e` to available precision

### 3.5.2 os

This module provides an interface to the operating system API. For example:

```
1 >>> import os  
2 >>> os.chdir('..')  
3 >>> os.unlink('filename_to_be_deleted')
```

Some of the `os` functions, such as `chdir`, are not thread-safe, *for example* they should not be used in a multi-threaded environment.

`os.path.join` is very useful; it allows the concatenation of paths in an OS-independent way:

```
1 >>> import os  
2 >>> a = os.path.join('path', 'sub_path')  
3 >>> print(a)  
4 path/sub_path
```

System environment variables can be accessed via:

```
1 >>> print(os.environ)
```

which is a read-only dictionary.

### 3.5.3 sys

The `sys` module contains many variables and functions, but the used the most is `sys.path`. It contains a list of paths where Python searches for modules. When we try to import a module, Python searches the folders listed in `sys.path`. If you install additional modules in some location and want Python to find them, you need to append the path to that location to `sys.path`.

```

1 >>> import sys
2 >>> sys.path.append('path/to/my/modules')

```

### 3.5.4 datetime

The use of the `datetime` module is best illustrated by some examples:

```

1 >>> import datetime
2 >>> print(datetime.datetime.today())
3 2008-07-04 14:03:90
4 >>> print(datetime.date.today())
5 2008-07-04

```

Occasionally you may need to time-stamp data based on the UTC time as opposed to local time. In this case you can use the following function:

```

1 >>> import datetime
2 >>> print(datetime.datetime.utcnow())
3 2008-07-04 14:03:90

```

The `datetime` module contains various classes: `date`, `datetime`, `time` and `timedelta`. The difference between two date or two datetime or two time objects is a `timedelta`:

```

1 >>> a = datetime.datetime(2008, 1, 1, 20, 30)
2 >>> b = datetime.datetime(2008, 1, 2, 20, 30)
3 >>> c = b - a
4 >>> print(c.days)
5 1

```

We can also parse dates and datetimes from strings for example:

```

1 >>> s = '2011-12-31'
2 >>> a = datetime.datetime.strptime(s, '%Y-%m-%d') #modified
3 >>> print(s.year, s.day, s.month)
4 2011 31 12 #modified

```

## 54 WEB AUTOMATION WITH PYTHON

Notice that “%Y” matches the 4-digits year, “%m” matches the month as a number (1-12), “%d” matches the day (1-31), “%H” matches the hour, “%M” matches the minute, and “%S” matches the second. Check the Python documentation for more options.

### 3.5.5 time

The time module differs from date and datetime because it represents time as seconds from the epoch (beginning of 1970).

```
1 >>> import time  
2 >>> t = time.time()  
3 1215138737.571
```

Refer to the Python documentation for conversion functions between time in seconds and time as a datetime.

### 3.5.6 urllib and json

The urllib is a module to download data or a web page from a URL.

```
1 >>> page = urllib.urlopen('http://www.google.com/')  
2 >>> html = page.read()
```

## 3.6 Regular Expressions

The most efficient way of searching for text/patterns in text is by using regular expressions. For example:

```

1 import urllib
2 import re
3 html = urllib.urlopen('http://www.depaul.com')
4 regex = re.compile('[\w\.\.]+@[\\w\.\.]+\')
5 emails = regex.findall(html)
6 print emails

```

`re.compile(...)` compiles the description of a pattern into an object, `regex`. The `.findall` method of the object finds all expressions matching the compiled pattern. Here are useful rules:

- `^` means “beginning of text”.
- `$` means “end of text of text”.
- `.` means any character.
- `\w` means any alphanumeric character or underscore.
- `\W` means any character not alphanumeric character and not underscore.
- `\s` means any form of whitespace (space, tab, etc.).
- `\S` means any char but not a whitespace.
- `[...]` can be used to build an OR sequence of characters.
- `(...)` can be used to build a AND sequence of consecutive matching characters.
- `(...)*` means any repetition of the sequence (...).
- `(...)+` means any non-zero repetition of the sequence (...).

A slash can be used to escape special characters. In a string `\` must be escaped as `\\\`. An actual “`\`” must therefore be escaped twice `\\\\\`.

Here are some examples of regular expressions (they are all approximations and oversimplifications):

- Match all email in a document

## 56 WEB AUTOMATION WITH PYTHON

```
1 re.compile("[\w\.]+@[ \w\.]+").findall(html)
```

- Match all tags in HTML

```
1 re.compile("\<(\w+)").findall(html)
```

- Match all images sources in HTML

```
1 re.compile("""\<img [^>]*src=[\'\'](.*\')[\'\']""").findall(html.lower())
```

- Match all links in HTML

```
1 re.compile("""\<a [^>]*href=[\'\'](.*\')[\'\']""").findall(html.lower())
```

# 4

## OS funtions from Python

Get current working directory with os.getcwd()

```
1 print os.getcwd()
```

Get the status of a file with os.stat()

```
1 print "Getting the status of: ", os.stat('/usr/bin/python')
```

Execute a shell command with os.system()

```
1 os.system('ls -l')
```

Return the current process id with os.getpid()

```
1 print os.getpid()
```

Change the owner and group id of path to the numeric uid and gid with os.chown()

```
1 os.chown(path, uid, gid)
```

Processes in the system run queue averaged over the last 1, 5, and 15 minutes

```
1 print os.getloadavg()
```

Check if a path exists with os.path.exists()

```
1 if os.path.exists("file.txt"):
```

Create a new directory named 'new\_directory' if it doesn't exist already"

```
1 os.path.exists("new_directory") or os.mkdir("new_directory")
```

Check if the path is a directory or a file with os.path.isdir() and os.path.isfile()

```
1 path = "/tmp"
```

## 58 WEB AUTOMATION WITH PYTHON

```
2 if os.path.isdir(path): print "That's a directory"
3 if os.path.isfile(path): print "That's a file"
```

Create a directory with os.mkdir()

```
1 print os.mkdir('new_directory', 0666)
```

Recursive create directories with os.makedirs()

```
1 os.makedirs('dir_a/dir_b/dir_c')
```

Remove a directory with os.rmdir()

```
1 print os.rmdir('directory')
```

Recursively remove empty directories with os.removedirs()

```
1 os.removedirs('dir_a/dir_b/dir_c')
```

Rename a file with os.rename()

```
1 print os.rename('/path/to/old/file', '/path/to/new/file')
```

Rename a file with shutil.move()

```
1 print shutil.move('/path/to/old/file', '/path/to/new/file')
```

Rename a file with shutil.copy()

```
1 print shutil.copy('/path/to/old/file', '/path/to/new/file')
```

Get the users home directory

```
1 print os.path.expanduser('~')
```

Check read permissions with os.access()

```
1 path = '/tmp/file.txt'
2 print os.access(path, os.R_OK)
```

Get the users environment with os.environ()

```
1 home = os.environ['HOME']
2 print home
```

Move focus to a different directory with os.chdir()

```
1 print os.chdir('/tmp')
```

Print out all directories, sub-directories and files with os.walk()

```
1 for root, dirs, files in os.walk("/tmp"):
2     print root
3     print dirs
4     print files
```

Get the last time a directory was accessed with os.path.getatime()

```
1 os.path.getatime('/tmp')
```

Get the last time a directory was modified with os.path.getmtime()

```
1 os.path.getmtime('/tmp')
```

Get the user ID with os.getuid()

```
1 if os.getuid() != 0: print "you are not root"
```

Get the group ID with os.getgid()

```
1 print os.getgid()
```

Return the name of the user logged in with os.getlogin()

```
1 print os.getlogin()
```

Returns a list of all files in a directory with os.listdir()

```
1 for filename in os.listdir("/tmp"):
2     print "This is inside /tmp", filename
```

Get the size of a file with os.path.getsize()

```
1 os.path.getsize("/tmp/file.txt")
```

Using Python in your daily work is a good way to automate system administration tasks, when you feel that your shell scripts are too limited.



# 5

## BeautifulSoup

Regular expressions are useful and indispensable, yet building the proper expressions to match generic tags, and tag attributes in HTML can be difficult. A better solution is to use a library designed specifically for this. One such library is BeautifulSoup. BeautifulSoup does not completely eliminate the need for regular expressions (in fact it uses regular expressions and you may have to build some to pass to some BeautifulSoup methods).

You can install BeautifulSoup from the Bash shell with

```
1 > sudo easy_install BeautifulSoup
```

From within Python you only need to import one class BeautifulSoup which we shall rename Soup for brevity:

```
1 >>> from BeautifulSoup import BeautifulSoup as Soup
```

Given any help text, for example:

```
1 >>> html = "<html><p>Para 1<p>Para 2<blockquote>Quote 1<blockquote>Quote 2"
```

You can use the Soup object to parse it and construct an object representation of the Document Object Model:

```
1 >>> soup = Soup(html)
```

this object can be serialized back into HTML (although it would not produce the exact same HTML, but an equivalent one):

```
1 >>> print soup.prettify()
2 <html>
```

## 62 WEB AUTOMATION WITH PYTHON

```
3  <p>
4      Para 1
5  </p>
6  <p>
7      Para 2
8      <blockquote>
9          Quote 1
10         <blockquote>
11             Quote 2
12         </blockquote>
13     </blockquote>
14 </p>
15 </html>
```

The `soup` object (an instance of `BeautifulSoup` or `BeautifulStoneSoup`) is a deeply-nested, well-connected data structure that corresponds to the structure of an XML or HTML document. The parser object contains two other types of objects: Tag objects, which correspond to tags like the `<TITLE>` tag and the `<B>` tags; and `NavigableString` objects, which correspond to strings like “Page title” and “This is paragraph”.

The `soup` object can be used to query the DOM and manipulate it. For example, you can ask for the first `<p/>` tag which has an “align” attribute equal to “center”:

```
1 >>> p = soup.find('p', align='center')
```

The output `p` object is also a `BeautifulSoup` object. Unless you copy it, any change to this object will also affect the `soup` object that contains it.

Given then `p` object you can retrieve its attributes, for example its `id`, its `class`, its contents, or its serialized content:

```
1 >>> print p['id']
2 >>> print p['class']
3 >>> print p.contents
4 >>> print p.string
```

Similarly you can search all the tags in the `soup` object which match a given criteria, for example app `<p/>` tags:

```
1 >>> soup.findAll('p')
```

or all tags with an `id` that matches a regular expression:

```
1 >>> soup.findAll(id=re.compile("^\w+"))
```

An alternative syntax for getting all the `<p/>` tags is:

```
1 >>> tags = soup.p
```

The '.' notation can be chained to obtain, for example the title inside the head of the document:

```
1 >>> print soup.head.title
```

You can also replace its contents with:

```
1 >>> soup.head.title.replaceWith('New Title')
```

Each BeautifulSoup object is a reference to a portion of the soup document. Any change will affect the original document.

Notice that any HTML document is a tree, therefore you can traverse the tree by looking for the parent of a given node, its siblings (next and previous), as well as the next and previous tags in the DOM structure:

```
1 >>> print soup.head.parent.name
2 >>> print soup.p.nextSibling
3 >>> print soup.p.previousSibling
4 >>> print soup.p.next
5 >>> print soup.p.previous
```

## 5.1 Cached Downloads

```
1 import urllib2
2 import shelve
3
4 def cached_download(url):
5     d = shelve.open('cached.pages')
6     if url in d:
7         data = d[url]
8     else:
9         try:
10             data = d[url] = urllib2.urlopen(url,timeout=5).read()
11         except:
12             data = None
13 return data
```

## 5.2 A simple web crawler

```
1 # author: Massimo Di Pierro
2 # license: BSD
3 from BeautifulSoup import BeautifulSoup
4 import collections
```

## 64 WEB AUTOMATION WITH PYTHON

```
5 import cPickle
6 import urllib2
7 import shelve
8 import socket
9 import re
10
11 TIMEOUT = 10 #seconds
12 EXTENSIONS = ('asp','aspx','php','html','htm')
13 regex_remove = re.compile(
14     '|<script.*?|</script>||<style.*?|</style>||!--.*?--|>',re.DOTALL)
15 regex_words = re.compile('[a-zA-Z\-\_][a-zA-Z\-\_]+')
16 regex_url = re.compile('(?P<scheme>\w+)://(?P<host>.*?)(?P<path_info>/.*)')
17
18 def cached_download(url):
19     d = shelve.open('cached.pages')
20     try:
21         data = d[url]
22     except KeyError:
23         try:
24             data = d[url] = urllib2.urlopen(url,timeout=TIMEOUT).read()
25         except (socket.gaierror, socket.error, AttributeError, IOError):
26             return None
27     return data
28
29 def parse_words(html):
30     body = html.find('body')
31     if not body: return set()
32     texts = body.findAll(text=True)
33     texts = map(lambda e:e.string.strip().lower(),texts)
34     texts = filter(lambda e:e,texts)
35     sets = [set(regex_words.findall(t)) for t in texts]
36     words = reduce(lambda a,b:a|b,sets,set())
37     return words
38
39 def normalize_url(new, scheme, host, path_info):
40     new = new.encode('utf8')
41     extension = new.split('?',1)[0].split('#',1)[0].rsplit('.')[-1].lower()
42     if (new.startswith('mailto:') or new.startswith('#') or
43         (not '/' in extension and not extension in EXTENSIONS) or
44         (':///' in new and not new.split('://')[0] in ('http','https'))):
45         return None
46     elif '://' in new:
47         return new
48     elif new.startswith('/'):
49         return scheme+':'+new
50     elif new.startswith('//'):
51         return '%s://%s%s' % (scheme, host, new)
52     else:
53         relative = path_info.rsplit('/',1)[0]
```

```

54     while new.startswith('../'):
55         relative, new = relative.rsplit('/',1)[0], new[3:]
56     return '%s:///%s%s/%s' % (scheme, host, relative, new)
57
58 def find_links(html, url = 'http://cdm.depaul.edu/', pattern=re.compile('.*')):
59     match = regex_url.match(url)
60     (scheme, host, path_info) = match.groups()
61     start = '/'.join(url.split('/')[3:])
62     links = html.findAll('a')
63     links = map(lambda link:link.get('href'),links)
64     links = filter(lambda e:e,links)
65     links = map(lambda link:normalize_url(link,scheme,host,path_info),links)
66     follow_links = set(filter(lambda e:e and pattern.match(e),links))
67     ignore_links = set(filter(lambda e:e and e not in follow_links,links))
68     return follow_links, ignore_links
69
70 def crawl(urls = ['http://web2py.com/'], pattern = None,
71           max_links=200, filename='maps.pickle', mode='ignore-check'):
72     if isinstance(urls,str): urls = [urls]
73     maps = collections.defaultdict(set)
74     discovered = set(urls)
75     ignored = set()
76     queue = [url for url in urls]
77     n = 0
78     if pattern is None:
79         pattern = '^|'.join(re.escape(url) for url in urls)
80     if isinstance(pattern,str):
81         pattern = re.compile(pattern)
82     while queue and (max_links is None or n<max_links):
83         url = queue.pop()
84         print '%s: RETRIEVING %s' % (n, url)
85         data = cached_download(url)
86         if data is None:
87             print '%s: INVALID %s' % (n, url)
88             continue
89         html = BeautifulSoup(data)
90         follow_links, ignore_links = find_links(html,url,pattern)
91         words = parse_words(html)
92         for word in words:
93             maps[word].add(url)
94         if mode == 'ignore-log':
95             for url in ignore_links - ignored:
96                 print 'IGNORING %s' % url
97                 ignored |= ignore_links
98         elif mode == 'ignore-check':
99             for url in ignore_links - ignored:
100                 if cached_download(url) is None:
101                     print '%s: IGNORING BROKEN %s' % (n, url)
102                 else:

```

## 66 WEB AUTOMATION WITH PYTHON

```
103         print '%s: IGNORING %s' % (n, url)
104     ignored |= ignore_links
105     for url in follow_links:
106         if url and not url in discovered:
107             queue.append(url)
108             discovered.add(url)
109     print '%s: PROGRESS #urls:%s #words:%s' % (n,len(discovered),len(maps))
110     n += 1
111     cPickle.dump(maps,open(filename,'w'))
112
113 def search(words, filename='maps.pickle'):
114     words = filter(lambda w:len(w)>2,words.lower().split())
115     maps = cPickle.load(open(filename))
116     return reduce(lambda a,b:a&b,[maps[word] for word in words])
117
118 if __name__=='__main__':
119     print crawl('http://depaul.edu/',pattern='^http://[\w\.]*depaul\.edu(.*',
120                 max_links=None)
121     print search('Python programming')
```

# 6

## Mechanize

You can install Mechanize from the Bash shell with

```
1 > sudo easy_install mechanize
```

Mechanize simulates a web browser and its interaction with the web. You can start it with:

```
1 >>> import mechanize  
2 >>> br = mechanize.Browser()
```

It is important to configure your browser so that it can fool third party web servers. Web servers set restriction and who can access web pages and in particular they do not like to be visited by robots. You have to pretend your browser is a “normal” one, for example Firefox:

```
1 >>> br.set_handle_robots( False )  
2 >>> br.addheaders = [('User-agent', 'Firefox')]
```

Once you have a browser `br` you can give instructions to it, for example, “visit the Google web page”:

```
1 >>> response = br.open("http://www.google.com/")
```

The response has attributes for example:

```
1 >>> print response.geturl()  
2 http://www.google.com/
```

or the returned status code:

```
1 >>> print response.code  
2 200
```

## 68 WEB AUTOMATION WITH PYTHON

or the response headers

```
1 >>> print response.info().headers
2 ['Date: Wed, 11 Sep 2013 18:09:10 GMT\r\n', 'Expires: -1\r\n', 'Cache-Control:
   private, max-age=0\r\n', 'Content-Type: text/html; charset=ISO-8859-1\r\n', '
   Set-Cookie: PREF=ID=a53b86f876c89fc9:U=548b6d330fff7040:FF=0:TM=1378922843:LM
   =1378922950:S=xPNYhTPc_HwsBqd8; expires=Fri, 11-Sep-2015 18:09:10 GMT; path=/;
   domain=.google.com\r\n', 'Server: gws\r\n', 'X-XSS-Protection: 1; mode=block\r
   \r\n', 'X-Frame-Options: SAMEORIGIN\r\n', 'Alternate-Protocol: 80:quic\r\n', '
   Connection: close\r\n']
```

Or a specific header:

```
1 >>> response.info().getheader('date')
2 'Wed, 11 Sep 2013 18:09:10 GMT'
```

Or the page content (HTML or binary depending on the data):

```
1 >>> print response.read()
```

What is most important is the ability of Mechanize to understand the page. You can ask it for example to follow a link and get another page. For example, follow a link that contains text matching a regular expressions:

```
1 >>> response = br.follow_link(text_regex=".*Google.*")
```

You can ask if the action was successful and the browser received HTML:

```
1 >>> assert br.viewing_html()
```

Because of Mechanize ability to understand HTML, you can ask it to locate all links in a page:

```
1 >>> for f in br.links(): print f
2 ...
3 Link(base_url='http://www.google.com/', url='http://www.google.com/intl/en/options/
   ', text='More &#9660;', tag='a', attrs=[('class', 'gb3'), ('href', 'http://www.
   .google.com/intl/en/options/'), ('onclick', 'this.blur();gbar.tg(event);return
   !1'), ('aria-haspopup', 'true')])
4 ...
```

And/or all forms in the page:

```
1 >>> for f in br.forms(): print f
2 <f GET http://www.google.ca/search application/x-www-form-urlencoded
3   <HiddenControl(ie=ISO-8859-1) (readonly)>
4   <HiddenControl(hl=en) (readonly)>
5   <HiddenControl(source=hp) (readonly)>
6   <TextControl(q=>
7     <SubmitControl(btnG=Google Search) (readonly)>
8     <SubmitControl(btnI=I'm Feeling Lucky) (readonly)>
9     <HiddenControl(gbv=1) (readonly)>>
```

This says: There is only one form on the page/ Hidden field id's are ie (page encoding), hl (language code), hp (? don't know), and gbv (also don't know). The only not-hidden field id is q, which is a text input, which is the search text.

And you can ask it to select a form, fill it, and submit it:

```
1 >>> formnames = [f.name for f in br.forms()]
2 >>> br.select_form(formnames[0])
```

or

```
1 >>> br.select_form(nr=0)
2 >>> br.form['q'] = 'foo'
3 >>> br.submit()
4 >>> response = br.response()
```



# 7

## Fabric

Fabric is a remote management tool written in Python.

You can install Fabric from the Bash shell with

```
1 > sudo easy_install fabric
```

To use Fabric you create a fabfile (just a Python program) which defines the operations you want to perform for example:

```
1 from fabric.api import local  
2  
3 def dir():  
4     local('ls -l')
```

than you run it with:

```
1 fab -f /path/to/fabfile.py dir
```

This calls the `uptime` function inside the "fabfile.py". From now on we will assume the "fabfile.py" in the user root folder and will omit the `-f` command line option.

The `local(...)` function executes the shell command passed as argument. In this case the `ls -l` command. The command in the example is executed on the local machine but the point of Fabric is to execute the command remotely and simultaneously on one or more machines. Here is a better example

```
1 from fabric.api import env, run  
2  
3 env.hosts = [ '192.168.1.100', '192.168.1.101', '192.168.1.102' ]
```

## 72 WEB AUTOMATION WITH PYTHON

```
4 env.user = "you"
5 env.password = "mysecret"
6 env.parallel = True
7
8 def dir():
9     run('ls -l')
```

Now a call to

```
1 fab dir
```

will produce a directory listing from all the machines at the hosts defined in `env.hosts`. One can also specify a host using the `-H` command line option:

```
1 fab -H 192.168.1.100 dir
```

You can use `-h` to find out more about command line options:

```
1 Usage: fab [options] <command>[:arg1,arg2=val2,host=foo,hosts='h1;h2',...] ...
2
3 Options:
4   -h, --help           show this help message and exit
5   -d NAME, --display=NAME
6                   print detailed info about command NAME
7   -F FORMAT, --list-format=FORMAT
8                   formats --list, choices: short, normal, nested
9   -l, --list            print list of possible commands and exit
10  --set=KEY=VALUE,...  comma separated KEY=VALUE pairs to set Fab env vars
11  --shortlist          alias for -F short --list
12  -V, --version         show program's version number and exit
13  -a, --no_agent        don't use the running SSH agent
14  -A, --forward-agent   forward local agent to remote end
15  --abort-on-prompts   abort instead of prompting (for password, host, etc)
16  -c PATH, --config=PATH
17                   specify location of config file to use
18  -D, --disable-known-hosts
19                   do not load user known_hosts file
20  -f PATH, --fabfile=PATH
21                   python module file to import, e.g. '../other.py'
22  --hide=LEVELS         comma-separated list of output levels to hide
23  -H HOSTS, --hosts=HOSTS
24                   comma-separated list of hosts to operate on
25  -i PATH              path to SSH private key file. May be repeated.
26  -k, --no-keys         don't load private key files from ~/.ssh/
27  --keepalive=N         enables a keepalive every N seconds
28  --linewise            print line-by-line instead of byte-by-byte
29  -n M, --connection-attempts=M
30                   make M attempts to connect before giving up
31  --no-pty              do not use pseudo-terminal in run/sudo
32  -p PASSWORD, --password=PASSWORD
```

```

33          password for use with authentication and/or sudo
34 -P, --parallel      default to parallel execution method
35 --port=PORT        SSH connection port
36 -r, --reject-unknown-hosts
37                      reject unknown hosts
38 -R ROLES, --roles=ROLES
39                      comma-separated list of roles to operate on
40 -S SHELL, --shell=SHELL
41                      specify a new shell, defaults to '/bin/bash -l -c'
42 --show=LEVELS       comma-separated list of output levels to show
43 --skip-bad-hosts   skip over hosts that can't be reached
44 --ssh-config-path=PATH
45                      Path to SSH config file
46 -t N, --timeout=N  set connection timeout to N seconds
47 -u USER, --user=USER username to use when connecting to remote hosts
48 -w, --warn-only     warn, instead of abort, when commands fail
49 -x HOSTS, --exclude-hosts=HOSTS
50                      comma-separated list of hosts to exclude
51 -z INT, --pool-size=INT
52                      number of concurrent processes to use in parallel mode

```

You can also declare and call functions which take arguments. For example here is a fabfile that allows you to apt-get install packages remotely:

```

1 from fabric.api import env, sudo
2
3 env.hosts = [ '192.168.1.100', '192.168.1.101', '192.168.1.102' ]
4 env.user = "you"
5 env.password = "mysecret"
6 env.parallel = True
7
8 def install(package):
9     sudo('apt-get -y install %s' % package)

```

and you can run it with:

```
1 fab install:python
```

A nice feature of Fabric is the ability to transfer files between the local and the remote host. Here is an example to copy a local file and unzip it at each at the remote destinations:

```

1 from fabric.api import env, sudo
2 from fabric.operations import put
3
4 env.hosts = [ '192.168.1.100', '192.168.1.101', '192.168.1.102' ]
5 env.user = 'you'
6 env.password = 'ysecret'
7 env.parallel = True

```

## 74 WEB AUTOMATION WITH PYTHON

```
8
9 def deploy(filename):
10    put(filename, '/home/you/destination/path')
11    run('unzip /home/you/destination/path/%s' % filename)
```

which you can use with

```
1 > fab deploy:myfile.zip
```

If you need to change folders:

```
1 from __future__ import with_statement
2 from fabric.api import run
3
4 dir = '/path/to/mydir'
5
6 with cd(dir):
7     run('ls -l')
```

Fabric can handle errors:

```
1 from __future__ import with_statement
2 from fabric.api import local, settings, abort
3 from fabric.contrib.console import confirm
4
5 def test():
6     with settings(warn_only=True):
7         result = local('./manage.py test my_app', capture=True)
8     if result.failed and not confirm(``Tests failed. Continue anyway?``):
9         abort(``Aborting at user request.'')
```

# 8

## User Management

To create a new user:

```
1 >>> import os
2 >>> import crypt
3 >>> password = 'testpassword'
4 >>> username = 'someuser'
5 >>> encrypted_password = crypt.crypt(password, '22')
6 >>> os.system('useradd -m -p %s %s' % (encrypted_password, username))
```

(the `-m` option creates the home user if does not exist).

To give superuser permissions to the new user use shell command `visudo` to edit `/etc/sudoers` file:

```
1 > export VISUAL=nano; visudo
2 > visudo
3 # users network=(group) commands
4
5 # joe mike and jude can run any command (ALL) from (ALL) terminals
6 joe, mike, jude ALL=ALL
7
8 # tim and ales can run any command from any terminal as members of the www-data
   group
9 tim, alex ALL=(www-data) ALL
10
11 # jack can run any command from any group from any machine in
   10.1.2.0/255.255.255.0
12 jack 10.1.2.0/255.255.255.0=(ALL) ALL
13
14 # cathy can print from any machine
15 cathy ALL=/usr/sbin/lpc, /usr/bin/lprm
```

## 76 WEB AUTOMATION WITH PYTHON

```
16  
17 # john can do everything and is not even prompted a password!  
18 john ALL=(ALL) NOPASSWD: ALL
```

To automate remote tasks we want to be able to login into a remote host without being prompted a password. We can achieve this by creating a certificate for the client machine. On the client machine:

```
1 > mkdir ~/.ssh  
2 > chmod 700 ~/.ssh  
3 > ssh-keygen -t rsa -b 4096
```

it will produce the following and as some questions, you can just press enter:

```
1 Generating public/private rsa key pair.  
2 Enter file in which to save the key (/home/username/.ssh/id_rsa):  
3 Enter passphrase (empty for no passphrase):  
4 Enter same passphrase again:  
5 Your identification has been saved in /home/username/.ssh/id_rsa.  
6 Your public key has been saved in /home/username/.ssh/id_rsa.pub.
```

Then use the following command

```
1 > ssh-copy-id <username>@<host>
```

which will automatically ssh into <username>@<host> and appends the local ~/.ssh/id\_rsa.pub to the remote ~/.ssh/authorized\_keys.

Troubleshooting. You may need to set permission on the remote host:

```
1 > chmod go-w ~/  
2 > chmod 700 ~/.ssh  
3 > chmod 600 ~/.ssh/authorized_keys
```

You may need to tell remote host to add the key:

```
1 > ssh-add
```

Now you can:

```
1 > ssh <username>@<host>
```

Always on the remote host, makes sure the file /etc/ssh/sshd\_config contains:

```
1 PubkeyAuthentication yes  
2 RSAAuthentication yes
```

You may need to restart the deamon. On the host:

```
1 sudo /etc/init.d/sshd restart
```

# 9

## EC2 and OpenStack

A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major classifications, based on their use and degree of correspondence to any real machine:

A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS). These usually emulate an existing architecture, and are built with the purpose of either providing a platform to run programs where the real hardware is not available for use (for example, executing software on otherwise obsolete platforms), or of having multiple instances of virtual machines leading to more efficient use of computing resources, both in terms of energy consumption and cost effectiveness (known as hardware virtualization, the key to a cloud computing environment), or both.

A process virtual machine (also, language virtual machine) is designed to run a single program, which means that it supports a single process. Such virtual machines are usually closely suited to one or more programming languages and built with the purpose of providing program portability and flexibility (amongst other things). An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual environment.

## 78 WEB AUTOMATION WITH PYTHON

Multiple VMs each running their own operating system (called guest operating system) are frequently used in server consolidation, where different services that used to run on individual machines to avoid interference are instead run in separate VMs on the same physical machine.

Software to run virtual machines include VMWare, VirtualBox, and KVM, and XEN.

There are many web services that allows customers to rent computers on which to run their own computer applications. The biggest is probably Amazon EC2.

### 9.1 EC2 Command Line API Tools

Display all available ec2 images

```
ec2-describe-images -x all
```

Generate a keypair which will be instead of a root password

```
ec2-create-keypair mb-keypair
```

Start running a EC2 Instance, this is when billing starts

```
ec2-run-instances ami-a9fe1bc0 -k mbkeypair
```

Open up the port 22 for ssh, similary 80 for port 80

```
ec2-authorize default -p 22
```

Display list of ec2 instances running

```
ec2-describe-instances
```

Terminate an ec2 running instances. This is where billing ends

```
ec2-terminate-instances ami-a9fe1bc0
```

OpenStack is a free and open software for creating a cloud infrasture like amazon EC2. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators

control while empowering its users to provision resources through a web interface.

Nova is a controller for cloud computing platforms, supports OpenStack and EC2.

```
>>> from novaclient import OpenStack
>>> nova = OpenStack(USERNAME, PASSWORD, AUTH_URL)
>>> nova.servers.list()
[<Server: buildslave-ubuntu-9.10>

>>> nova.flavors.list()
[<Flavor: 256 server>,
 <Flavor: 512 server>,
 <Flavor: 1GB server>,
 <Flavor: 2GB server>,
 <Flavor: 4GB server>,
 <Flavor: 8GB server>,
 <Flavor: 15.5GB server>]
>>> fl = nova.flavors.find(ram=512)
>>> nova.servers.create("my-server", flavor=fl)
<Server: my-server>
```

For more info: <https://github.com/openstack/python-novaclient>

Boto is another python library to manage virtualization. While Nova is designed for OpenStack and supports EC2 via a compatibility layer, Boto is designed for EC2 and supports OpenStack via a compatibility layer.

Both Nova and Boto can run from command line:

```
> nova boot myUbuntuServer --image "3afe97b2-26dc-49c5-a2cc-a2fc8d80c001" --
```



# Bibliography

- [1] <http://www.python.org>
- [2] <http://farmdev.com/talks/unicode/>
- [3] <http://www.crummy.com/software/BeautifulSoup/>
- [4] <http://wwwsearch.sourceforge.net/mechanize/>
- [5] <http://docs.fabfile.org/en/1.8/>
- [6] <https://help.ubuntu.com/community/SSH/OpenSSH/Keys>
- [7] <http://www.garron.me/en/linux/visudo-command-sudoers-file-sudo-default.html>



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

DePaul University

CTI: School of Computer Science,  
Telecommunications and Information Technology

## CSC 309: Object Oriented Programming in C++

Massimo Di Pierro



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Textbooks and Compiler

Course title:  
**Object Oriented Programming in C++**

Instructor:  
**Prof. Massimo Di Pierro**  
(PhD in High Energy Theoretical Physics from Univ. of Southampton, UK)

Textbook:  
**Applications Programming in C++**  
Johnsonbaugh and Kalin, Prentice Hall

Optional reference book:  
**C++ in Plain English, 3/E**  
Overland, John Wiley & Sons

Suggested compiler and IDE:  
**Bloodshed Dev-C++ (mingw gcc) version 4**  
download from [www.bloodshed.net/devcpp.html](http://www.bloodshed.net/devcpp.html)

Course web page:  
<http://www.cs.depaul.edu/courses/syllabus.asp>



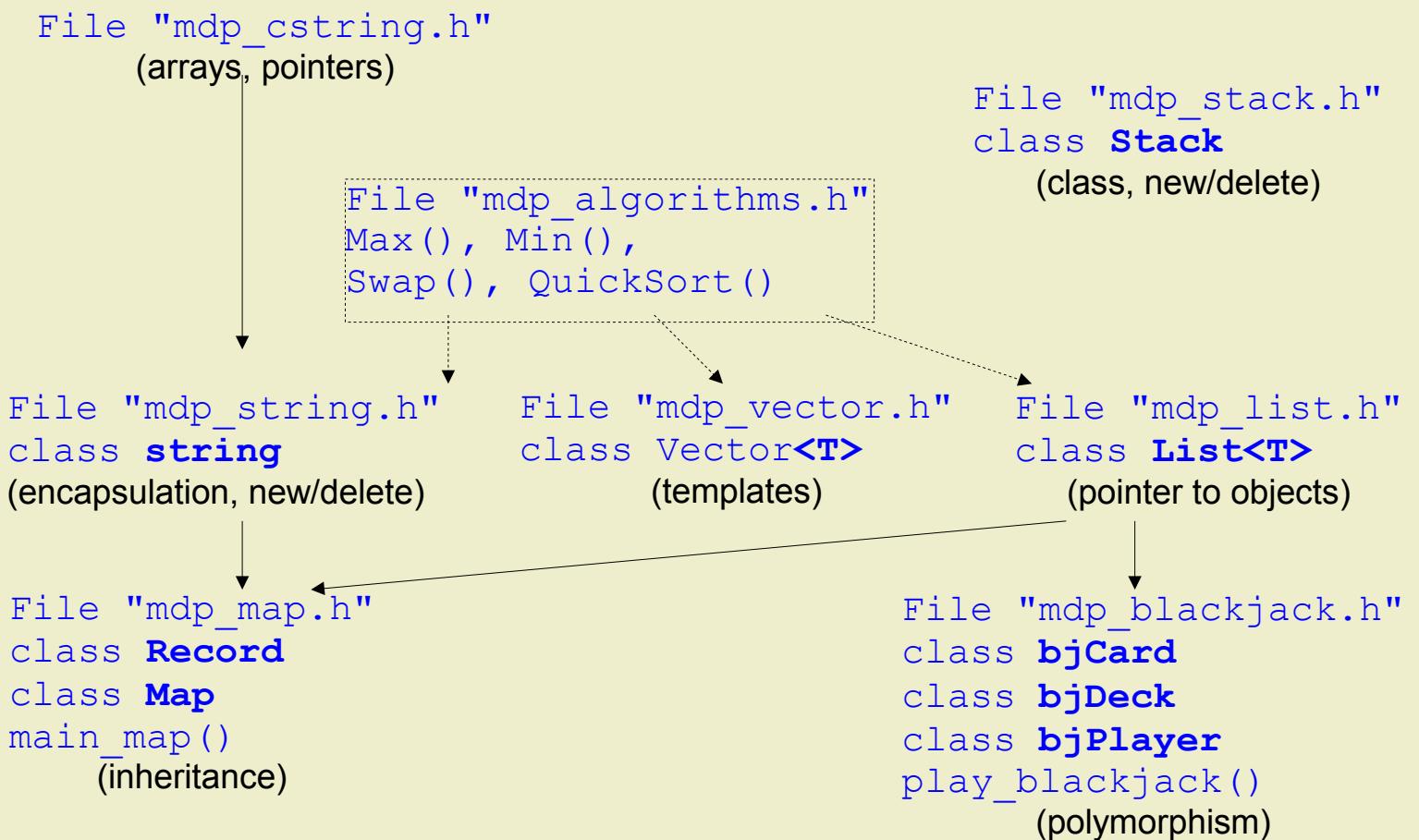
- Week 1: Introduction to C++ programming
- Week 2: Pointers, arrays and dynamic allocation
- Week 3: Encapsulation: array of characters vs class string
- Week 4: more on Classes and Objects (class Stack)
- Week 5: Classes, Objects and Templates (class Vector, List)
- Week 6: **Midterm**
- Week 7: Inheritance (class Map)
- Week 8: Interfaces and Polymorphism
- Week 9: File Input/Output with streams
- Week 10: Overview of the Standard Template Libraries



- Week 1: Introduction to C++ programming  
(p. 1.\* , 2.1-2.10)
- Week 2: Pointers, arrays and dynamic allocation  
(p. 3.\* , 4.1-4.3, 4.6, 4.9, 2.12)
- Week 3: Encapsulation: array of characters vs class string  
(p. 4.5, 4.7, 5.1, 5.2, 5.5, 8.3, 8.5, 9.5, cstring, string)
- Week 4: more on Classes and Objects  
(p. 5.7, 5.9, 8.\* , class Stack)
- Week 5: Classes, Objects and Templates  
(p. 8.\* , 10.1-10.3, class Vector, class List)
- Week 7: Inheritance  
(p. 6.\* , class Map)
- Week 8: Interfaces and Polymorphism  
(p. 7.\* , play Blackjack)
- Week 9: File Input/Output with streams  
(p. 2.5, 2.13, class stream, class fstream)
- Week 10: Overview of the Standard Template Libraries



## Course hierarchy





CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 1

## Introduction to C++ programming



- 1960: Many computer languages were invented (including **Algol**)
- 1970: Ken Thompson invented the **B** language (from Algol)  
Norwegian Air Force invented **Simula** and the concept of **class**
- 1971: Dennis Ritchie (Bell Labs) invented the **C** as an extension of the B language.  
(90% of Unix was written in C)
- 1983: Bjarne Stroustrup invented "C with classes". This later became known as **C++**. BS worked at the Computing Laboratory in Cambridge and Bell Labs (now AT&T + Lucent)



## Features

portable  
hardware dependent code  
explicit memory management  
automatic garbage collection  
polymorphism  
classes  
inheritance  
multiple inheritance  
interfaces  
templates  
operator overloading  
standardized graphic library

	<u>C</u>	<u>C++</u>	<u>Java</u>
portable	y~	y~	y
hardware dependent code	y	y	n
explicit memory management	y	y	n
automatic garbage collection	n~	n~	y
polymorphism	n	y	y
classes	n	y	y
inheritance	n	y	y
multiple inheritance	n	y	n
interfaces	n	y	y
templates	n	y	n
operator overloading	n	y	n
standardized graphic library	n~	n~	y

It is not possible to write an operating system in Java !!!  
C++ programs are 2-10 times faster than Java programs !!!



**Tip:** Insert scaffolding code at the top of any program after  
`#include <iostream>`

File "mdp\_scaffolding.h"

```
Class __scaffolding__class  {
public:
    ~__scaffolding__class() {
        cout << "press ENTER to continue...\\n";
        cin.ignore(256, '\\n');
        cin.get();
    }
} __scaffolding__;
```



Output shell

[program output]  
press ENTER to continue...



## Typical C++ file structure



### File "myprg.cpp"

```
#include "mylib.h"
int main() {
    myfunc(3);
    return 0;
}
```

### File "mylib.h"

```
void myfunc(int);
```

### File "mylib.cpp"

```
#include "mylib.h"

void myfunc(int i) {
    cout << i << endl;
}
```

### bash shell (cygwin)

```
> ls
mylib.h
mylib.cpp
myprg.cpp

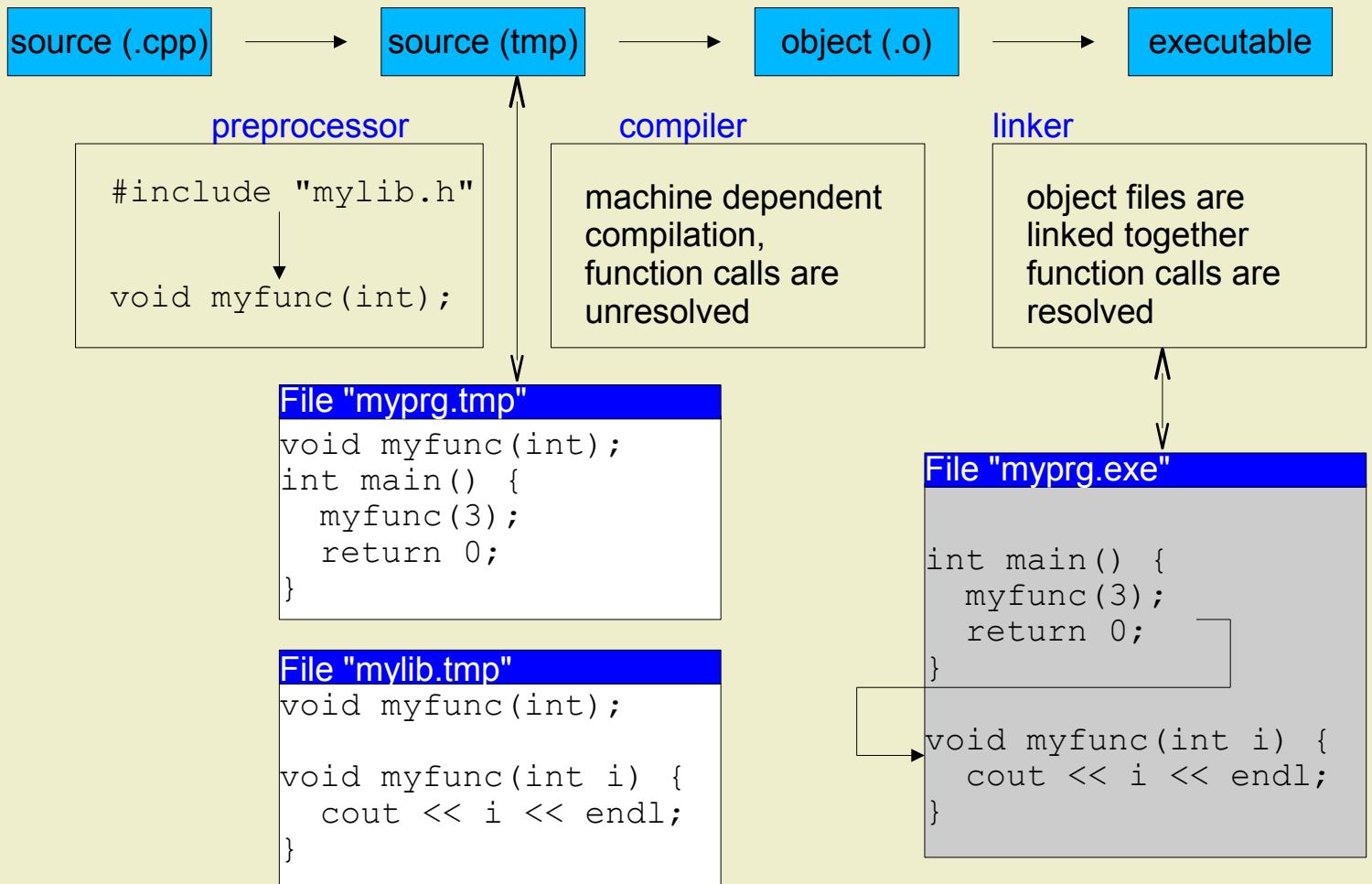
> g++ -ansi -c mylib.cpp -o mylib.o
> ls *.o
mylib.o

> g++ -ansi -c myprg.cpp -o myprg.o
> ls *.o
myprg.o
mylib.o

> g++ myprg.o mylib.o -o myprg.exe
> ./myprg.exe
3
```



## Compilation steps





## Function Main

### Java Program

```
import java.io.*;  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello World");  
    }  
}
```

### Program "hello world\_01.cpp"

```
#include <iostream>  
  
int main(int arg, char** args) {  
    cout << "Hello World\n";  
    return 0;  
}
```

In C++ as in Java statements end with semicolon and not with the newline.



#include is a preprocessor directive rather than a statement.

Preprocessor directives (start with #) are not followed by semicolon

All C++ statements, except preprocessor directives and closed } brackets end with semicolon.

Exception: closed } bracket terminating a class declaration must be followed by semicolon.

```
Program "hello_world_01.cpp"
#include <iostream>

class myclass { int i; };

int main(int arg, char** args) {
    cout << "Hello World\n";
    return 0;
}
```



In C and C++ comments can be bounded by `/* */` and can be multi line. The use of this kind of comment is discouraged since they cannot be nested.

In C++ (not in C) single line comment are indicated with `//`

Program "hello\_world\_01.cpp"

```
#include <iostream>

/* this is typical
   old style C comment */

// This is a typical
// C++ comment

int main(int arg, char** args) {
    cout << "Hello World\n"; // this is another comment
    return 0;
}
```



## Function Main

### Program "hello\_world\_02.cpp"

```
#include <iostream>
int main(int argc, char** argv) {
    cout << "Hello World\n";
    return 0;
}
```

### Program "hello\_world\_03.cpp"

```
#include <iostream>
int main() {
    cout << "Hello World\n";
    return 0;
}
```

### Program "hello\_world\_04.cpp" (deprecated)

```
#include <iostream>
void main() {
    cout << "Hello World\n";
}
```

### output shell

```
Hello World
press ENTER to continue...
```



Program "cin\_01.cpp"

```
#include <iostream>
void main() {
    int i;
    cout << "Type a number\n";
    cin >> i;
    cout << "You typed " << i << endl;
}
```

output shell

```
Type a number
123
You typed 123
press ENTER to continue...
```

Program "file\_io\_01.cpp"

```
#include <iostream>
#include <fstream>
void main() {
    int i;
    ifstream ifile;
    ifile.open("source.dat");
    ifile >> i;
    ifile.close();
    ofstream ofile;
    ofile.open("destination.dat");
    ofile << "i=" << i << endl;
    ofile.close();
}
```



## Types

### Program "print\_2.cpp"

```
#include <iostream>
void main(int argc, char**argv) {
    int i=2;
    cout << "i=" << i << endl;
}
```

### Program "print\_2.3.cpp"

```
#include <iostream>
void main(int argc, char**argv) {
    float i=2.3;
    cout << "i=" << i << endl;
}
```

### output shell

```
i=2
press ENTER to continue...
```

### output shell

```
i=2.3
press ENTER to continue...
```

bool	1bit (?)	true or false
char	8 bits	-128 to 127 or 0 to 255
unsigned char	8 bits	0 to 255
short	16 bits	-32768 to 32767
unsigned short	16 bits	0 to 65535
int	32 bits	same as long
unsigned int	32 bits	unisgned short or unsigned long
long	32 bits	-(~2M) to (~2M)
unsigned long	32 bits	0 to (~4M)
float	32 bits	up to +/- 3.4e+38
double	64 bits	up to +/- 1.8e+308



In C++ assignments ( $i=j$ ) are expressions (i.e. return a value)

Program "assignments\_01.cpp"

```
#include <iostream>

void main(int argc, char**argv) {
    int i,j,k;

    i=9*(j=k=1);

    cout << i << j << k << endl;
}
```

output shell

```
911
press ENTER to continue...
```

cout << a << b; // prints a  
cout << b; // prints b

i=9\*(j=k=1); → k=1  
↓  
i=9\*(j=1); → j=1;  
↓  
i=9\*(1);  
↓  
i=9; → i=9;



## for

### Program "for\_01.cpp"

```
#include <iostream>
void main() {
    int i;
    for(i=0; i<5; i++)
        cout << i << endl;
    cout << "and i=" << i << endl;
}
```

### output shell

```
0
1
2
3
4
and i=5
press ENTER to continue...
```

### Program "for\_02.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(; i<5 ;) {
        cout << i << endl;
        i++;
    }
    cout << "and i=" << i << endl;
}
```

### Program "for\_03.cpp"

```
#include <iostream>
void main() {
    for(int i=0; i<5; i++)
        cout << i << endl;
}
```

Careful



## while

### Program "while\_01.cpp"

```
#include <iostream>
void main() {
    int i=0;
    while(i<5) {
        cout << i << endl;
        i++;
    }
    cout << "and i=" << i << endl;
}
```

### output shell

```
0
1
2
3
4
and i=5
press ENTER to continue...
```

### Program "for\_02.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(; i<5 ;) {
        cout << i << endl;
        i++;
    }
    cout << "and i=" << i << endl;
}
```

**while (expression) { ... }**

equivalent to

**for (; expression; ) { ... }**



## break

### Program "while\_02.cpp"

```
#include <iostream>
void main() {
    int i=0;
    while(true) {
        cout << i << endl;
        if(++i==5) break;
    }
    cout << "and i=" << i << endl;
}
```

### output shell

```
0
1
2
3
4
and i=5
press ENTER to continue...
```

### Program "for\_04.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(i=0; true ;) {
        cout << i << endl;
        if(++i==5) break;
    }
    cout << "and i=" << i << endl;
}
```

```
i=4; (i++)==4;
i=4; (++i)==5;
i=4; (i--)==4;
i=4; (--i)==3;
```



## if ... else ...

### Program "if\_01.cpp"

```
#include <iostream>
void main() {
    int i=0;
    if(i==0) cout << "true\n";
}
```

### output shell

```
true  
press ENTER to continue...
```

### Program "if\_02.cpp"

```
#include <iostream>
void main() {
    int i=1;
    if(i==0)
        cout << "true\n";
    else
        cout << "false\n";
}
```

### output shell

```
false  
press ENTER to continue...
```

### Program "if\_03.cpp"

```
#include <iostream>
void main() {
    int i=1;
    cout <<
        ((i==0)?"true\n":"false\n");
}
```

discouraged

Logical operators  
(same as Java)

!	not
&&	and
	or
==	equal
!=	not equal

Logical values

0	false
1,2,..	true



## switch and break

### Program "switch\_01.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(i=0; i<5; i++) {
        switch(i) {
            case 0:
                cout << ".\n";
                break;
            case 1:
                cout << "..\n";
                break;
            case 2:
                cout << "... \n";
                break;
            default:
                cout << "default\n";
        }
    }
}
```

### output shell

```
.
..
...
default
default
press ENTER to continue...
```

Note: in this example **break** breaks the switch statement and not the for loop.

Therefore **break** is usually required!



## switch without break

Program "switch\_02.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(i=0; i<5; i++) {
        switch(i) {
            case 0:
                cout << ".\n";
            case 1:
                cout << "..\n";
            case 2:
                cout << "....\n";
            default:
                cout << "default\n";
        }
    }
}
```

output shell

```
.
..
...
default
..
...
default
...
default
default
default
press ENTER to continue...
```



## CSC 309 – OOP in C++

Prof. Massimo Di Pierro

Program "while\_switch.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(i=0;true;i++) {
        switch(i) {
            case 5: break;
        }
        cout << i << endl;
    }
    cout << "end i=" << i << endl;
}
```

infinite loop

goto

discouraged

Program "goto\_01.cpp"

```
#include <iostream>
void main() {
    int i=0;
    for(i=0;true; i++) {
        switch(i) {
            case 5: goto end_loops;
        }
        cout << i << endl;
    }
end_loops:
    cout << "end i=" << i << endl;
}
```

The use of `goto` is discouraged since it is inelegant and never necessary.

To exit nested loops use

`try ... catch ...`

instead...

output shell

```
0
1
2
3
4
end i=5
press ENTER to continue...
```



## try ... catch ... (exceptions)

### Program "try\_01.cpp"

```
#include <iostream>

void main() {
    int i=0;
    try {
        for(i=0; true; i++) {
            switch(i) {
                case 5: throw 0;
            }
            cout << i << endl;
        }
    } catch(int j) {
        cout << "end i=" << i << endl;
    }
}
```

### output shell

```
0
1
2
3
4
end i=5
press ENTER to continue...
```

```
char* hello="Hello";
try {
    switch(...) {
        case ...: throw 0;
        case ...: throw 1;
        case ...: throw 2;
        case ...: throw hello;
    }
} catch(int j) {
    cout << "j=" << j << endl;
} catch(char* s) {
    cout << "s=" << s << endl;
}
```



Program "try\_02.cpp"

```
#include <iostream>
#include "mdp_exception.h"

void main() {
    int i=0;
    try {
        throw Exception("Whatever");
    } catch (Exception e) {
        cout << e.value() << endl;
    }
}
```

output shell

```
Whatever.
```



## Function call

### Program "global\_01.cpp"

```
#include <iostream>

int square(int i) {
    cout << "square called with i=" << i << endl;
    return i*i;
}

void print(int i) {
    cout << "print called with i=" << i << endl;
}

void main() {
    print(square(7));
}
```

► In this example **square** and **print** are global functions (they do not belong to any class and are visible to any other function within the scope (in this case the file))

### output shell

```
square called with i=7
print called with i=49
press ENTER to continue...
```



### Program "global\_02.cpp"

```
#include <iostream>

int i;

void square() {
    cout << "square called with i=" << i << endl;
    i=i*i;
}

void print() {
    cout << "print called with i=" << i << endl;
}

void main() {
    i=7;
    square();
    print();
    cout << "here i=" << i << endl;
}
```

discouraged

► In this example **i** is a global variable (it does not belong to any class or function and is visible to any function within the scope (in this case the file))

### output shell

```
square called with i=7
print called with i=49
here i=49
press ENTER to continue...
```



Program "by\_value.cpp"

```
#include <iostream>

void swap(int a, int b) {
    int c;
    c=a; a=b; b=c;
}

void main() {
    int i=3, j=4;
    swap(i, j);
    cout << "i=" << i << ", ";
    cout << "j=" << j << endl;
}
```

wrong

output shell

```
i=3, j=4
press ENTER to continue...
```

```
memory:  0|0|3|4|0|0|0|0|3|4|?|0|0
variable:   i  j          a  b  c
```

Passing by value or by reference

Program "by\_value.cpp"

```
#include <iostream>

void swap(int& a, int& b) {
    int c;
    c=a; a=b; b=c;
}

void main() {
    int i=3, j=4;
    swap(i, j);
    cout << "i=" << i << ", ";
    cout << "j=" << j << endl;
}
```



output shell

```
i=4, j=3
press ENTER to continue...
```

```
memory:  0|0|3|4|0|0|0|0|0|0|?|0|0
variable:   i  j          a      c
                           b
```



## Reference variables

### Program "by\_value.cpp"

```
#include <iostream>

void main() {
    int i=3;
    int j=i;
    j=4;
    cout << "i=" << i << ", ";
    cout << "j=" << j << endl;
}
```



### output shell

```
i=3, j=4
press ENTER to continue...
```

```
memory: 0|0|3|4|0|0|0|0|0|
variable:   i j
```

### Program "by\_value.cpp"

```
#include <iostream>

void main() {
    int i=3;
    int& j=i;
    j=4;
    cout << "i=" << i << ", ";
    cout << "j=" << j << endl;
}
```



### output shell

```
i=4, j=4
press ENTER to continue...
```

```
memory: 0|0|3|0|0|0|0|0|0|
variable:   i
                      j
```



Program "static\_01.cpp"

```
#include <iostream>

int j=0;

void increment(int i) {
    cout << "j was " << j;
    j=j+i;
    cout << ", j is " << j << endl;
}

void main() {
    increment(2);
    increment(3);
}
```

discouraged

Static variables

Program "static\_02.cpp"

```
#include <iostream>

void increment(int i) {
    static int j=0;
    cout << "j was " << j;
    j=j+i;
    cout << ", j is " << j << endl;
}

void main() {
    increment(2);
    increment(3);
}
```

output shell

```
j was 0, j is 2
j was 2, j is 5
press ENTER to continue...
```

output shell

```
j was 0, j is 2
j was 2, j is 5
press ENTER to continue...
```



### Program "static\_02.cpp"

```
#include <iostream>

void increment(int i) {
    static int j=0;
    cout << "j was " << j;
    j=j+i;
    cout << ", j is " << j << endl;
}

void main() {
    increment(2);
    increment(3);
}
```



### output shell

```
j was 0, j is 2
j was 2, j is 5
press ENTER to continue...
```

### Returning by reference

### Program "static\_03.cpp"

```
#include <iostream>

int& increment(int i) {
    static int j=0;
    cout << "j was " << j;
    j=j+i;
    cout << ", j is " << j << endl;
    return j;
}

void main() {
    increment(2)=10;
    increment(3);
}
```



### output shell

```
j was 0, j is 2
j was 10, j is 13
press ENTER to continue...
```



<u>C style</u>	<u>C++ style</u>	<u>functions</u>
stdio.h	cstdio	printf, scanf, gets, puts, fopen, fclose, fgets, fputf, fwrite, fread, feof, ftell, fseek, ...
string.h	cstring	<b>strlen, strcpy, strcmp, strcat, ...</b>
stdlib.h	cstdlib	atof, atoi, atol, exit, abort
math.h	cmath	pow, exp, log, sin, cos, ...
complex.h	ccomplex	( <i>complex numbers</i> )
time.h	ctime	time, clock
assert.h	cassert	assert
ctype.h	cctype	toupper, tolower
signal.h	csygnal	signal
stdarg.h		( <i>functions with variable args</i> )
iostream.h	iostream	( <i>stream IO functions</i> )
	<b>string</b>	( <i>Java like string class</i> )
	(STL)	( <i>standard template library</i> )



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 2

## Pointers, Arrays and Dynamic Allocation



Program "memory\_01.cpp"

```
#include <iostream>

void main() {
    int i=-111;
    float a=3.14;
    cout << i << endl;
    cout << a << endl;
    cout << &i << endl;
    cout << &a << endl;
}
```

&i means address of i

output shell

```
-111
3.14
254fdd0
254fdd4
press ENTER to continue...
```

(int) -111 in binary is **fffffff91**  
(float) 3.14 in binary is **4048f5c3**

virtual address	memory content	allocated variables
...	00	(?)
254fdcfc	00	(?)
<b>254fdd0</b>	<b>ff</b>	<b>i</b>
254fdd1	ff	"
254fdd2	ff	"
254fdd3	91	"
<b>254fdd4</b>	<b>40</b>	<b>a</b>
254fdd5	48	"
254fdd6	f5	"
254fdd7	c3	"
254fdd8	00	(?)
...	00	(?)



Program "memory\_01.cpp"

```
#include <iostream>

void main() {
    int i=-111;
    float a=3.14;
    cout << i << endl;
    cout << a << endl;
    cout << &i << endl;
    cout << &a << endl;
}
```

&i means address of i

output shell

```
-111
3.14
254fdd0
254fdd4
press ENTER to continue...
```

Ignoring binary representation...

virtual address	memory content	allocated variables
...	?	
254fdcfc	?	
<b>254fdd0</b>	-111	<b>i</b>
<b>254fdd4</b>	3.14	<b>a</b>
254fdd8	?	
...	?	

or equivalent representation

```
address: ... f 0      4      8 ..
memory:   ?|?|-111|3.14|?|?
variable:          i       a
```



## Declaration of pointers

### Program "memory\_02.cpp"

```
#include <iostream>

void main() {
    int* p;           // p is declared as a pointer to integer
    int i=-111;
    p=&i;            // p = address of i

    cout << i << endl;
    cout << sizeof(i) << endl;
    cout << p << endl;
    cout << p+1 << endl;
    cout << p+2 << endl;
}
```

### output shell

```
-111
4
254fdd0
254fdd4
254fdd8
press ENTER to continue...
```

**int\*** is type *pointer to integer*

**p** is declared as a *pointer to integer*

**p = address of i**

virtual address	memory content	allocated variables
...	?	
254fda8	254fdd0	<b>p</b>
...	...	
254fdd0	-111	<b>i</b>
254fdd4	?	
254fdd8	?	
...	?	



## Arithmetic of pointers

### Program "memory\_02.cpp"

```
#include <iostream>

void main() {
    char* p; // p is declared as a pointer to char
    char i='c';
    p=&i; // p = address of i

    cout << i << endl;
    cout << sizeof(i) << endl;
    cout << p << endl;
    cout << p+1 << endl;
    cout << p+2 << endl;
}
```

### output shell

```
c
1
254fdd0
254fdd1
254fdd2
press ENTER to continue...
```

**char\*** is type *pointer to integer*

**p** is declared as a *pointer to char*

**p = address of i**

virtual address	memory content	allocated variables
...	?	
254fda8	254fdd0	<b>p</b>
...	...	
<b>254fdd0</b>	c	i
254fdd1	?	
254fdd2	?	
...	?	



## Uses of &

The symbol & can be used in four ways:

- 1) Passing a variable by reference (in the declaration of the arguments of a function)
- 2) Getting the address of a variable
- 3) Declaring a variable by reference (i.e. a new name for an existing variable)
- 4) Returning by reference

```
int& func() {  
    static int n;  
    return n;  
}
```

advanced

### Program "memory\_03.cpp"

```
#include <iostream>  
  
void print_address_of(int& k)  
    cout << &k << endl;  
}  
  
void main() {  
    int i=5;  
    int& j=i;  
    print_address_of(i);  
    print_address_of(j);  
}
```

### output shell

```
254fdfa  
254fdfa  
press ENTER to continue...
```



## Meaning of \*

### Program "memory\_02.cpp"

```
#include <iostream>

void main() {
    int* p;           ←
    int i=5;
    p=&i;           ←
    cout << i << endl;

    *p = 3;          ←
    cout << i << endl;
    cout << p << endl;
}
```

### output shell

```
5
3
254fdd0
press ENTER to continue...
```

**int\*** is type *pointer to integer*  
**p** is declared as a *pointer to integer*  
**p = address of i**  
**object pointed by p = 3**

virtual address	memory content	allocated variables
...	?	
254fda8	254fdd0	p
...	...	
254fdd0	2	i
254fdd4	?	
254fdd8	?	
...	?	



## Uses of \*

The symbol \* can be used in three ways:

- 1) Ordinary multiplication
- 2) Declare a pointer to something
- 3) Get the object pointed by a pointer

### Program "memory\_04.cpp"

```
#include <iostream>

void main() {
    float a=2.4172;
    float* p;
    p=&a;
    *p=3.14159;
    cout << a << endl;
    cout << *p << endl;
}
```

### output shell

```
3.14159
3.14159
press ENTER to continue...
```



## Casting and conversion

Program "casting\_01.cpp"

```
#include <iostream>

void main() {
    float a=3.14159;
    int i;
    i=(int) a;
    cout << "a=" << a << endl;
    cout << "i=" << i << endl;
}
```

output shell

```
a=3.14159
i=3
press ENTER to continue...
```

float is converted (truncated) to integer

Program "casting\_02.cpp"

```
#include <iostream>

void main() {
    float a=3.14159;
    int* p;
    p=(int*) &a;
    cout << "a=" << a << endl;
    cout << "i=" << *p << endl;
}
```

output shell

```
a=3.14159
i=1078530000
press ENTER to continue...
```

The same 32 bits are written as a float and read as an integer



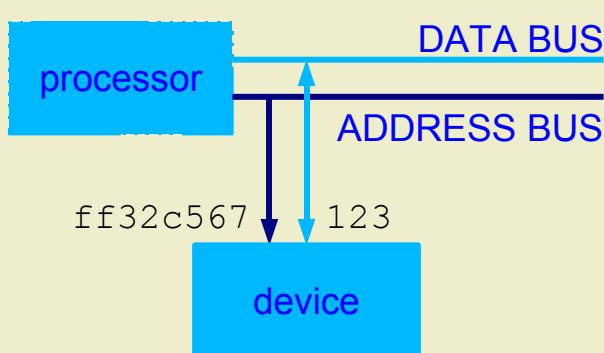
## Warning

Programs running in **User mode** should never access memory addresses that were not allocated by the program itself.

This may result in one of the following:

- 1) a runtime error: **segmentation fault**
- 2) corruption of data

Programs running in **Kernel mode** can use pointers to access physical memory and/or devices connected to the system bus.



Program "driver\_01.cpp"

```
int driver() {
    int* p=0xff32c567
    *p=123;          // write
    return *p;        // read
}
```

advanced

0x... indicates that ... is expressed in hexadecimal (a common notation).



### User mode

```
Program "direct_memory_access.cpp"
#include <iostream>

void main() {
    cout << "Hello World\n";
}
```

### Kernel mode

```
Program "console.cpp"
// ...
driver(x,y,"Hello World",11);
// ...
```

### Program "video\_card\_driver\_02.cpp"

```
void driver(int x, int y, char* s, int n) {
    char *video_card=0xef56da00;
    int i;
    for(i=0; i<n i++);
        video_card[80*y+x+i]=s[i];
}
```



## Arrays as Pointers

### Program "array\_01.cpp"

```
#include <iostream>

void main() {
    int array[3]={2,3,5};
    int *p;
    p=array;
    cout << *p << endl;
    cout << *(p+1) << endl;
    cout << *(p+2) << endl;
}
```

### Program "array\_02.cpp"

```
#include <iostream>

void main() {
    int array[3]={2,3,5};
    int *p;
    p=array;
    cout << p[0] << endl;
    cout << p[1] << endl;
    cout << p[2] << endl;
}
```

### output shell

```
2
3
5
press ENTER to continue...
```

C style arrays are implemented as pointers (even in C++)



## Passing arrays

Program "array\_03.cpp"

```
#include <iostream>

void set_array(int p[]) {
    p[0]=1; p[1]=2;
    cout << p[0] << p[1] << endl;
}

void main() {
    int a[2]={3,5};
    set_array(a);
    cout << a[0] << a[1] << endl;
}
```

output shell

```
12
12
press ENTER to continue...
```

Program "array\_04.cpp"

```
#include <iostream>

void set_array(int* p) {
    p[0]=1; p[1]=2;
    cout << p[0] << p[1] << endl;
}

void main() {
    int a[2]={3,5};
    set_array(a);
    cout << a[0] << a[1] << endl;
}
```

output shell

```
12
12
press ENTER to continue...
```

C-style arrays are always passed by reference  
(although the pointer to memory can be passed by value or by reference)



## Warning

While Java checks for array bounds and eventually return and `ArrayIndexOutOfBoundsException`, **C and C++ do not check for out of bound errors**. In the event this occurs there are two possibilities:

- 1) The program continues and eventually performs incorrectly.
- 2) The operative system catches the error and kills the program with a segmentation fault error (the most common error in the history of C/C++).

Program "bounds\_01.cpp"

```
#include <iostream>

void main() {
    int a[2]={3,5};
    a[3]=2;
    cout << a[3] << endl;
}
```

wrong

output shell  
2  
press ENTER to continue...

OR

output shell  
segmentation fault  
press ENTER to continue...



## Multidimensional arrays

### Program "array\_04.cpp"

```
#include <iostream>

const int N=2;

void print_array(int p[N][N]) {
    // access by p[i][j]
}

void main() {
    int a[N][N];
    print_array(a);
}
```

### Program "array\_05.cpp"

```
#include <iostream>

const int N=2;

void print_array(int* p) {
    // access p[i*N+j]
}

void main() {
    int a[N][N];
    print_array(a);
}
```

Array is always passed by reference  
(although the pointer to memory can be passed by value or by reference)

Warning: the notation **int\*\* p** exists but its meaning is different from **int p[][]**.  
**int\*\* p** means p is pointer to an arrays of pointers to integers. **int p[][]** means a pointer to a 2 dimensional array of integers. **int p[][]** is a pointer of type **int\* p**;



## More on passing by reference

### Program "reference\_01.cpp" (only C++)

```
#include <iostream>

void set(int& i) {
    i=3;
}

void main() {
    int j=5;
    set(j);
    cout << j << endl;
}
```

### Program "reference\_02.cpp" (C style)

```
#include <iostream>

void set(int* p) {
    *p=3;
}

void main() {
    int j=5;
    set(&j);
    cout << j << endl;
}
```

output shell

```
3
press ENTER to continue...
```

The two methods for passing by reference are equivalent.  
The pure C++ notation (left window) is cleaner (no use of \* and less subject to programmer errors) and, therefore, to be preferred.



Java Program

```
import java.io.*;  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        int p[] = new int[3]; // allocation  
        for(i=0; i<3; i++) {  
            p[i] = i;  
            System.out.println(toString(p[i]));  
        } // deallocation automatic  
    }  
}
```

output shell

```
0  
1  
2
```

press ENTER to continue...

Program "dynamic.cpp"

```
#include <iostream>  
  
void main(int arg, char** args) {  
    int* p = new int[3]; // allocation  
    for(i=0; i<3, i++) {  
        p[i] = i;  
        cout << p[i] << endl;  
    }  
    delete[] p; // deallocation  
}
```



```
class* var = new class[size];
```

Look for `sizeof(class)*size` bytes in memory, ask the Kernel to reserve the memory of the current process and return a pointer to the beginning of that memory. The pointer returned is of type `class*` and is stored into `var`. (*allocation*)

```
delete[] var;
```

Ask the Kernel to free (for other processes to use) the portion of memory, starting at pointer `var`, that was allocated by this process.  
(*deallocation*)

Remarks:

- 1) Anything that is allocated must be deallocated.
- 2) The same memory cannot be deallocated twice. This would result in a runtime error: ***bus error*** (the second most common error in the history of C and C++).



Program "dynamic\_02.cpp"

```
#include <iostream>

void main(int arg, char** args) {
    int* p=new int;
    *p=5;
    cout << p << endl;
    cout << *p << endl;
    delete p;
}
```

Single Object

`new type;`



`delete addr;`

Program "dynamic\_3.cpp"

```
#include <iostream>

void main(int arg, char** args) {
    int* p=new int[2];
    p[0]=5; p[1]=3;
    cout << p << endl;
    cout << p[0] << ", " << p[1] << endl;
    delete[] p;
}
```

Array of Objects

`new type[];`



`delete[] addr;`



Program "deallocation\_01.cpp" wrong

```
#include <iostream>

void set(char* p) {
    p[0]='a'; p[1]='b';
    delete[] p;
}

void main() {
    char* s=new char[2];
    set(s);
    cout << s[0] << s[1] << endl;
    delete[] s;
}
```

output shell

```
ab
bus error
press ENTER to continue...
```

## Warning using delete

Program "deallocation\_02.cpp"

```
#include <iostream>

void set(char* p) {
    p[0]='a'; p[1]='b';
}

void main() {
    char* s=new char[2];
    set(s);
    cout << s[0] << s[1] << endl;
    delete[] s;
}
```

output shell

```
ab
press ENTER to continue...
```



Tip: use these new/delete operators for debugging.

Program "mdp\_dynalloc.h"

```
#include "malloc.h"
void* operator new(size_t size) {
    cout << "allocating " << size << " bytes";
    void *p=malloc(size);
    cout << " at " << p << endl;
    return p;
}

void operator delete[] (void* pointer) {
    cout << "deallocating from " << pointer << endl;
    free(pointer);
}
```

**strict prototypes**

Program "test\_dynalloc\_01.cpp"

```
#include <iostream>
#include <malloc.h>
#include "dynalloc.h"
void main() {
    float* p=new float[7];
    delete[] p;
}
```

**OS calls**

**output shell**

```
allocating 28 bytes at 0x2670540
deallocating from 0x2670540
press ENTER to continue...
```



Example: average  
(passing arrays)

Program "average.cpp"

```
#include <iostream>
#include "mdp_dynalloc.h"

float average(float* p, long size) {
    float a=0;
    for(int i=0; i<size; i++) a+=p[i];
    return a/size;
}

void main() {
    float *p;
    long size;
    cout << "size="; cin >> size;
    p=new float[size];
    for(int i=0; i<size; i++) {
        cout << "p[" << i << "]=";
        cin >> p[i];
    }
    cout << "average=" << average(p,size) << endl;
    delete[] p;
}
```

output shell

```
size=3
allocating 12 bytes at 0x2670580
p[0]=2
p[1]=3.5
p[2]=1.25
average=2.25
deallocating from 0x2670580
press ENTER to continue...
```



Example: max  
(passing arrays)

Program "max\_1.cpp"

```
#include <iostream>
#include "mdp_dynalloc.h"

float max(float* p, long size) {
    float a=p[0];
    for(int i=1; i<size; i++) if(p[i]>a) a=p[i];
    return a;
}

void main() {
    float *p;
    long size;
    cout << "size="; cin >> size;
    p=new float[size];
    for(int i=0; i<size; i++) {
        cout << "p[" << i << "]=";
        cin >> p[i];
    }
    cout << "maximum=" << max(p,size) << endl;
    delete[] p;
}
```

output shell

```
size=3
allocating 12 bytes at 0x2670520
p[0]=2
p[1]=3.5
p[2]=1.25
maximum=3.5
deallocating from 0x2670520
press ENTER to continue...
```



Example: max  
(passing and returning arrays)

Program "max\_02.cpp"

```
#include <iostream>
#include "mdp_dynalloc.h"

float* input_size(long size) {
    float* p=new float[size];
    for(int i=0; i<size; i++) {
        cout << "p[" << i << "]=";
        cin >> p[i];
    }
    return p;
}
void max(float* p, long size) {
    float a=p[0];
    for(int i=1; i<size; i++) if(p[i]>a) a=p[i];
    cout << "maximum=" << a << endl;
}
void main() {
    long size;
    cout << "size="; cin >> size;
    float* p=input_size(size);
    max(p, size);
    delete[] p;
}
```

output shell

```
size=3
allocating 12 bytes at 0x2670520
p[0]=10
p[1]=12.45
p[2]=8.16
maximum=12.45
deallocating from 0x2670520
press ENTER to continue...
```



## Warning

If there is not enough memory available Java new operator throws an OutOfMemoryException. **C++ new operator throws bad\_alloc**

**The thrown object should be caught!**

Another common practice is to check for the return value of new.

### Program "out\_of\_memory.cpp"

```
#include <iostream>

void main() {
    char* p=new char[10000000000];
    if(p==0)
        cout << "out of memory\n";
    else {
        cout << "memory allocated\n";
        delete[] p;
    }
}
```

output shell

memory allocated  
press ENTER to continue...

OR

output shell

out of memory  
press ENTER to continue...



Program "max\_03.cpp"

```
#include <iostream>
#include "mdp_dynalloc.h"

float max(float* p, long size)    float a=p[0];
  for(int i=1; i<size; i++) if(p[i]>a) a=p[i];
  return a;
}
void main() {
  float *p;
  long size;
  cout << "size="; cin >> size;
  try {
    p=new float[size];
    if(p==0) throw Exception("OutOfMemory");
    for(int i=0; i<size; i++) {
      cout << "p[" << i << "]="; cin >> p[i];
    }
    cout << "maximum=" << max(p,size) << endl;
    delete[] p;
  } catch (Exception e) {
    cout << e.value() << endl;
  }
}
```

output shell  
size=3  
allocating 12 bytes at 0x2670520  
p[0]=10  
p[1]=12.45  
p[2]=8.16  
maximum=12.45  
deallocating from 0x2670520  
press ENTER to continue...

output shell  
size=100000000  
allocating 400000000 bytes at 0x0  
out of memory  
press ENTER to continue...



## Passing a pointer to pointers

While multidimensional arrays ([]) can be passed in two ways:

1) by copy

2) by reference (as it were a 1-dimensional array)

pointer to pointers (\*\*\*) should be passed as such.

While pointers (\*) and dynamically allocated arrays (\*\*) have to be deallocated,

regular arrays ([]), a in the example) are automatically deallocated.

Program "passing\_multiaray.cpp"

```
#include <iostream>

void f(int x[][10]) { }

void g(int* x) { }

void h(int** x) { }

void main() {
    int a[10][10]
    int** b=new int*[10];
    for(int i=0; i<10; i++)
        b[i]=new int[10];
    f(a);
    g(a);
    h(p);
    // do something ...
    for(int i=0; i<10; i++)
        delete[] b[i];
    delete[] b;
}
```



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 3

## Encapsulation: array of characters vs class string



Java Program

```
import java.io.*;  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        String s="Hello World";           ←  
        System.out.println(s);  
    }  
}
```

Program "array of char 01.cpp"

```
#include <iostream>  
  
void main(int arg, char** args) {  
    char* s="Hello World";           ←  
    cout << s << endl;  
}
```

address:

memory: ...|?|2765fe45|?|?|?|?|H|e|l|l|o||W|o|r|l|d|\0|?|?|..  
variable: s

Object

output shell

Hello World

press ENTER to continue...

Pointer to array of  
characters null ('\0')  
terminated



Program "array\_of\_char\_01.cpp"

```
#include <iostream>

void print(char* p) {
    for(;*p]!='\0';p++)
        cout << *p;
    cout << endl;
}

void main() {
    char* s="Hello World";
    cout << s << endl;
    print(s);
}
```

File "mdp\_cstring.h"

```
// ...
ostream& operator<< (ostream& os,
                      char* p) {
    for(;*p]!='\0';p++) os << *p;
    return os;
}
// ...
```

advanced



output shell

```
Hello World
Hello World
press ENTER to continue...
```

address:

memory: ...|?|2765fe45|?|2765fe45|?|?|H|e|l|l|o||W|o|r|l|d|\0|?|...
variable: s p (in print)

2765fe45



## strlen (length of C-string)

Program "use\_strlen\_01.cpp"

```
#include <iostream>
#include <cstring>

void main() {
    char* s="Hello World";
    cout << s << endl;
    cout << strlen(s) << endl;
    print(s);
}
```

File "mdp\_cstring.h"

```
// ...
int strlen(char* p) {
    int length=0;
    for(;*p]!='\0';p++) length++;
    return length;
}
// ...
```

advanced



output shell

```
Hello World
11
press ENTER to continue...
```

address:

memory: ... | ? | 2765fe45 | ? | 2765fe45 | ? | ? | H | e | l | l | o | | W | o | r | l | d | \0 | ? | ...
variable: s p (in print)

2765fe45



## strcpy (copy C-strings)

### Program "use strcpy\_01.cpp"

```
#include <iostream>
#include <cstring>

void main() {
    char* s="Hello World\n";
    char r[13];
    strcpy(r,s);
    cout << "r=" << r << endl;

    char* t;
    t=new char[strlen(s)+1];
    strcpy(t,s);
    cout << "t=" << t << endl;
    delete[] t;
}
```

### File "mdp\_cstring.h"

```
// ...
void strcpy(char *q, char* p) {
    int i;
    for(i=0; i<strlen(p)+1; i++)
        q[i]=p[i];
} // ...
```

advanced



unsafe

safe

### output shell

```
r=Hello World
t=Hello World
press ENTER to continue...
```



## strcat (concatenate C-strings)

### Program "use\_strcat\_01.cpp"

```
#include <iostream>
#include <cstring>

void main() {
    char* r="Hello ";
    char* s="World\n";
    int size=strlen(r)+strlen(s)+1;
    char* t=new char[size];
    strcpy(t,r);
    strcat(t,s);
    cout << "r=" << r << endl;
    cout << "s=" << s << endl;
    cout << "t=" << t << endl;
    delete[] t;
}
```

### File "mdp\_cstring.h"

```
// ...
void strcat(char *q, char* p) {
    int i, j=strlen(q);
    for(i=0; i<strlen(p)+1; i++)
        q[i+j]=p[i];
}
// ...
```

advanced



### output shell

```
r>Hello
s=World
t>Hello World
press ENTER to continue...
```



## strcmp (compare C-strings)

### Program "use\_strcmp\_01.cpp"

```
#include <iostream>
#include <cstring>

void main() {
    char* r="test\n";
    char* s=new char[strlen(r)+1];
    strcpy(s,r);
    cout << (void*) r << endl;
    cout << (void*) s << endl;
    if(strcmp(r,s)==0)
        cout << "r is equal to s\n";
    else
        cout << "r and s differ\n";
    delete[] s;
}
```

### File "mdp\_cstring.h"

```
// ....
int strcmp(char *q, char* p) {
    int i;
    for(i=0; i<strlen(p)+1; i++)
        if(q[i]<p[i])      return -1;
        else if(q[i]>p[i]) return +1;
    return 0;
// ...
```

advanced



### output shell

```
0x401322
0x2670540
r is equal to s
press ENTER to continue...
```



## Passing C-strings

Program "passing\_cstrings\_01.cpp"

```
#include <iostream>
#include <cstring>

void set(char s[]) {
    strcpy(s, "Hello World");
    cout << s << endl;
}

void main() {
    char s[12] = "01234567890";
    set(s);
    cout << s << endl;
}
```

output shell

```
Hello World
Hello World
press ENTER to continue...
```

Program "passing\_cstrings\_02.cpp"

```
#include <iostream>
#include <cstring>

void set(char* s) {
    strcpy(s, "Hello World");
    cout << s << endl;
}

void main() {
    char s[12] = "01234567890";
    set(s);
    cout << s << endl;
}
```

output shell

```
Hello World
Hello World
press ENTER to continue...
```



**Java Program**

```
import java.io.*;  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        int i;  
        for(i=0; i<args.length; i++)  
            System.out.println(args[i]);  
    }  
}
```

**Program "use\_args.cpp"**

```
#include <iostream>  
  
int main(int argc, char** args) {  
    int i;  
    for(i=0; i<argc, i++)  
        cout << args[i] << endl;  
    return 0;  
}
```

**output shell**

```
>use_args.exe a 3 xx  
  
use_args.exe  
a  
3  
xx  
press ENTER to continue
```



C-strings are indispensable in C++ because many libraries use them (for example to pass a filename to a file IO function).

C-strings are unsafe because C++ does not check arrays for out of bounds errors.

Tip: pass a C-string to functions together with the array size and check for out of bound errors.

Program "passing cstrings\_03.cpp"

```
#include <iostream>
#include <cstring>

void f(char* s, int size) {
    cout << "array size = " << size << endl;
    cout << "string size = " << strlen(s) << endl;
}

void main(int arg, char** args) {
    const int N=128;
    char s[N]={"aabbccdd"};
    f(s, N);
}
```

output shell

```
array size = 128
string size = 8
press ENTER to continue...
```



*"Remember ... with power comes great responsibility"*

C++ dynamic allocation is powerful tool but an improper use may easily result in data corruption, incorrect computation and/or runtime errors (segmentation fault, bus error).

Use it responsibly: use encapsulation to hide pointers!

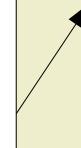
Program "string\_01.cpp"

```
#include "mdp_string"

void main(int argc, char** argv) {
    String a="Hello";
    String b=" World";
    String c=a+b;
    cout << c << endl;
    cout << "length=" << c.length() << endl;
}
```

output shell

```
Hello World
length=11
press ENTER to continue...
```





### Java example of class

```
public class MyClass {  
    // member variables  
  
    // constructor  
  
    // other methods  
}
```

### C++ example of class

```
class MyClass {  
public:  
    // member variables  
  
    // constructor (allocate stuff)  
    // destructor (deallocate stuff)  
    // copy constructor (how to copy class)  
    // assignment operator (how assign class)  
  
    // other methods  
};
```



## C++ example of class

```
class MyClass {  
public:  
    // member variables  
  
    MyClass() {  
        cout << "constructor: initializing member variables (new)\n";  
    }  
    ~MyClass() {  
        cout << "destructor: freeing memory (delete)\n";  
    }  
    MyClass(const MyClass& a) {  
        cout << "copy constructor: copy a into calling object\n";  
    }  
    MyClass& operator=(const MyClass& a) {  
        cout << "assignment operator: copy a into calling object\n";  
    }  
    // other methods  
};
```



C++ example of class

```
class MyClass {  
public:  
    someClass* pointer;  
  
    MyClass() {  
        pointer=new someClass[...];  
    }  
    ~MyClass() {  
        if(pointer!=0) delete[] pointer;  
    }  
    MyClass(const MyClass& a) {  
        ...  
        for(i...) pointer[i]=a.pointer[i];  
    }  
    MyClass& operator=(const MyClass& a) {  
        if (&a==this) return (*this);  
        ...  
        return (*this);  
    }  
    // other methods  
};
```



## class string

File "mdp\_string.h" (constructors/methods overview)

```
class String {  
private:  
    char* s;  
    int size;  
public:  
  
    String();                                // constructor  
    virtual ~String();                         // destructor  
    String(const String& p);                  // copy constructor  
    String& operator=(const String &p); // assignment operator  
  
    String(char* p);                          // converter constructor  
    String& operator=(char* p);               // converter assignment  
  
    char* c_str() const;                     // other  
    int length() const;                      // other  
    void resize(int i);                      // other  
};
```



For convenience: size = length() + 1 and length() will return size - 1



File "mdp\_string.h"

```
String() {
    cout << "call to constructor\n";
    s=new char[size=1];
    s[0]='\0';
}

~String() {
    cout << "call to destructor\n";
    delete[] s;
}

void resize(int i) {
    cout << " call to resize\n";
    if(s!=0) delete[] s;
    s=new char[size=i+1];
    s[0]='\0';
}
```



constructor

destructor

Program "test\_string\_01.cpp"

```
void main() {
    String a;
    a.resize(100);
}
```

output shell

```
call to constructor
call to resize
call to destructor
press ENTER to
continue...
```





File "mdp\_string.h"

```
Int length() const {
    return size-1;
}

String (const string& p) {
    cout << "call to c.c.\n";
    size=0; s=0;
    resize(p.length());
    strcpy(s,p.s);
}

String (char* p) {
    cout << "call to converter\n";
    size=0; s=0;
    resize(strlen(p));
    strcpy(s,p);
}
```

string length

copy constructor

converter

output shell

```
call to converter
call to resize
call to destructor
press ENTER to
continue...
```

Program "test\_string\_02.cpp"

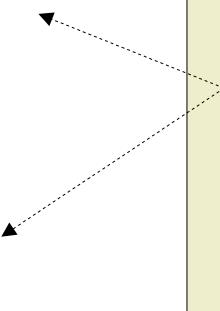
```
void main() {
    String a("This is a test");
}
```



## assignment operator

### File "mdp\_string.h"

```
String& operator= (const String& p) {
    cout << "operator=(string)\n";
    if(&p==this) return (*this);
    resize(p.length());
    strcpy(s,p.s);
    return *this;
}
String& operator= (char* p) {
    cout << "operator=(char*)\n";
    resize(strlen(p));
    strcpy(s,p);
    return *this;
}
char* c_str() const {
    return s;
}
```



### assignment operators

### Program "test\_string\_03.cpp"

```
void main() {
    String a;
    a="This is a test";
    cout << a.c_str() << endl;
}
```

### output shell

```
call to constructor
operator=(char*)
    call to resize
This is a test
call to destructor
press ENTER to
continue...
```





Program "test\_string\_04.cpp"

```
void main() {
    String a("This is a test");
    String b="this is test";
    String c;

    cout << "checkpoint 1\n";
    c="This is a test";

    cout << "checkpoint 2\n";
    c=a;
}
```

**calls to operator=**

**calls to c.c.**

**output shell**

```
call to converter      (a)
call resize            (a)
call to converter      (b)
call resize            (b)
call to constructor   (c)
checkpoint 1
operator=(char*)      (c)
call to resize         (c)
checkpoint 2
operator=(string)     (c)
call to resize         (c)
call destructor        (c)
call destructor        (b)
call destructor        (a)
press ENTER to continue...
```

Note that the symbol = following a class declaration constitutes a call to the copy constructor (c.c. or the converter) and not operator=



## example of iostream

### File "mdp\_string.h"

```
ostream& operator<<(ostream &os, String p) {  
    os << p.c_str();  
    return os;  
}  
  
istream& operator>>(istream& is, const String& p) {  
    static char buffer[1024];  
    if(is.peek()=='\n') is.ignore(1, '\n');  
    is.getline(buffer, 1024);  
    p=buffer; //here buffer is copied in p  
    return is;  
}
```

input/output  
for arbitrary stream



Attention:  
different from  
standard string!

### Program "test\_string\_05.cpp"

```
void main() {  
    String a;  
    cin >> a;  
    cout << "you typed\n";  
    cout << a << endl;  
}
```

### output shell

```
call to constructor      (a)  
nothing to say  
operator=(char*)  
    call to resize      (p)  
you typed  
nothing to say  
call to destructor  
press ENTER to continue...
```



File "mdp\_string.h"

```
string operator+(const String& a,
                  const String& b) {
    String c;
    c.resize(a.length() + b.length());
    strcpy(c.c_str(), a.c_str());
    strcat(c.c_str(), b.c_str());
    return c;
}
```



Program "test\_string\_06.cpp"

```
void main() {
    String a, b, c;
    cin >> a;
    cin >> b;
    c = a + b;
    cout << c;
}
```

output shell

[comments removed]  
aaaa  
bbbb  
aaaabbbb  
press ENTER to continue...



File "mdp\_string.h"

```
bool operator==(const String& a,
                  const String& b) {
    if(strcmp(a.c_str(), b.c_str())==0)
        return true;
    return false;
}

bool operator!=(const String& a,
                  const String& b) {
    if(strcmp(a.c_str(), b.c_str())!=0)
        return false;
    return true;
}
```



Program "test\_string\_06.cpp"

```
void main() {
    String a,b;
    cin >> a;
    cin >> b;
    if(a==b) cout << " == \n";
    if(a!=b) cout << " != \n";
}
```

output shell

[comments removed]  
aaaa  
bbbb  
!=  
press ENTER to continue...



File "mdp\_string.h"

```
bool operator<(const String& a,
                  const String& b) {
    return (strcmp(a.c_str(), b.c_str())<0);
}

bool operator>(const String& a,
                  const String& b) {
    return (strcmp(a.c_str(), b.c_str())>0);
}

bool operator<=(const String& a,
                  const String& b) {
    return (strcmp(a.c_str(), b.c_str())<=0);
}

bool operator>=(const String& a,
                  const String& b) {
    return (strcmp(a.c_str(), b.c_str())>=0);
}
```



Program

```
void main() {
    String a,b;
    cin >> a;
    cin >> b;
    if(a<b)
        cout << "<\n";
    if(a>b)
        cout << ">\n";
}
```

output shell

```
aaaa
aaba
<
press ENTER to continue...
```



Functions `strlen`, `strcpy`, `strcmp` and `strcat` are standard C/C++ libraries and are declared in the header "cstring" or "string.h"

class `string` is now standard in C++ and is declared in the header "string". In this lectures we built a similar class `string` in the file "mdp\_sstring.h"

The class `string` that we use in these lectures is based on 8bits characters and not 16bits wide characters (`wchar_t`) as in Java. Although the latter exists in C++ its use is not common. This difference must be take into account when transferring strings from Java to C/C++ and vice versa.

Program "test\_string\_07.cpp"

```
#include <iostream>
#include "mdp_string"
using namespace std;
void main() {
    String a,b,c;
    cin >> a;
    cin >> b;
    c=a+b;
    cout << c << endl;
}
```

in Visual C++  
this is required to use  
class `string` (ANSI '97)



## Passing a class by reference

Through encapsulation and operator overloading C++ allows us to extend the language.

As for int or float, a string object can be passed by reference or by value (with a call to c.c.).

**Program "test\_string\_08.cpp"**

```
#include <iostream>
#include "mdp_string.h"

void f(String s) {
    s="bbb";
}
void main() {
    String a="aaa";
    f(a);
    cout << a << endl;
}
```

**output shell**  
aaa  
press ENTER to continue...

**Program "test\_string\_09.cpp"**

```
#include <iostream>
#include "mdp_string.h"

void g(String& s) {
    s="bbb";
}
void main() {
    String a="aaa";
    g(a);
    cout << a << endl;
}
```

**output shell**  
bbb  
press ENTER to continue...



## Returning a class by reference

A class string (as any other class) should not be returned by reference unless 1) one returns **\*this**; 2) one returns an argument of that was passed by reference; 3) one returns a static local variable.

```
Program "test_string_10.cpp"
#include <iostream>
#include "mdp_string.h"

string& f() {
    return String("Hello");
}
void main() {
    cout << f() << endl;
}
```

wrong

```
Program "test_string_11.cpp"
#include <iostream>
#include "mdp_string.h"

string g() {
    return String("Hello");
}
void main() {
    cout << g() << endl;
}
```

output shell  
>**g++ test\_string\_10.cpp**  
Error:  
initialization of non-const

output shell  
Hello  
press ENTER to continue...



The class T declared below is typical for almost any class one may need,

If the class member variables do not include pointers it is possible to omit the destructor, the copy constructor and the assignment operator from the declaration, since the default ones are probably good enough (class S).

Program "class T.cpp"

```
class T {  
private:  
    // member variables and member pointers  
public:  
    T();                      // constructor  
    virtual ~T();             // destructor  
    T(const T&);            // copy constructor  
    T operator=(const T&);   // assignment op.  
    // member functions  
};  
  
ostream& operator<<(ostream& os, const T&);  
bool operator==(const T&, const T&);  
bool operator!=(const T&, const T&);  
// other operators ... and functions
```

Program "class S.cpp"

```
class S {  
private:  
    // member variables  
    // no pointers  
public:  
    S();  
    // member functions  
};  
  
ostream& operator<<(ostream& os, const S&);  
// other operators  
// and functions
```



Most important rules of C++ programming:

**Everything that is allocated must be deallocated (only once)**

**One should not pass or return by value an object that contains pointer(s), unless one has properly explicitly defined the copy constructor.**

**One should not call the assignment operator (=) of an object that contains pointer(s) unless one has explicitly properly defined the assignment operator.**

Failure to comply with these rules will result in memory leaks, runtime errors or wrong results!



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 4

## Classes and Objects (class Stack)



### Java Program

```
import java.io.*;  
  
public class HelloWorld {  
    public static void main(String args[]) {  
        System.out.println("Hello World");  
    }  
}
```

### Program "hello world 02.cpp"

```
#include <iostream>  
  
class HelloWorld {  
public:  
    static void main(int argc, char** args) {  
        cout << "Hello World\n";  
    }  
};  
  
void main(int argc, char** args) {  
    HelloWorld::main(argc, args)  
}
```



In C++ class and struct are similar except that class members are private by default (unless otherwise specified) while struct members are public by default (unless otherwise specified).

Program "class vs struct 01.cpp"

```
#include <iostream>

class A {
    void m() { } // by default private
};

struct B {
    void m() { } // by default public
};
```

by default private:

by default public:



Program "access modifiers 01.cpp"

```
#include <iostream>
class A {
public:
    int one() { return 1; }
protected:
    int two() { return 2; }
private:
    int three() { return 3; }
};

class B : public A {
public:
    int four() { return two()+one()+1; }
};

void main() {
    A a;
    B b;
    cout << a.one() << endl;
    cout << b.one() << endl;
    cout << b.four() << endl;
}
```

visible everywhere

only visible within the class  
and within derived classes

only visible within the class  
cannot be inherited

members of class B see  
one() and two() not three()

Only A::one(), B::one()  
and B::four() are visible.



Program "static and inline 01.cpp"

```
#include <iostream>

class A {
public:
    static int one() { return 1; }
    int two() { return 2; }
    inline int three() { return 3; }
};

void main() {
    cout << A::one() << endl;
    A a;
    cout << a.two() << endl;
    cout << a.three() << endl;
}
```

static member functions  
can be called even if the  
class is not instantiated

inlined functions are  
expanded inline by the  
compiler without function  
call. To be used for speed.



## friend

### Program "friend\_01.cpp"

wrong

```
#include <iostream>

class IntC {
private: int i;
public:
    void set(int j) { i=j; }
    int get() { return i; }
};

void print(IntC& a) {
    cout << a.i << endl;
}
void main() {
    IntC a;
    a.set(123);
    print(a);
}
```

### Program "friend\_02.cpp"

```
#include <iostream>

class IntC {
private: int i;
public:
    void set(int j) { i=j; }
    int get() { return i; }
    friend void print(IntC& a) {
        cout << a.i << endl;
    }
};

void main() {
    IntC a;
    a.set(123);
    print(a);
}
```

**friend** functions are called as ordinary functions (not methods) but can access private member variables and methods.



### Example: class Stack

Program "test\_stack\_01.cpp"

```
#include <iostream>
#include "mdp_stack.h"

void main() {
    Stack a;
    for(int i=0; i<5; i++) a.push(i);
    for(int i=0; i<5; i++) cout << a.pop(i);
}
```

output shell

43210

press ENTER to continue...

File "mdp\_stack.h" (constructors / methods overview)

```
class Stack {
public:
    enum {MaxStack = 5}; // constant!
    Stack();
    bool isEmpty() const;
    bool isFull() const;
    void push(int n);
    int pop();
    friend ostream& operator<<(ostream& os, const Stack& s);
private:
    int top; // pointer within the stack
    int arr[MaxStack]; // stack container
};
```



## Example: class Stack

File "mdp\_stack.h" (continue)

```
Stack() { top=-1; }
bool isEmpty() const { return top < 0; }
bool isFull() const { return top == MaxStack-1; }
void push(int n) {
    if(isFull())
        throw Exception("StackFullException");
    else
        arr[++top]=n;
}
int pop() {
    if(isEmpty())
        throw Exception("StackEmptyException");
    return arr[top--];
}
ostream& operator<<(ostream& os, const Stack& s) {
    if(s.isEmpty()) os << "[]";
    else {
        os << "[";
        for(int i=s.top; i>=0; i--) os << i << ":" << s.arr[i] << " ";
        os << "]";
    }
    return os;
}
```





## Example: better class Stack

```
void main() {
    Stack a(10);
    for(int i=0; i<10; i++) a.push(i);
    for(int i=0; i<10; i++) cout << a.pop(i);
}
```

```
output shell
9876543210
press ENTER to continue...
```

### File "mdp\_stack.h"

```
class Stack {
public:
    Stack(int i=5);
    ~Stack();
    Stack(const Stack& s);
    Stack& operator=(const Stack& s);
    bool isEmpty() const;
    bool isFull() const;
    void push(int n);
    int pop();
    friend ostream& operator<<(ostream& os, const Stack& s);
private:
    int MaxStack;
    int top; // pointer within the stack
    int* arr; // stack container
};
```

### File "mdp\_stack.h" (continue)

```
Stack(int i) {
    arr=new int[MaxStack=i];
    top=-1;
}

~Stack() {
    delete[] arr;
}
```





## Example: better class Stack

### File "mdp\_stack.h" (continue)

```
Stack(const Stack& s) {  
    top=s.top;  
    MaxStack=s.MaxStack;  
    arr=new int[MaxStack];  
    for(int i=0; i<MaxStack; i++)  
        arr[i]=s.arr[i];  
}  
operator=(const Stack& s) {  
    if(&s==this) return (*this);  
    delete[] arr;  
    top=s.top; MaxStack=s.MaxStack;  
    arr=new int[MaxStack];  
    for(int i=0; i<MaxStack; i++)  
        arr[i]=s.arr[i];  
    return *this;  
}
```

Copy Constructor



Assignment operator

```
void main() {  
    Stack s1(10); // constructor call  
    for(int i=0; i<10; i++) s1.push(i);  
    Stack s2=s1; // c.c call  
    for(int i=0; i<10; i++) cout << s2.pop();  
}
```

output shell

9876543210

press ENTER to continue...



The keyword **this**, used within a class, is a pointer to the present object (instantiation of the class).

**\*this** is the object itself.

The assignment operator  
must **return (\*this)**;

```
class T {  
    T& operator=(...) {  
        ...  
        return (*this);  
    }  
}
```

```
(*this).operator[](i)=a;
```

The keyword **this** is also commonly  
used to call overloaded operators

```
class T {  
    float a[100];  
    float& operator[](int i) {  
        return a[i];  
    }  
    void set(int i, float a) {  
        (*this)[i]=a;  
    }  
}
```



EQUIVALENT NOTATION

(\*something) .whatever

```
#include <iostream>

class Euclid {
public:
    float pi() {
        return 3.14159;
    }
}

void main() {
    Euclid* x=new Euclid;
    cout << (*x).pi() << endl;
    delete x;
}
```

something->whatever

```
#include <iostream>

class Euclid {
public:
    float pi() {
        return 3.14159;
    }
}

void main() {
    Euclid* x=new Euclid;
    cout << x->pi() << endl;
    delete x;
}
```



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 5

## Classes, Objects and Templates (class Vector, List)



```
class A {  
    int value[100];  
} a,c;  
class B {  
    int value[100];  
} b;
```

$a+b$

equivalent

**operator+(a,b)**

```
A operator+(const A& a,  
              const B& b) {  
    A c;  
    for(int i=0; i<100; i++)  
        c.value[i]=  
            a.value[i]+b.value[i];  
    return c;  
}
```

**a.operator+(b)**

```
A A::operator+(const B& b) {  
    A c;  
    for(int i=0; i<100; i++)  
        c.value[i]=  
            a.value[i]+b.value[i];  
    return c;  
}
```

Same for any **operator@** where @ can be any of the following:

**new new[] delete delete[] + - \* / % | &**  
**>> << >>= <<= > < >= <= == != && || -> ->\* ,**



### Java Program

```
import java.io.*;  
  
public class TestOverloading {  
    public static int square(int x) {  
        return x*x;  
    }  
    public static float square(float x) {  
        return x*x;  
    }  
    public static main() {  
        int i=2;  
        float a=3.1415926535897;  
        System.out.println(toString(square(i)));  
        System.out.println(toString(square(a)));  
    }  
}
```



## Name Overloading

### Program "overloading\_01.cpp"

```
#include <iostream>

int square(int x) {
    return x*x;
}

float square(float x) {
    return x*x;
}

void main() {
    int i=2;
    float a=3.1415926535897;
    cout << square(i)<< endl;
    cout << square(a)<< endl;
}
```

different functions:  
same names but different  
arguments and different bodies



Program "overloading\_02.cpp"

```
#include <iostream>

class C {
public:
    static int square(int x) {      ←
        return x*x;
    }
    static float square(float x) ←{
        return x*x;
    }
};

void main() {
    int i=2;
    float a=3.1415926535897;
    cout << C::square(i) << endl;
    cout << C::square(a) << endl;
}
```

different functions:  
same names but different  
arguments and different bodies



## Templates

### Program "templates\_01.cpp"

```
#include <iostream>

template<class T>
T square(T x) {
    return x*x;
}

void main() {
    int i=2;
    float a=3.1415926535897;
    cout << square(i) << endl;
    cout << square(a) << endl;
}
```

```
#include <iostream>

int square(int x) {
    return x*x;
}

float square(float x) {
    return x*x;
}

double square(double x) {
    return x*x;
}

// etc. etc...
```

different functions:  
same names and same bodies but  
different argument types



## CSC 309 – OOP in C++

Prof. Massimo Di Pierro

### Example: min and max

File "mdp\_algorithms.h"

```
template<class T>
T Min(const T& a, const T& b) {
    if(a<b) return a;
    return b;
}
```



File "mdp\_algorithms.h"

```
Template<class T>
T Max(const T& a, const T& b) {
    if(a>b) return a;
    return b;
}
```



Program "test\_min\_max\_01.cpp"

```
#include <iostream>
using namespace std;
#include "mdp_algorithms.h"

void main() {
    int a=3;
    int b=5;
    cout << Min(a,b) << endl;
    cout << Max(a,b) << endl;
}
```

output shell

```
3
5
press ENTER to continue...
```



### Example: swap with templates

File "mdp\_algorithms.h"

```
template<class T>
void Swap(T& a, T& b) {
    T c=a;
    a=b;
    b=c;
}
```



Program "test\_swap\_01.cpp"

```
#include <iostream>
using namespace std;
#include "mdp_string.h"
#include "mdp_algorithms.h"

void main() {
    String a="Hello";
    String b="World";
    Swap(a,b);
    cout << a << endl;
    cout << b << endl;
}
```

output shell

```
World
Hello
press ENTER to continue...
```



## Overloading operator[] and/or operator()

```
class A {  
    int value[100];  
} a;
```

a[i]=...  
↓  
**a.operator[](i)=**

```
int& A::operator[](int i)  
{  
    return value[i];  
}
```

...=a[i]

f(a[i])

a.operator+(b)

```
const int& A::operator[](int i)  
const {  
    return value[i];  
}
```

If you have the one to the left you probably want the one to the right,  
otherwise operator[] cannot be called within const methods.



## Example: Dynamic Vector

File "test\_vector\_01.cpp"

```
#include <iostream>
#include "mdp_vector.h"

Vector<int> square(Vector<int> a) {
    Vector<int> b(a.length());
    for(int i=0; i<a.length(); i++)
        b[i]=a[i]*a[i];
    return b;
}

void main() {
    Vector<int> a,b;
    a.resize(3);
    a[0]=3;
    a[1]=4;
    a[2]=5;
    cout << "a=" << a << endl;
    b=square(a);
    cout << "b=" << b << endl;
}
```



output shell

```
a=[3, 4, 5]
b=[9, 16, 25]
press ENTER to continue...
```



## Example: Dynamic Vector

### File "mdp\_vector.h"

```
Template <class T>
class Vector {
private:
    int size;
    T* p;

public:
    void resize(int i) {
        if(size!=0) delete[] p;
        size=i;
        if(size>0) p=new T[size];
    }

    Vector(int i=0) {
        size=0;
        resize(i);
    }

    ~Vector() {
        resize(0);
    }
}
```

```
Vector(const Vector& a) {
    size=0;
    resize(a.size);
    for(int i=0; i<size; i++)
        p[i]=a.p[i];
}

Vector& operator=(const Vector& a) {
    if(&a==this) return (*this);
    resize(a.size);
    for(int i=0; i<size; i++)
        p[i]=a.p[i];
    return (*this);
}

int length() const {
    return size;
}
```





## Example: Dynamic Vector

File "mdp\_vector.h" (continue)

```
T& operator[](int i) {
    if(i<0 || i>=size)
        throw Exception("VectorIndexOutOfBoundsException");
    return p[i];
}

const T& operator[](int i) const {
    if(i<0 || i>=size)
        throw Exception("VectorIndexOutOfBoundsException");
    return p[i];
}

friend ostream& operator<<(ostream& os, const Vector& a) {
    os << "[";
    if(a.size>0)
        cout << a[0];
    for(int j=1; j<a.size; j++)
        os << ", " << a[j];
    os << "]";
    return os;
}

}; // end class
```





## Example: check phases

### Program "check\_phases.cpp"

```
void check_phases() {
    const double Pi=3.14159265358979323846264;
    const int N=10;
    int i;
    double phase;

    Array<Complex> psi(N);
    Array<Complex> phi(N);
    Array<Complex> chi(N);

    for(i=0; i<N; i++) {
        phase=2.0*Pi*i/N;
        psi[i]=cos(phase)+I*sin(phase);
        phi[i]=cos(2.0*phase)+I*sin(2.0*phase);
        chi[i]=psi[i]*psi[i]-phi[i];
    }
    cout << chi << endl;
}
```

Test high school trigonometry:

$$\cos(2a) = \cos(a)^2 - \sin(a)^2$$

$$\sin(2a) = 2 \sin(a) \cos(a)$$

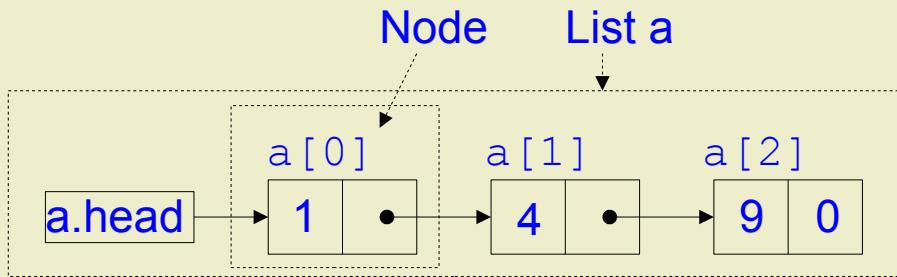
(in case you did not believe it!)

### output shell

```
[ 0+0*I, 0+0*I, 0+0*I,
 0+0*I, 0+0*I, 0+0*I,
 0+0*I, 0+0*I, 0+0*I,
 0+0*I ]
press ENTER to continue
...
```



## Example: Linked List



output shell  
a=[1, 4, 9]

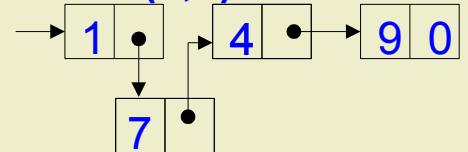
append(9)



remove(1)



insert(1,7)



output shell  
a=[1, 4, 9]  
b=[1, 5]  
press ENTER to continue...

Program "test\_list\_01.cpp"

```
#include <iostream.h>
#include "mdp_list.h"

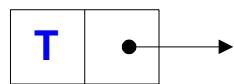
void main() {
    List<int> a,b;
    a.append(1);
    a.append(9);
    a.insert(1,4);
    cout << "a=" << a << endl;
    b=a;
    b[1]=5;
    b.remove(2);
    cout << "b=" << b << endl;
    pause();
}
```



## Example: Linked List

File "mdp\_list.h"

```
template <class T>
class List {
protected:
    class Node {
public:
    T      value;
    Node* next;
    Node(T a, Node* b=0) {
        value=a;
        next=b;
    }
};
```



```
private:
    Node* head;
    int size;
```



```
Public:
    List() { // constructor
        head=0;
        size=0;
    }
    ~List() { erase(); }
    erase() {
        for(int j=size-1; j>=0; j--)
            remove(j);
    }
    List(const List& list) { // c.c.
        head=0;
        size=0;
        for(int i=0;i<list.length();i++)
            append(list[i]);
    }
    List& operator=(const List& list) {
        if(&list==this) return (*this);
        erase();
        for(int i=0;i<list.length();i++)
            append(list[i]);
        return (*this);
    }
```





## Example: Linked List

File "mdp\_list.h" (continue)

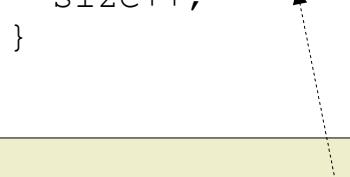
```
void append(T a) {  
    if(head==0) {  
        head=new Node(a, 0);  
        size++;  
    } else {  
        Node* p=head;  
        while(p->next!=0) p=p->next;  
        p->next=new Node(a, 0);  
        size++;  
    }  
}
```



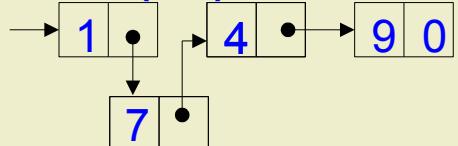
**append(9)**



```
void insert(int i, T a) {  
    if(i<0 || i>=size) throw  
        Exception("Out of bounds");  
    if(i==0)  
        head=new Node(a, head);  
    else {  
        Node* p=head;  
        for(int j=0;j<i-1;j++)  
            p=p->next;  
        p->next=new Node(a, p->next);  
    }  
    size++;
```



**insert(1,7)**



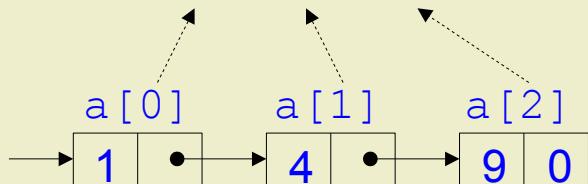


## Example: Linked List

File "mdp\_list.h" (continue)

```
T& operator[](int i) {
    if(i<0 || i>=size)
        throw Exception("ListIndexOutOfBoundsException");
    Node* p=head;
    for(int j=0; j<i; j++) p=p->next;
    return p->value;
}

const T& operator[](int i) const {
    if(i<0 || i>=size)
        throw Exception("ListIndexOutOfBoundsException");
    Node* p=head;
    for(int j=0; j<i; j++) p=p->next;
    return p->value;
}
```





## Example: Linked List

File "mdp\_list.h" (continue)

```
void remove(int i) {
    Node* q;
    if(i<0 || i>=size) throw
        Exception("Out of bounds");
    if(i==0) {
        q=head;
        head=head->next;
        delete q;
    } else {
        Node* p=head;
        for(int j=0; j<i-1; j++)
            p=p->next;
        q=p->next;
        p->next=q->next;
        delete q;
    }
    size--;
}
```

remove(1)



```
int length() const {
    return size;
}

friend ostream& operator<<
(ostream& os, const List& list) {
    os << "[";
    if(list.size>0)
        os << list[0];
    for(int j=1; j<list.size; j++)
        os << ", " << list[j];
    os << "]";
    return os;
}
}; // end class
```



output shell

[1, 4, 9]



### Example: remove newlines

Program "test\_replace\_string.h"

```
void replace_string() {
    int i,j,k;
    string in, out, filename, line;
    List<String> a;
    ifstream file;
    cout << "Insert a file name    :"; cin >> filename;
    cout << "String to be replaced:"; cin >> in;
    cout << "To be replaced with   :"; cin >> out;
    file.open(filename.c_str());
    while(true) {
        file >> line; if(file.fail()) break;
        a.append(line);
    }
    for(i=0; i<a.length(); i++) {
        for(k=0;
            k<a[i].length() && (j=a[i].find(in,k))>=0;
            k=k+j+out.length())
            a[i]=a[i].replace(j,in.length(),out);
        cout << a[i] << endl;
    }
    file.close();
}
```





## Example: QuickSort

File "mdp\_algorithms.h"

```
template<class T>
void QuickSort(T& A, int p, int r) {
    int i,j,q;
    if(p<r) {
        i=p-1;
        j=r+1;
        while(true) {
            for(i++;A[i]<A[p];i++);
            for(j--;A[j]>A[p];j--);
            if(i<j) Swap(A[i],A[j]);
            else { q=j; break; }
        }
        InsertionSort(A,p,q);
        InsertionSort(A,q+1,r);
    }
}

template<class T>
void QuickSort(T& A) {
    QuickSort(A,0,A.length()-1);
}
```



Program "test\_sort.h"

```
void sort_string() {
    String s;
    cout << "Input string:";
    cin >> s;
    InsertionSort(s);
    cout << "Sorted string :"
         << s << endl;
}
```



output shell

0872936451

0123456789

press ENTER to continue...



## Example: QuickSort

Program "test\_sort.h"

```
void sort_vector_of_int() {
    int i;
    cout << "Elements of the vector (1-10):"; cin >> i;
    cout << endl;
    Vector<int> a(i);
    for(i=0;i<a.length(); i++)
        cout << "a[" << i << "]="; cin >> a[i];

    InsertionSort(a);

    cout << "Sorted vector:\n";
    for(i=0;i<a.length(); i++)
        cout << "a[" << i << "]=" << a[i] << endl;
}
```





## Example: QuickSort

Program "test\_sort.h"

```
void sort_vector_of_string() {
    int i;
    cout << "Elements of the vector (1-10):";
    cin >> i;
    Vector<String> a(i);
    for(i=0; i<a.length(); i++)
        cout << "a[" << i << "]="; cin >> a[i];

    InsertionSort(a);

    cout << "Sorted vector:\n";
    for(i=0;i<a.length(); i++)
        cout << "a[" << i << "]=" << a[i] << endl;
}
```





## Example: QuickSort

Program "test\_sort.h"

```
void sort_list_of_string() {
    int i;
    String input;
    cout << "Insert array elements [ENTER] to finish";
    List<String> a;
    for(i=0;; i++)
        cout << "a[" << i << "]="; cin >> input;
        if(input!="") a.append(input); else break;

    InsertionSort(a);

    cout << "Sorted list:\n";
    for(i=0;i<a.length(); i++)
        cout << "a[" << i << "]=" << a[i] << endl;
}
```





CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 6

## MIDTERM



CSC 309 – OOP in C++  
Prof. Massimo Di Pierro

Week 7

## Inheritance (class Map)



## Inheritance

### Java Program

```
public class BC {  
    private int i;  
    public int get() {  
        return i;  
    }  
}  
  
public class DC extends BC {  
    public int getTwice() {  
        return 2*get();  
    }  
}
```

### Program "inheritance\_01.cpp"

```
class BC {  
    private: int i;  
    public: int get() {  
        return i;  
    }  
};  
  
class DC : public BC {  
    public: int getTwice() {  
        return 2*get();  
    }  
};
```



## Example: Map

### Output

```
a)ppend, d)elete, f)ind, p)rint, e)xit.:a  
Key :ccc  
Body:I love this class  
  
a)ppend, d)elete, f)ind, p)rint, e)xit.:a  
Key :bbb  
Body>Hello World  
  
a)ppend, d)elete, f)ind, p)rint, e)xit.:p  
0 : bbb : Hello World  
1 : ccc : I love this class  
  
a)ppend, d)elete, f)ind, p)rint, e)xit.:f  
Key :bbb  
Body:I love this class  
  
a)ppend, d)elete, f)ind, p)rint, e)xit.:e  
press ENTER to continue...
```

append record and sort

append record and sort

print all records

find record by key

Exit



## Example: Map

### Program "mdp\_map.h"

```
Void main_map() {
    int i;
    String choice, key, body;
    Map<String, String> db;
    while(true) {
        cout << "\n(a)ppend, (d)elete, (f)ind, (p)rint, (e)xit.:";
        cin >> choice;
        switch(choice[0]) {
            case 'a': cout << "Key :"; cin >> key;
                        cout << "Body:"; cin >> body;
                        db.appendRecord(key, body); break;
            case 'd': cout << "Key :"; cin >> key;
                        if(db.hasKey(key)) db.deleteRecord(key); break;
            case 'f': cout << "Key :"; cin >> key;
                        if(db.hasKey(key)) cout << "Body:" << db(key) << endl;
                        break;
            case 'p': for(i=0; i<db.length(); i++)
                        cout << i << " : " << db[i].key
                            << " : " << db[i].body << endl;
                        break;
            case 'e': return;
        }
    }
}
```





## Example: Map

File "mdp\_map.h"

```
Template<class S, class T>
class Record {
public:
    S key;
    T body;
    Record() {}
    Record(const S& s, const T& t) {
        key=s;
        body=t;
    }
    friend bool operator<(const Record& a,
                           const Record& b) {
        return (a.key<b.key);
    }
    friend bool operator>(const Record& a,
                           const Record& b) {
        return (a.key>b.key);
    }
};
```





## Example: Map

### File "mdp\_map.h" (continued)

```
Template<class S, class T>
class Map : public List<Record<S,T> > {

public:
    Map() { } // very important!!!
    int recordIndex(const S& key) const {
        for(int i=0; i<length(); i++)
            if((*this)[i].key==key) return i;
        return -1;
    }

    bool hasKey(const S& key) const {
        if(recordIndex(key)<0) return false;
        return true;
    }

    void appendRecord(const S& key, const T& body) {
        append(Record<S,T>(key,body));
        InsertionSort(*this);
    }
}
```





File "mdp\_map.h" (continued)

```
Public:  
    bool deleteRecord(const S& key) {  
        int i=recordIndex(key);  
        if(i<0) return false;  
        remove(i);  
        return true;  
    }  
  
    T& operator()(const S& key) {  
        int i=recordIndex(key);  
        if(i<0) throw Exception("MapIndexOutOfBoundsException");  
        return (*this)[i].body;  
    }  
  
    const T& operator()(const S& key) const {  
        int i=recordIndex(key);  
        if(i<0) throw Exception("MapIndexOutOfBoundsException");  
        return (*this)[i].body;  
    }
```





## Example: Map

### File "mdp\_map.h" (continued)

```
bool save(string filename) {
    ofstream file;
    file.open(filename.c_str());
    for(int i=0; i<length(); i++) {
        file << "RECORD N. " << i << endl;
        file << (*this)[i].key << endl;
        file << (*this)[i].body << endl;
    }
    file.close();
    return true;
}
```





## Example: Map

### File "mdp\_map.h" (continued)

```
bool load(String filename) {
    String dummy;
    S key;
    T body;
    ifstream file;
    file.open(filename.c_str());
    if(!file) return false;
    while(true) {
        file >> dummy;
        if(file.fail()) break;
        file >> key;
        file >> body;
        appendRecord(key, body);
    }
    file.close();
    return true;
}
}; // end class
```





## Inheritance, Interfaces and Polymorphism

*Polymorphism: overloading virtual methods*



## Inheritance and polymorphism

Program "inheritance\_02.cpp"

```
#include <iostream>
class BC {
public:
    void f() {
        cout << "BC::f()\n";
    }
};
class DC : public BC {
public:
    void f() {
        cout << "DC::f()\n";
    }
};
void main() {
    BC b;
    DC d;
    b.f();
    d.f();
}
```

Program "inheritance\_03.cpp"

```
#include <iostream>
class BC {
public:
    void f() {
        cout << "BC::f()\n";
    }
};
class DC : public BC {
public:
    void f() {
        cout << "DC::f()\n";
    }
};
void main() {
    BC* b1=new BC;
    BC* b2=new DC;
    b1->f();
    b2->f();
}
```

output shell

```
BC::f()
DC::f()
press ENTER to continue ...
```

output shell

```
BC::f()
BC::f()
press ENTER to continue ...
```



## Inheritance: constructors and destructors

### Program "inheritance\_03.cpp"

```
#include <iostream>

class BC {
public:
    BC() {
        cout << "BC constructor\n";
    }
};

class DC : public BC {
public:
    DC() {
        cout << "DC constructor\n";
    }
};

void main() {
    DC d;
}
```

### Program "inheritance\_04.cpp"

```
#include <iostream>

class BC {
public:
    ~BC() {
        cout << "BC destructor\n";
    }
};

class DC : public BC {
public:
    ~DC() {
        cout << "DC destructor\n";
    }
};

void main() {
    DC d;
}
```

### output shell

```
BC constructor
DC constructor
press ENTER to continue ...
```

### output shell

```
DC destructor
BC destructor
press ENTER to continue ...
```



## Inheritance: constructor warning

Program "inheritance\_05.cpp" dangerous

```
#include <iostream>

class BC {
public:
    BC() {
        cout << "BC constructor\n";
    }
};

class DC : public BC {
public:
    DC(int n) {           .....fix
        cout << "DC constructor\n";
    }
};

void main() {
    DC d;
}
```

Program "inheritance\_06.cpp"

```
#include <iostream>

class BC {
public:
    BC() {
        cout << "BC constructor\n";
    }
};

class DC : public BC {
public:
    DC(int n) : BC() {           .....fix
        cout << "DC constructor\n";
    }
};

void main() {
    DC d;
}
```

output shell  
compiler error

output shell  
BC constructor  
DC constructor  
press ENTER to continue ...



## Inheritance and Polymorphism: virtual functions

Program "virtual\_01.cpp"

```
#include <iostream>
class BC {
public:
    void speak() { .....fix
        cout << "BC speaks\n";
    }
};
class DC : public BC {
public:
    void speak() {
        cout << "DC speaks\n";
    }
};
void main() {
    BC *d=new DC();
    (*d).speak();
    delete d;
}
```

dangerous

Program "virtual\_02.cpp"

```
#include <iostream>
class BC {
public:
    virtual void speak() {
        cout << "BC speaks\n";
    }
};
class DC : public BC {
public:
    void speak() {
        cout << "DC speaks\n";
    }
};
void main() {
    BC *d=new DC();
    (*d).speak();
    delete d;
}
```

output shell

```
BC speaks
press ENTER to continue ...
```

output shell

```
DC speaks
press ENTER to continue ...
```



## Inheritance and Polymorphism: destructor warning

Program "virtual\_03.cpp"

```
#include <iostream>
class BC {
public:
    ~BC() { .....fix
        cout << "BC destructor\n";
    }
};

class DC : public BC {
public:
    ~DC() {
        cout << "DC destructor\n";
    }
};

void main() {
    BC* d=new DC();
    delete d;
}
```

dangerous

Program "virtual\_04.cpp"

```
#include <iostream>
class BC {
public:
    virtual ~BC() {
        cout << "BC destructor\n";
    }
};

class DC : public BC {
public:
    ~DC() {
        cout << "DC destructor\n";
    }
};

void main() {
    BC* d=new DC();
    delete d;
}
```

output shell

```
BC destructor
press ENTER to continue ...
```

output shell

```
DC destructor
BC destructor
press ENTER to continue ...
```



Java methods are all virtual by default  
(because Java does not distinguish compile-time binding vs run-time binding)

C++ methods are all non-virtual by default  
(because C++ prefers compile-time binding when possible)

If you understand inheritance in Java and want imitate it,  
declare all your C++ methods virtual...

...except the constructors which cannot be virtual!

Tip: declare the destructor always virtual.



## Inheritance and Polymorphism: name hiding

Program "virtual\_05.cpp"

wrong

```
#include <iostream>
class BC {
public:
    void m(int i) {
        cout << "m(int)\n";
    }
};

class DC : public BC {
public:
    void m() {
        cout << "m()\n";
    }
};

void main() {
    DC d;
    d.m(3); // not defined
}
```

Fix

Program "virtual\_06.cpp"

```
#include <iostream>
class BC {
public:
    void m(int i) {
        cout << "m(int)\n";
    }
};

class DC : public BC {
public:
    void m() {
        cout << "m()\n";
    }
    void m(int i) { BC::m(i); }
};

void main() {
    DC d;
    d.m(3); // OK here
}
```

derived method m() **hides** all  
inherited methods with same name

output shell  
m(int)  
press ENTER to continue ...



### Java Program

```
public interface IC {  
    public int get();  
}  
  
public class DC implements IC {  
    private int i;  
    public int get() {  
        return i;  
    }  
}
```

### Program "interface\_01.cpp"

```
class IC {  
    public: virtual int get()=0;  
};  
  
class DC : public IC {  
    private: int i;  
    public: int get() {  
        return i;  
    }  
};
```

An interface class (IC) in C++ is an ordinary class which methods are all purely virtual (**virtual ... =0**). Such a class is called Abstract Base Class.



Program "interface\_02.cpp"

```
#include <iostream>
class Widget {
public:
    virtual void show() { cout << "I, Widget\n"; }
};
class Square : public Widget {
public:
    virtual void show() { cout << "I, Square\n"; }
};
class Circle : public Widget {
public:
    virtual void show() { cout << "I, Circle\n"; }
};

void main() {
    Widget* w=(Widget*) new Square;
    (*w).show();
    delete w;
}
```

w can be Widget, Square or Circle

output shell

I, Square

press ENTER to continue ...



Program "interface\_03.cpp"

```
#include <iostream>
class Widget {
public:
    virtual void show()=0;
};
class Square : public Widget {
public:
    virtual void show() { cout << "I, Square\n"; };
};
class Circle : public Widget {
public:
    virtual void show() { cout << "I, Circle\n"; };
};

void main() {
    Widget* w=(Widget*) new Square;
    (*w).show();
    delete w;
}
```

Abstract Base Class  
(Interface Class)

must have  
show()

w can be Square or Circle but not  
Widget

output shell  
I, Square  
press ENTER to continue ...



Deck contains 40 cards (4 colors and 10 values: 1,2,3,4,5,6,7,8,9 and 10)

Cards are shuffled.

Each player (including Dealer) gets two cards.

Each player applies **his/her own strategy**:  
makes a bet against and asks for one or more card(s).

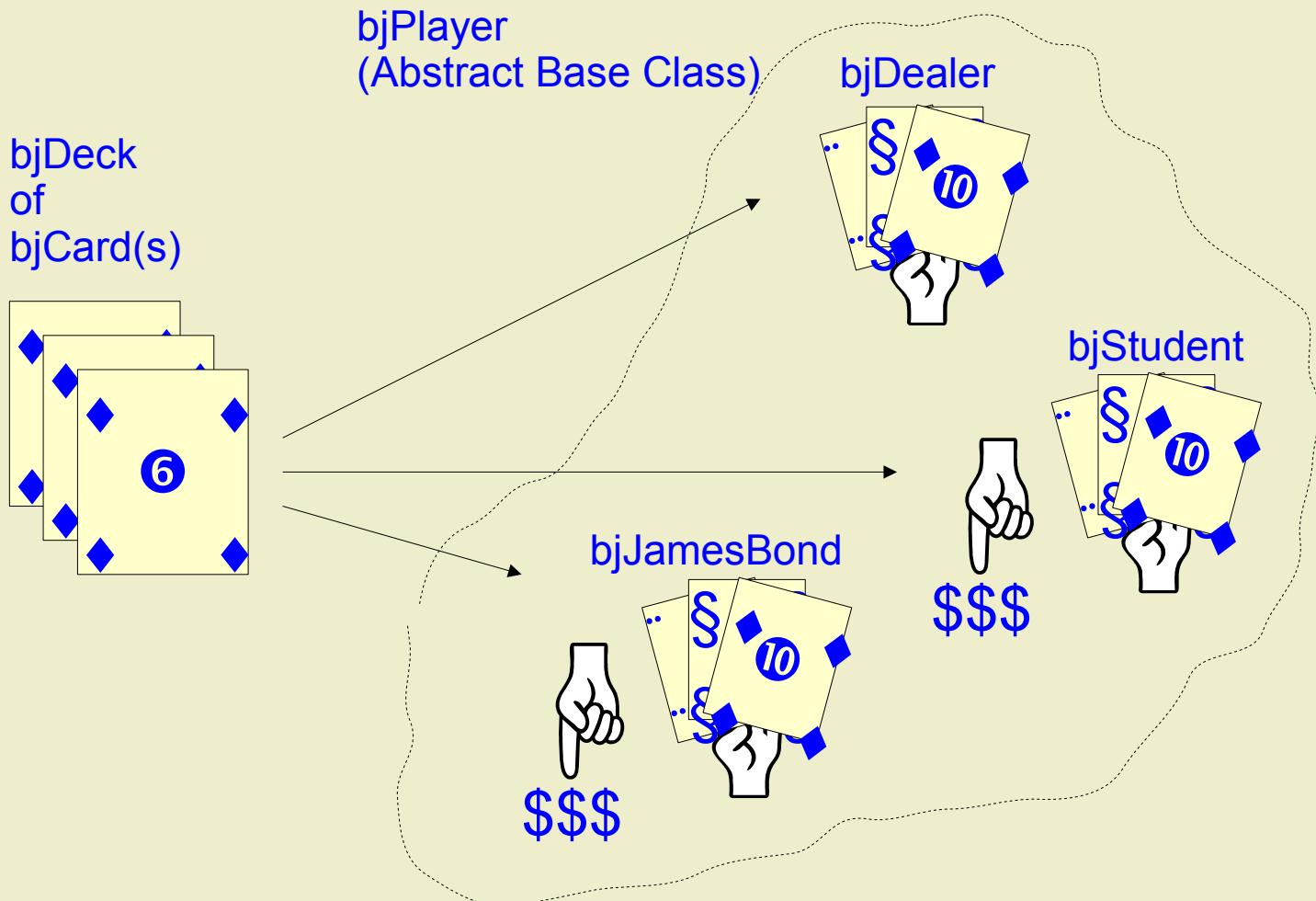
If a player gets a score closer to 21 than the Dealer,  
the Dealer pays player.

If a player exceeds 21 he/she pays the Dealer.

If the Dealer exceeds 21 he pays all players that did not exceed  
21 with their relative bets.



### Example: simplified Blackjack





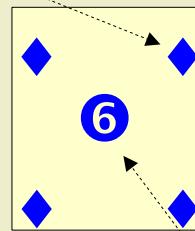
Example: simplified Blackjack

File "mdp\_blackjack.h"

```
class bjCard {  
public:  
    int c; // Color  
    int v; // Value  
    int color() const {  
        return c;  
    }  
    int value() const {  
        return v;  
    }  
    bjCard() {}  
    bjCard(int cc, int vv) {  
        c=cc;  
        v=vv;  
    }  
};
```



Color=♣ ♦ ♥ ♠



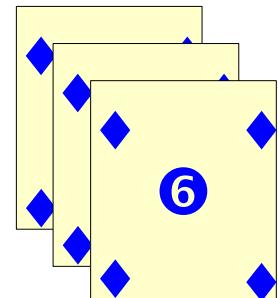
Value=1 2 3 4 5 6 7 8 9 10



## Example: simplified Blackjack

File "mdp\_blackjack.h" (continue)

```
class bjDeck {  
public:  
    enum {minColor=0, maxColor=3, minValue=1, maxValue=10 };  
    List<bjCard> cards;  
    bjDeck() {  
        int color, value;  
        for(color=minColor; color<=maxColor; color++)  
            for(value=minValue; value<=maxValue; value++)  
                cards.append(bjCard(color,value));  
    }  
    int remainingCards() const {  
        return cards.length();  
    }  
    bjCard getCard() {  
        if(cards.length()==0)  
            throw Exception("bjDeckEmptyException");  
        bjCard topcard=cards[0];  
        cards.remove(0);  
        return topcard;  
    }  
}
```





## Example: simplified Blackjack

File "mdp\_blackjack.h" (continue)

```
void shuffle(int n=2) {
    int i,j,k;
    for(k=0; k<n; k++) {
        for(i=0; i<cards.length(); i++) {
            // function rand() requires #include "stdlib.h"
            j=rand() % cards.length();
            Swap(cards[i],cards[j]);
        }
    }
};
```



♣:1	♣:2	♣:3	♣:4	♣:5	♣:6	♣:7	♣:8	♣:9	♣:10
♦:1	♦:2	♦:3	♦:4	♦:5	♦:6	♦:7	♦:8	♦:9	♦:10
♥:1	♥:2	♥:3	♥:4	♥:5	♥:6	♥:7	♥:8	♥:9	♥:10
♠:1	♠:2	♠:3	♠:4	♠:5	♠:6	♠:7	♠:8	♠:9	♠:10

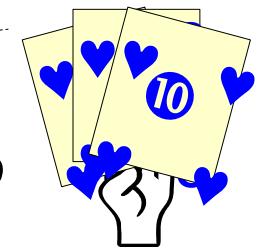
♣:1	♣:2	♣:3	♣:4	♣:5	♣:6	♣:7	♣:8	♣:9	♣:10
♦:1	♦:2	♦:3	♦:4	♦:5	♦:6	♦:7	♦:8	♦:9	♦:10
♥:1	♥:2	♦:5	♥:4	♥:5	♥:6	♥:7	♥:8	♥:9	♥:10
♠:1	♠:2	♠:3	♠:4	♠:5	♠:6	♠:7	♠:8	♠:9	♠:10



## Example: simplified Blackjack

File "mdp\_blackjack.h" (continue)

```
class bjPlayer {
public:
    string name;
    List<bjCard> hand;
    int bet;
    int portfolio;
    bjPlayer() { portfolio=0; handReset(); }
    void handReset() {
        bet=0;
        hand.erase();
    }
    void askCard(bjDeck& deck) {
        hand.append(deck.getCard());
    }
    int handValue() const {
        int value=0;
        for(int i=0; i<hand.length(); i++)
            value=value+hand[i].value();
        return value;
    }
    virtual void play(bjDeck&)=0;
};
```



\$\$\$

\$\$  
\$\$\$  
\$\$\$\$

game strategy





File "mdp\_blackjack.h" (continue)

```
class bjDealer : public bjPlayer {  
public:  
    void play(bjDeck& deck) {           ←-----  
        bet=100;  
        while(handValue()<17) askCard(deck);  
    }  
};
```

**game strategy**





File "mdp\_blackjack.h" (continue)

```
class bjStudent : public bjPlayer {  
public:  
    void play(bjDeck& deck) {      ←-----  
        bet=100;  
        while(handValue()<15) {  
            bet=bet+100;  
            askCard(deck);  
        }  
    }  
};
```

**game strategy**





File "mdp\_blackjack.h" (continue)

```
class bjJamesBond : public bjPlayer {  
public:  
    void play(bjDeck& deck) {      ←-----  
        bet=100;  
        while(handValue()<19) {  
            bet=2*bet;  
            askCard(deck);  
        }  
    }  
};
```

**game strategy**





## Example: simplified Blackjack

Program "mdp\_blackjack.h" (continue)

```
void play_blackjack() {
    bjDeck fulldeck, deck;
    List<bjPlayer*> players;
    int i,match, nmatches=100;

    // select the players
    players.append((bjPlayer*) new bjDealer);
    players.append((bjPlayer*) new bjStudent);
    players.append((bjPlayer*) new bjJamesBond);
    // append more if you like ...

    for(match=0; match<nmatches; match++) {
        deck=fulldeck;
        deck.shuffle();
        // each player asks for two cards and plays
        for(i=0; i<players.length(); i++) {
            players[i]->handReset();
            players[i]->askCard(deck);
            players[i]->askCard(deck);
            players[i]->play(deck);
        }
    }
}
```





## Example: simplified Blackjack

Program "mdp\_blackjack.h" (continue )

```
// settle bets
for(i=1; i<players.length(); i++) {
    if(players[i]->handValue()<=21 &&
       players[i]->handValue()>players[0]->handValue()) {
        players[i]->portfolio+=players[i]->bet;
        players[0]->portfolio-=players[i]->bet;
    } else {
        players[i]->portfolio-=players[i]->bet;
        players[0]->portfolio+=players[i]->bet;
    }
}
// print outcome of the match
cout << "MATCH N. " << match << endl;
for(i=0; i<players.length(); i++) {
    cout << "    player: " << i
        << ", bet: " << players[i]->bet
        << ", hand:" << players[i]->handValue()
        << ", portfolio: " << players[i]->portfolio << endl;
}
} // for ... match ...
// deallocate players ...
}
```





## Simulate different Blackjack strategies

### Output

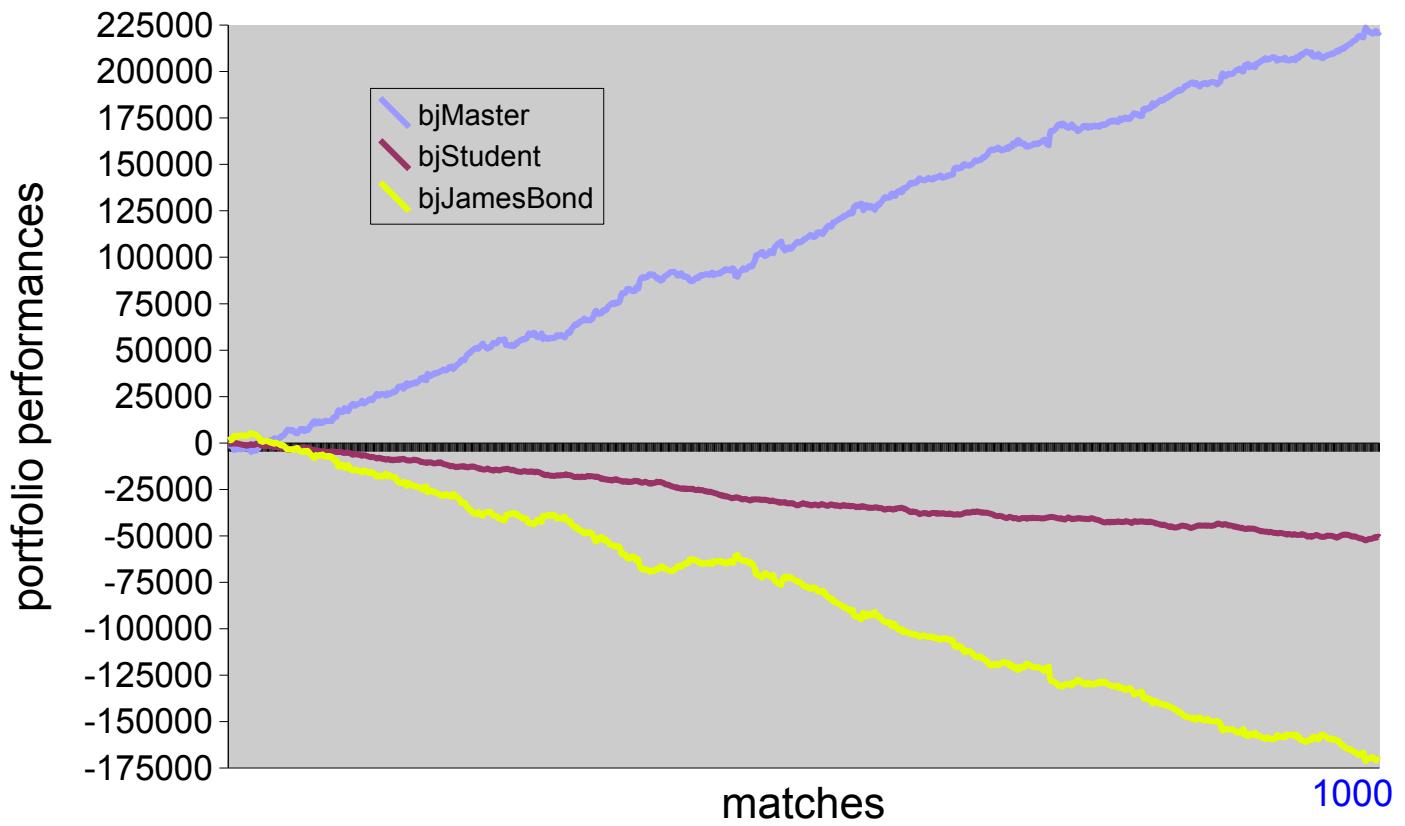
```
MATCH N. 0
    player: 0, bet: 0, hand:22, portfolio: 3500
    player: 1, bet: 300, hand:15, portfolio: -300
    player: 2, bet: 3200, hand:23, portfolio: -3200
MATCH N. 1
    player: 0, bet: 0, hand:22, portfolio: 3900
    player: 1, bet: 200, hand:15, portfolio: -500
    player: 2, bet: 200, hand:21, portfolio: -3400
MATCH N. 2
    player: 0, bet: 0, hand:19, portfolio: 3400
    player: 1, bet: 300, hand:20, portfolio: -200
    player: 2, bet: 200, hand:20, portfolio: -3200
...
MATCH N. 99
    player: 0, bet: 0, hand:20, portfolio: 48400
    player: 1, bet: 200, hand:17, portfolio: -19600
    player: 2, bet: 400, hand:24, portfolio: -28800
```

press ENTER to continue...

player 0 always  
wins on the  
long run!



## Blackjack: long-term performances





### Example: simplified Blackjack interactive game

File "mdp\_blackjack.h" (continue)

```
class bjInteractive : public bjPlayer {
public:
    void play(bjDeck& deck) {
        int i, b, c;
        bet=0;
        cout << "\nYour turn " << name << endl;
        while(handValue()<21) {
            cout << "You have the following cards:\n";
            for(i=0; i<hand.length(); i++)
                cout << hand[i].value() << " ";
            cout << "\nHow much do you want to bet? ";
            cin >> b;
            bet=bet+b;
            cout << "Your total bet is " << bet << endl;
            cout << "Do you want a card (0 - no, 1, yes)?";
            cin >> c;
            if(c==1) askCard(deck); else break;
        }
    }
};
```





Program "mdp\_blackjack.h" (continue)

```
void play_blackjack() {
    bjDeck fulldeck, deck;
    List<bjPlayer*> players;
    int i,match, nmatches=100;

    // selecet the players
    players.append((bjPlayer*) new bjDealer);
    players.append((bjPlayer*) new bjStudent);
    players.append((bjPlayer*) new bjJamesBond);
    players.append((bjPlayer*) new bjInteractive);

    players[3]->name="Massimo";

    for(match=0; match<nmatches; match++) {
        deck=fulldeck;
        deck.shuffle();
        // each player asks for two cards and plays
        for(i=0; i<players.length(); i++) {
            players[i]->handReset();
            players[i]->askCard(deck);
            players[i]->askCard(deck);
            players[i]->play(deck);
        }
    }
}
```





## Playing against the virtual players

### Output

```
Your turn Massimo
You have the following cards:
5 8
How much do you want to bet? 1000
Your total bet is 1000
Do you want a card (0 - no, 1, yes)?1
You have the following cards:
5 8 1
How much do you want to bet? 1000
Your total bet is 2000
Do you want a card (0 - no, 1, yes)?0
MATCH N. 0
    player: 0 , bet: 100, hand:19, portfolio: 700
    player: 1 , bet: 300, hand:15, portfolio: -300
    player: 2 , bet: 1600, hand:21, portfolio: 1600      ←-----
    player: 3 Massimo, bet: 2000, hand:14, portfolio: -2000

...
press ENTER to continue...
```



### Our classes

String  
Vector<T>  
List<T>  
Map<S,T>



### STL classes

string  
vector<T>  
list<T>  
map<S,T>

#### Advantages:

Portable (ANSI C++)  
Safe, use exceptions  
No need for iterators  
Database sorts elements

#### Advantages:

Commonly used  
Faster, Optimized for speed  
Extensive libraries and docs

#### Disadvantages:

Not optimized for speed  
Not many methods implemented

#### Disadvantages:

Different implementation may vary  
Use of exception not guaranteed  
map does not sort elements  
functions sort requires use of iterators

# CSC321. Notes (DRAFT)

Massimo Di Pierro

School of Computer Science, Telecommunications and Information Systems  
DePaul University, 243 S. Wabash Av, Chicago, IL 60604, USA

January 28, 2004

## **Abstract**

These notes provide a concise review to some of the material covered in the csc321 course. These notes are not intended as a substitution for the textbook. Attention: these notes are in a draft stage and may contain errors. Please report errors to mdipierro@cs.depaul.edu

# CONTENTS

<b>1 Theory</b>	<b>3</b>
1.1 Introduction . . . . .	3
1.1.1 Pseudocodes vs Java vs Python: . . . . .	4
1.2 Math Review . . . . .	4
1.2.1 Symbols . . . . .	4
1.2.2 Set Theory Review . . . . .	5
1.2.3 Logarithms . . . . .	11
1.2.4 Finite sums . . . . .	12
1.2.5 Limits ( $x \rightarrow \infty$ ) . . . . .	13
1.3 Sorting functions . . . . .	16
1.3.1 Order of growth of functions . . . . .	17
1.4 Introduction to study of algorithms . . . . .	19
1.4.1 Maximum and Minimum . . . . .	19
1.4.2 Insertion Sort . . . . .	22
1.4.3 Recurrence relations . . . . .	22
1.4.4 MergeSort . . . . .	31
1.4.5 Order of growth of algorithms . . . . .	32
<b>2 Data Structures and Algorithms</b>	<b>33</b>
2.1 CSC321 program . . . . .	33
2.1.1 Lists . . . . .	34
2.1.2 Trees . . . . .	35
2.1.3 Graphs . . . . .	36
2.2 Algorithms . . . . .	38
2.2.1 Algorithm types: definitions . . . . .	38
2.2.2 Sorting algorithms . . . . .	39
2.2.3 Searching algorithms . . . . .	45
2.2.4 Graph algorithms . . . . .	50

2.2.5	More examples . . . . .	57
2.3	NP and NPC . . . . .	65
<b>I</b>	<b>Appendices</b>	<b>67</b>
2.4	Programs in Java . . . . .	68
<b>3</b>	<b>Practice Exercises</b>	<b>72</b>
3.1	Limits . . . . .	72

# 1. THEORY

## 1.1. Introduction

An algorithm is a step-by-step procedure for solving a problem, and it is typically developed before doing any programming. In fact, it is independent of any programming language. Efficient algorithms can have a dramatic effect on our problem-solving capabilities. The issues that will concern us when developing and analyzing algorithms are:

1. correctness: of the problem specification, of the proposed algorithm, and of its implementation in some programming language (we will not worry about the third one: program verification is another subject altogether).
2. amount of work done: i.e., running time of the algorithm in terms of the input size (independent of hardware and programming language).
3. amount of space used: here we mean the amount of extra space beyond the size of the input (independent of hardware and programming language). We will say that an algorithm is *in place* if the amount of extra space is constant with respect to input size.
4. simplicity, clarity: unfortunately the simplest is not always the best in other ways.
5. optimality: can we prove that it does the best of any algorithm?

### 1.1.1. Pseudocodes vs Java vs Python:

	<b>Book pseudo-code</b>	<b>Java/C++</b>	<b>Python</b>
assignment	$a \leftarrow b$ ( <i>ambiguous!</i> )	$a = b;$	$a = b$
comparison	<code>if <math>a = b</math></code>	<code>if (<math>a == b</math>)</code>	<code>if <math>a == b</math>:</code>
loops	<code>for <math>a \leftarrow 0</math> to <math>n - 1</math></code>	<code>for(<math>a = 0; a &lt; n; a ++</math>)</code>	<code>for <math>a</math> in <math>range(0, n)</math>:</code>
block	indentation	{...}	indentation
function	<code>function <math>F(a)</math></code>	<code>void <math>F(\text{int } a[])</math> {</code>	<code>def <math>F(a)</math>:</code>
function call	$F[a]$	$F(a)$	$F(a)$
arrays/lists	$A_i$	$A[i]$	$A[i]$
member	? member[ $A$ ]	$A.\text{member}$	$A.\text{member}$
nothing	0	<i>null</i> / <code>void*</code>	<i>None</i>

## 1.2. Math Review

### 1.2.1. Symbols

$\infty$	infinity	
$\wedge$	and	
$\vee$	or	
$\cap$	intersection	
$\cup$	union	
$\in$	element or In	(1.1)
$\forall$	for each	
$\exists$	exists	
$\Rightarrow$	implies	
:	such that	
iff	if and only if	

### 1.2.2. Set Theory Review

#### Important Sets

<b>0</b>	empty set	
$\mathbb{N}$	natural numbers $\{0,1,2,3,\dots\}$	
$\mathbb{N}^+$	positive natural numbers $\{1,2,3,\dots\}$	
$\mathbb{Z}$	all integers $\{\dots,-3,-2,-1,0,1,2,3,\dots\}$	
$\mathbb{R}$	all real numbers	
$\mathbb{R}^+$	positive real numbers (not including 0)	
$\mathbb{R}^*$	positive numbers including 0	

(1.2)

#### Set operations

$\mathcal{A}$ ,  $\mathcal{B}$  and  $\mathcal{C}$  are some generic sets.

- **Intersection**

$$\mathcal{A} \cap \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ and } x \in \mathcal{B}\} \quad (1.3)$$

- **Union**

$$\mathcal{A} \cup \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ or } x \in \mathcal{B}\} \quad (1.4)$$

- **Difference**

$$\mathcal{A} - \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ and } x \notin \mathcal{B}\} \quad (1.5)$$

Corresponding Python functions:

```
def Intersection(A,B):
    C=[]
    for element in A:
        if element in B:
            C=C+[element]
    return C

def Union(A,B):
    C=[]
    for element in A+B:
        if not element in C:
            C=C+[element]
```

```

    return C

def Difference(A,B):
    C=[]
    for element in A:
        if not element in B:
            C=C+[element]
    return C

```

### Set laws

- Empty set laws

$$\begin{aligned}\mathcal{A} \cup \emptyset &= \mathcal{A} \\ \mathcal{A} \cap \emptyset &= \emptyset\end{aligned}$$

- Idempotency laws

$$\begin{aligned}\mathcal{A} \cup \mathcal{A} &= \mathcal{A} \\ \mathcal{A} \cap \mathcal{A} &= \mathcal{A}\end{aligned}$$

- Commutative laws

$$\begin{aligned}\mathcal{A} \cup \mathcal{B} &= \mathcal{B} \cup \mathcal{A} \\ \mathcal{A} \cap \mathcal{B} &= \mathcal{B} \cap \mathcal{A}\end{aligned}$$

- Associative laws

$$\begin{aligned}\mathcal{A} \cup (\mathcal{B} \cup \mathcal{C}) &= (\mathcal{A} \cup \mathcal{B}) \cup \mathcal{C} \\ \mathcal{A} \cap (\mathcal{B} \cap \mathcal{C}) &= (\mathcal{A} \cap \mathcal{B}) \cap \mathcal{C}\end{aligned}$$

- Distributive laws

$$\begin{aligned}\mathcal{A} \cap (\mathcal{B} \cup \mathcal{C}) &= (\mathcal{A} \cap \mathcal{B}) \cup (\mathcal{A} \cap \mathcal{C}) \\ \mathcal{A} \cup (\mathcal{B} \cap \mathcal{C}) &= (\mathcal{A} \cup \mathcal{B}) \cap (\mathcal{A} \cup \mathcal{C})\end{aligned}$$

- Absorption laws

$$\begin{aligned}\mathcal{A} \cap (\mathcal{A} \cup \mathcal{B}) &= \mathcal{A} \\ \mathcal{A} \cup (\mathcal{A} \cap \mathcal{B}) &= \mathcal{A}\end{aligned}$$

- DeMorgan laws

$$\begin{aligned}\mathcal{A} - (\mathcal{B} \cup \mathcal{C}) &= (\mathcal{A} - \mathcal{B}) \cap (\mathcal{A} - \mathcal{C}) \\ \mathcal{A} - (\mathcal{B} \cap \mathcal{C}) &= (\mathcal{A} - \mathcal{B}) \cup (\mathcal{A} - \mathcal{C})\end{aligned}$$

## More set definitions

- $\mathcal{A}$  is a **subset** of  $\mathcal{B}$  iff  $\forall x \in \mathcal{A}, x \in \mathcal{B}$
- $\mathcal{A}$  is a **proper subset** of  $\mathcal{B}$  iff  $\forall x \in \mathcal{A}, x \in \mathcal{B}$  and  $\exists x \in \mathcal{B}, x \notin \mathcal{A}$
- $P = \{S_i, i = 1, \dots, N\}$  (a set of sets  $S_i$ ) is a **partition** of  $\mathcal{A}$  iff  $S_1 \cup S_2 \cup \dots \cup S_N = \mathcal{A}$  and  $\forall i, j, S_i \cap S_j = \emptyset$
- The number of elements in a set  $\mathcal{A}$  is called the **cardinality** of set  $\mathcal{A}$ .
- $\text{cardinality}(\mathbb{N}) = \text{countable infinite } (\infty)$
- $\text{cardinality}(\mathbb{R}) = \text{uncountable infinite } (\infty) !!!$

## Cantor's argument

Cantor proved that the real numbers in any interval (for example in  $[0, 1]$ ) are more than the integer numbers, therefore real numbers are uncountable. The proof proceeds as follow:

1. Consider the real numbers in the interval  $[0, 1)$  not including 1.
2. Assume that these real numbers are countable. Therefore is it possible to associate each of them to an integer

$$\begin{array}{rccc} 1 & \longleftrightarrow & 0.xxxxxxx... \\ 2 & \longleftrightarrow & 0.xxxxxxx... \\ 3 & \longleftrightarrow & 0.xxxxxxx... \\ 4 & \longleftrightarrow & 0.xxxxxxx... \\ 5 & \longleftrightarrow & 0.xxxxxxx... \\ \dots & \dots & \dots \end{array} \tag{1.6}$$

(here the  $x$  represent a decimal digits of a real numbers)

3. Now construct a number  $\alpha = 0.yyyyyyyyy....$  where the first decimal digit differs from the first decimal digit of the first real number of table 1.6, the second decimal digit differs from the second decimal digit of the second real number of table 1.6 and so on and on for all the infinite decimal digits:

$$\begin{aligned}
 1 &\longleftrightarrow 0.\overline{x}xxxxxx... \\
 2 &\longleftrightarrow 0.x\overline{x}xxxxxx... \\
 3 &\longleftrightarrow 0.xx\overline{x}xxxxxx... \\
 4 &\longleftrightarrow 0.xxx\overline{x}xxxxx... \\
 5 &\longleftrightarrow 0.xxxx\overline{x}xxxx... \\
 \dots &\dots \dots
 \end{aligned} \tag{1.7}$$

4. The new number  $\alpha$  is a real number and by construction it is not in the table. In fact it differs with each item for at least one decimal digit. Therefore the existence of  $\alpha$  disproves the assumption that all real numbers in the interval  $[0, 1)$  are listed in the table.

### Godel's Theorem

Godel used a similar diagonal argument to prove that there are as many problems (or theorems) as real numbers and as many algorithms (or proofs) as natural numbers. Since there is more of the former than the latter it follows that there are problems for which there is no corresponding solving algorithm. Another interpretation of Goedel's theorem is that, in any formal language, for example mathematics, there are theorems that cannot be proved.

**Proposition 1.** *It is impossible to write a computer program to test if a given algorithm stops or enters into an infinite loop.*

**Example 2.** *Just as an example of the above statement, it is not known if the following program stops:*

```
def verify(i):
    s=str(i*i)
    d=[]
    for c in s:
```

```

if not c in d:
    d.append(c)
if len(d)==2:
    return 1
else:
    return 0

def next(i):
    i=long(i)
    while 1:
        i=i+2
        if verify(i):
            print i, i*i
            break

next(81619)

```

While one day this problem may be solved there are many other problems there are still unsolved, actually there is an infinite number of them!

## Relations

- A **Cartesian Product** is defined as

$$\mathcal{A} \times \mathcal{B} = \{(a, b) : a \in \mathcal{A} \text{ and } b \in \mathcal{B}\} \quad (1.8)$$

- A **binary relation**  $R$  between two sets  $\mathcal{A}$  and  $\mathcal{B}$  if a subset of their Cartesian product.
- A binary relation is **transitive** if  $aRb$  and  $bRc$  implies  $aRc$
- A binary relation is **symmetric** if  $aRb$  implies  $bRa$
- A binary relation is **reflexive** if  $aRa$  is always true for each  $a$ .

**Example 3.**  $a < b$  for  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$  is a relation (transitive)

**Example 4.**  $a > b$  for  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$  is a relation (transitive)

**Example 5.**  $a = b$  for  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$  is a relation (transitive, symmetric and reflexive)

**Example 6.**  $a \leq b$  for  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$  is a relation (transitive, and reflexive)

**Example 7.**  $a \geq b$  for  $a \in \mathcal{A}$  and  $b \in \mathcal{B}$  is a relation (transitive, and reflexive)

- A relation  $R$  that is transitive, symmetric and reflexive is called an **equivalence relation** and is often indicated with the notation  $a \sim b$ .

**Theorem 8.** An equivalence relation is the same as a partition.

Corresponding Python functions:

```
def CartesianProduct(A,B):  
    C=[]  
    for a in A:  
        for b in B:  
            C=C+[(a,b)]  
    return C
```

## Functions

- A **function** between two sets  $\mathcal{A}$  and  $\mathcal{B}$  is a binary relation on  $\mathcal{A} \times \mathcal{B}$  and is usually indicated with the notation  $f : \mathcal{A} \rightarrow \mathcal{B}$
- The set  $\mathcal{A}$  is called **domain** of the function.
- The set  $\mathcal{B}$  is called **codomain** of the function.
- A function **maps** each element  $x \in \mathcal{A}$  into an element  $f(x) = y \in \mathcal{B}$
- The **image** of a function  $f : \mathcal{A} \rightarrow \mathcal{B}$  is the set  $\mathcal{B}' = \{y \in \mathcal{B} : \exists x \in \mathcal{A}, f(x) = y\} \subseteq \mathcal{B}$
- If  $\mathcal{B}' = \mathcal{B}$  then a function is said to be **surjective**.
- If for each  $x$  and  $x'$  in  $\mathcal{A}$  where  $x \neq x'$  implies that  $f(x) \neq f(x')$  (i.e. if not two different elements of  $\mathcal{A}$  are mapped into the same element in  $\mathcal{B}$ ) the function is said to be a **bijection**.

- A function  $f : \mathcal{A} \rightarrow \mathcal{B}$  is invertible if it exists a function  $g : \mathcal{B} \rightarrow \mathcal{A}$  such that for each  $x \in \mathcal{A}, g(f(x)) = x$  and  $y \in \mathcal{B}, f(g(y)) = y$ . The function  $g$  is indicated with  $f^{-1}$ .
- A function  $f : \mathcal{A} \rightarrow \mathcal{B}$  is a surjection and a bijection iff  $f$  is an invertible function.

**Example 9.**  $f(n) \stackrel{\text{def}}{=} n \bmod 2$  with domain  $\mathbb{N}$  and codomain  $\mathbb{N}$  is not a surjection nor a bijection.

**Example 10.**  $f(n) \stackrel{\text{def}}{=} n \bmod 2$  with domain  $\mathbb{N}$  and codomain  $\{0, 1\}$  is a surjection but not a bijection

**Example 11.**  $f(x) \stackrel{\text{def}}{=} 2x$  with domain  $\mathbb{N}$  and codomain  $\mathbb{N}$  is not a surjection but is a bijection (in fact it is not invertible on odd numbers)

**Example 12.**  $f(x) \stackrel{\text{def}}{=} 2x$  with domain  $\mathbb{R}$  and codomain  $\mathbb{R}$  is not a surjection and is a bijection (in fact it is invertible)

### 1.2.3. Logarithms

If  $x = a^y$  with  $a > 0$  then  $y = \log_a x$  with domain  $x \in (0, \infty)$  and co-domain  $y = (-\infty, \infty)$ . If the base  $a$  is not indicated the natural  $\log a = e = 2.7183\dots$  is assumed.

Useful properties of logarithms:

$$\begin{aligned}\log_a x &= \frac{\log x}{\log a} \\ \log xy &= (\log x) + (\log y) \\ \log \frac{x}{y} &= (\log x) - (\log y) \\ \log x^n &= n \log x\end{aligned}$$

Note that  $\log^n x = (\log x)^n$  is not the same as  $\log x^n = \log(x^n)$ , therefore the former is not equal to  $n \log x$ .

### 1.2.4. Finite sums

#### Definition

$$\sum_{i=0}^{i < n} f(i) \stackrel{def}{=} f(0) + f(1) + \dots + f(n-1) \quad (1.9)$$

Corresponding Python functions:

```
def Sum(f,min_i,max_i):
    a=0
    for i in range(min_i,max_i):
        a=a+f(i)
    return a
```

#### Properties

- **Linearity I**

$$\begin{aligned} \sum_{i=0}^{i \leq n} f(i) &= \sum_{i=0}^{i < n} f(i) + f(n) \\ \sum_{i=a}^{i \leq b} f(i) &= \sum_{i=0}^{i \leq b} f(i) - \sum_{i=0}^{i < a} f(i) \end{aligned}$$

- **Linearity II**

$$\sum_{i=0}^{i < n} af(i) + bg(i) = a \left( \sum_{i=0}^{i < n} f(i) \right) + b \left( \sum_{i=0}^{i < n} g(i) \right) \quad (1.10)$$

Proof:

$$\begin{aligned} \sum_{i=0}^{i < n} af(i) + bg(i) &= (af(0) + bg(0)) + \dots + (af(n-1) + bg(n-1)) \\ &= af(0) + \dots + af(n-1) + bg(0) + \dots + bg(n-1) \\ &= a(f(0) + \dots + f(n-1)) + b(g(0) + \dots + g(n-1)) \\ &= a \left( \sum_{i=0}^{i < n} f(i) \right) + b \left( \sum_{i=0}^{i < n} g(i) \right) \end{aligned} \quad (1.11)$$

**Example 13.**

$$\begin{aligned}
 \sum_{i=0}^{i < n} c &= cn \text{ for any constant } c \\
 \sum_{i=0}^{i < n} i &= \frac{1}{2}n(n - 1) \\
 \sum_{i=0}^{i < n} i^2 &= \frac{1}{6}n(n - 1)(2n - 1) \\
 \sum_{i=0}^{i < n} i^3 &= \frac{1}{4}n^2(n - 1)^2 \\
 \sum_{i=0}^{i < n} x^i &= \frac{x^n - 1}{x - 1} \text{ (geometric sum)} \\
 \sum_{i=0}^{i < n} \frac{1}{i(i + 1)} &= 1 - \frac{1}{n} \text{ (telescopic sum)}
 \end{aligned}$$

**1.2.5. Limits ( $x \rightarrow \infty$ )**

In these section we will only deal with limits ( $x \rightarrow \infty$ ) of positive functions.

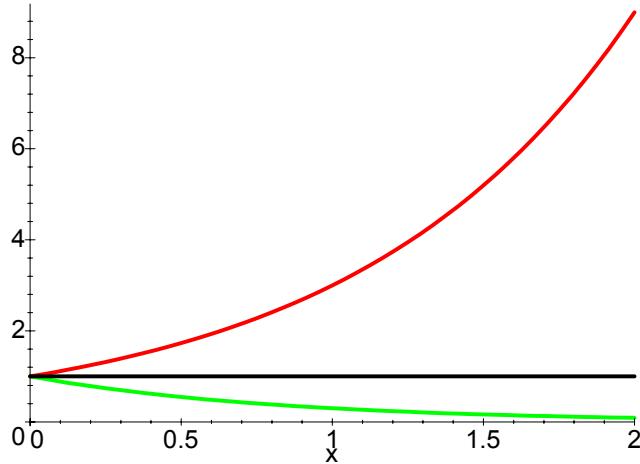
**Example 14.** Example of limits:

$$\lim_{x \rightarrow \infty} a^x = 0 \text{ if } (a < 1) \quad (1.12)$$

$$\lim_{x \rightarrow \infty} a^x = 1 \text{ if } (a = 1) \quad (1.13)$$

$$\lim_{x \rightarrow \infty} a^x = \infty \text{ if } (a > 1) \quad (1.14)$$

As we can see from the following plot



### Practical rule

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = ? \quad (1.15)$$

First compute limits of numerator and denominator separately

$$\begin{aligned}\lim_{x \rightarrow \infty} f(x) &= a \\ \lim_{x \rightarrow \infty} g(x) &= b\end{aligned}$$

- If  $a \in \mathbb{R}$  and  $b \in \mathbb{R}^+$  then

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \frac{a}{b} \quad (1.16)$$

- If  $a \in \mathbb{R}$  and  $b = \infty$  then

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \quad (1.17)$$

- If  $(a \in \mathbb{R}^+ \text{ and } b = 0)$  or  $(a = \infty \text{ and } b \in \mathbb{R})$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \infty \quad (1.18)$$

- If  $(a = 0 \text{ and } b = 0)$  or  $(a = \infty \text{ and } b = \infty)$  use de l'Hopital rule

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = \lim_{x \rightarrow \infty} \frac{f'(x)}{g'(x)} \quad (1.19)$$

and start again!

- Else ... the limit does not exist (typically oscillating functions or non-analytic functions).

**Theorem 15.** For any  $a \in \mathbb{R}$  or  $a = \infty$

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = a \Rightarrow \lim_{x \rightarrow \infty} \frac{g(x)}{f(x)} = 1/a$$

### Table of Derivatives

$f(x)$	$f'(x)$	
$c$	0	
$ax^n$	$anx^{n-1}$	
$\log x$	$\frac{1}{x}$	
$e^x$	$e^x$	
$a^x$	$a^x \log a$	
$x^n \log x, n > 0$	$x^{n-1}(n \log x + 1)$	

(1.20)

### Practical rules to compute derivatives

$$\begin{aligned}
\frac{d}{dx} (f(x) + g(x)) &= f'(x) + g'(x) \\
\frac{d}{dx} (f(x) - g(x)) &= f'(x) - g'(x) \\
\frac{d}{dx} (f(x)g(x)) &= f'(x)g(x) + f(x)g'(x) \\
\frac{d}{dx} \left( \frac{1}{f(x)} \right) &= -\frac{f'(x)}{f(x)^2} \\
\frac{d}{dx} \left( \frac{f(x)}{g(x)} \right) &= \frac{f'(x)}{g(x)} - \frac{f(x)g'(x)}{g(x)^2} \\
\frac{d}{dx} f(g(x)) &= f'(g(x))g'(x)
\end{aligned}$$

## Useful limits (computed using above rules)

**Example 16.**

$$\lim_{x \rightarrow \infty} \frac{x - 3}{2x + 5} = \frac{1}{2} \quad (1.21)$$

**Example 17.**

$$\lim_{x \rightarrow \infty} \frac{a_n x^n + a_{n-1} x^{n-1} + \dots + a_0}{b_m x^m + b_{m-1} x^{m-1} + \dots + b_0} = \begin{cases} 0 & \text{if } n < m \\ \frac{a_n}{b_m} & \text{if } n = m \\ \infty & \text{if } n > m \end{cases} \quad (1.22)$$

**Example 18.**

$$\lim_{x \rightarrow \infty} \frac{(\log x)^n}{P_m(x)} = 0 \quad (1.23)$$

where  $P_m(x)$  indicates any polynomial in  $x$ .

**Example 19.**

$$\lim_{x \rightarrow \infty} \frac{e^{nx}}{P_m(x)} = \infty \quad (1.24)$$

### 1.3. Sorting functions

The limit

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \quad (1.25)$$

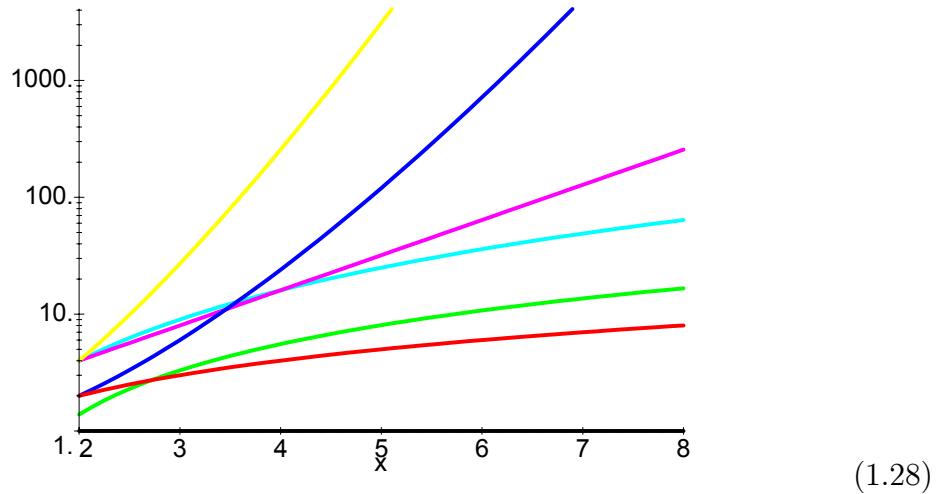
means that  $g(x)$  grows more than  $f(x)$  when  $x \rightarrow \infty$ . This induces a relation between  $f(x)$  and  $g(x)$ :

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \implies f(x) \text{ “grows less than” } g(x) \quad (1.26)$$

We can then sort functions according with their behavior at infinity for example:

$$1 \prec \log x \prec x \prec x \log x \prec x^2 \prec 2^x \prec x! \prec x^x \quad (1.27)$$

where the symbol  $\prec$  in this context reads “grow less than”.



### 1.3.1. Order of growth of functions

#### Definitions

$$\begin{aligned}
 O(g(x)) &\stackrel{\text{def}}{=} \{f(x) : \exists x_0, c_0, \forall x > x_0, 0 \leq f(x) < c_0 g(x)\} \\
 \Omega(g(x)) &\stackrel{\text{def}}{=} \{f(x) : \exists x_0, c_0, \forall x > x_0, 0 \leq c_0 g(x) < f(x)\} \\
 \Theta(g(x)) &\stackrel{\text{def}}{=} O(g(x)) \cap \Omega(g(x)) \\
 o(g(x)) &\stackrel{\text{def}}{=} O(g(x)) - \Omega(g(x)) \\
 \omega(g(x)) &\stackrel{\text{def}}{=} \Omega(g(x)) - O(g(x))
 \end{aligned}$$

#### Intuitive meaning

- $O(g(x))$  = Functions that grow no faster than  $g(x)$  when  $x \rightarrow \infty$
- $\Omega(g(x))$  = Functions that grow no slower than  $g(x)$  when  $x \rightarrow \infty$
- $\Theta(g(x))$  = Functions that grow at the same rate as  $g(x)$  when  $x \rightarrow \infty$
- $o(g(x))$  = Functions that grow slower than  $g(x)$  when  $x \rightarrow \infty$
- $\omega(g(x))$  = Functions that grow faster than  $g(x)$  when  $x \rightarrow \infty$

## Practical rules

1. Compute the limit

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = a \quad (1.29)$$

2. Then look it up on the table

$a$ is positive or zero	$\implies f(x) \in O(g(x)) \Leftrightarrow f \preceq g$
$a$ is positive or infinity	$\implies f(x) \in \Omega(g(x)) \Leftrightarrow f \succeq g$
$a$ is positive	$\implies f(x) \in \Theta(g(x)) \Leftrightarrow f \sim g$
$a$ is zero	$\implies f(x) \in o(g(x)) \Leftrightarrow f \prec g$
$a$ is infinity	$\implies f(x) \in \omega(g(x)) \Leftrightarrow f \succ g$

(1.30)

The rules assume the limits exist. The inverse is not true. For example:

$$f(x) \in \Theta(g(x)) \not\Rightarrow \lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} \text{ is positive} \quad (1.31)$$

**Theorem 20.** Any polynomial  $T(n) = P_m(n)$  of degree  $m$  is  $T(n) \in \Theta(n^m), \in O(n^m)$

**Theorem 21.**  $T_1(n) \in O(f(n))$  and  $T_2 \in O(g(n)) \Rightarrow T_1(n)+T_2(n) \in O(\max(f(n), g(n)))$

**Theorem 22.**  $T_1(n) \in O(f(n))$  and  $T_2 \in O(g(n)) \Rightarrow T_1(n)T_2(n) \in O(f(n)g(n))$

**Example 23.** Is  $x \log x + 3x$  in  $O(x^2)$ ?

$$\lim_{x \rightarrow \infty} \frac{x \log x + 3x}{x^2} \xrightarrow{l'Hopital} \lim_{x \rightarrow \infty} \frac{1/x}{2} = 0 \quad (1.32)$$

answer: Yes

**Example 24.** Is  $x \log x$  in  $\Omega(x^3)$ ?

$$\lim_{x \rightarrow \infty} \frac{x \log x}{x^3} \xrightarrow{l'Hopital^2} \lim_{x \rightarrow \infty} \frac{1/x}{6x} = 0 \quad (1.33)$$

answer: No

## 1.4. Introduction to study of algorithms

### 1.4.1. Maximum and Minimum

Consider the following algorithms which determine respectively the minimum and maximum elements in the input array/list A:

```
def Minimum(A):
    j=0
    for i in range(1,len(A)):
        if A[i]<A[j]:
            j=i
    return A[j]

def Maximum(A):
    j=0
    for i in range(1,len(A)):
        if A[i]>A[j]:
            j=i
    return A[j]
```

Each of these algorithms perform:

- One assignment (line 02)
- $n = \text{len}(A)$  comparisons (line 03)
- $n - 1$  comparisons (line 04)
- line 05 is executed only if line 05 condition is true. In the worst case line 05 is executed  $n - 1$  times.

Therefore in the worst case this program performs  $2n - 1$  comparisons and  $n$  assignments.

We say this program running time goes like

$$T(n) = c_1(2n - 1) + c_2n = (2c_1 + c_2)n - c_1 \quad (1.34)$$

And it is obvious to prove that  $T(n) \in \Theta(n)$ .

### Example: loop0

```
def loop0(n):
    for i in range(0,n):
        print i
```

$$T(n) = \sum_{i=0}^{i < n} 1 = n \in \Theta(n) \Rightarrow \text{loop0} \in \Theta(n)$$

### Example: loop1

```
def loop1(n):
    for i in range(0,n*n):
        print i
```

$$T(n) = \sum_{i=0}^{i < n^2} 1 = n^2 \in \Theta(n^2) \Rightarrow \text{loop1} \in \Theta(n^2)$$

### Example: loop2

```
def loop2(n):
    for i in range(0,n):
        for j in range(0,n):
            print i,j
```

$$T(n) = \sum_{i=0}^{i < n} \sum_{j=0}^{j < n} 1 = \sum_{i=0}^{i < n} n = n^2 + \dots \in \Theta(n^2) \Rightarrow \text{loop2} \in \Theta(n^2)$$

### Example: loop3

```
def loop3(n):
    for i in range(0,n):
        for j in range(0,i):
            print i,j
```

$$T(n) = \sum_{i=0}^{i < n} \sum_{j=0}^{j < i} 1 = \sum_{i=0}^{i < n} i = \frac{1}{2}n(n-1) \in \Theta(n^2) \Rightarrow \text{loop3} \in \Theta(n^2)$$

### Example: loop4

```
def loop4(n):
    for i in range(0,n):
        for j in range(0,i*i):
            print i,j
```

$$\begin{aligned} T(n) &= \sum_{i=0}^{i<n} \sum_{j=0}^{j<i^2} 1 = \sum_{i=0}^{i<n} i^2 = \frac{1}{6}n(n-1)(2n-1) \in \Theta(n^3) \\ \Rightarrow \text{loop4} &\in \Theta(n^3) \end{aligned}$$

### Example: factorial0

```
def factorial0(n):
    prod=1
    for i in range(2,n+1):
        prod=prod*i
    return prod
```

$$T(n) = \sum_{i=1}^{i \leq n} 1 = n \in \Theta(n) \Rightarrow \text{factorial0} \in \Theta(n)$$

### Example: concatenate0

```
def concatenate0(n):
    for i in range(n*n):
        print i
    for j in range(n*n*n):
        print j
```

$$T(n) = \Theta(\max(n^2, n^3)) \Rightarrow \text{concatenate0} \in \Theta(n^3)$$

### Example: concatenate1

```
def concatenate1(n):
    if a<0:
        for i in range(n*n):
            print i
    else:
        for j in range(n*n*n):
            print j
```

$$T(n) = \Theta(\max(n^2, n^3)) \Rightarrow \text{concatenate1} \in \Theta(n^3)$$

### 1.4.2. Insertion Sort

Consider the following algorithm the sorts the elements in the input array/list A:

```
def InsertionSort(A):
    for i in range(1,len(A)):
        j=i-1
        while(j>=0):
            if A[j]>A[j+1]:
                (A[j], A[j+1]) = (A[j+1], A[j])
                j=j-1
            else:
                break
        pass
```

Prove that  $T(n) \in \Theta(n^2)$ .

### 1.4.3. Recurrence relations

The running time  $T(n)$  of recursive algorithms is determined by:

1. Writing the recurrence relation in  $T(n)$  for the algorithm.
2. Solving the recurrence relation or, at least, put a bound on  $T(n)$ .

**In these notes we assume  $f(n)$  is a positive monotonic increasing function,  $a \geq 1$  and  $b \geq 2$ .**

### Recurrence Relations solvable by recursion (type I)

$$T(n) = \begin{cases} c & \text{if } n \leq 1 \\ aT(n-1) + f(n) & \text{if } n > 1 \end{cases} \quad (1.35)$$

The result is:

$$T(n) = a^{n-1} [c - f(1)] + \sum_{i=1}^{i \leq n} a^{n-i} f(i) \quad (1.36)$$

The order of growth of  $T(n)$  can be determined by substituting in the value of  $a$  and the function  $f(n)$ .

Here are two cases of interest:

**Theorem 25.** If  $P_m(n)$  is a polynomial of degree  $m$

$$T(n) = T(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(n^{m+1})$$

**Theorem 26.** If  $P_m(n)$  is a polynomial of degree  $m$  and  $a > 1$

$$T(n) = aT(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(a^n)$$

**Theorem 27.** If  $P_m(n)$  is a polynomial of degree  $m$  and  $a < 1$

$$T(n) = aT(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(n^m)$$

Your are not allowed to use the above three theorems in solving problems. I want to see proofs using eq.(1.36).

### Recurrence Relations solvable using Master Theorem (type II)

$$T(n) = \begin{cases} c & \text{if } n \leq 1 \\ aT(n/b) + f(n) & \text{if } n > 1 \end{cases} \quad (1.37)$$

We do not solve it in  $T(n)$  but we can determine the order of growth of  $T(n)$  by using the following theorem:

## Master Theorem

Given a recurrence relation of the form (1.37) find the best polynomial bound on the function  $f(n)$  in the following way:

Determine the lowest value of  $k \in \mathbb{R}^*$  (0 or positive but not  $\infty$ ) so that

$$\lim_{n \rightarrow \infty} \frac{f(n)}{n^k} = \alpha \in \mathbb{R}^* \text{ (0 or positive real number but not } \infty\text{)}$$

- If there is not such a  $k$  because  $f(n)$  grows more than any polynomial then

$$\lim_{n \rightarrow \infty} \frac{af(n/b)}{f(n)} < 1 \Rightarrow T(n) \in \Theta(f(n)) \quad (1.38)$$

$$\lim_{n \rightarrow \infty} \frac{af(n/b)}{f(n)} \geq 1 \Rightarrow T(n) \in \Omega(f(n)) \quad (1.39)$$

else ...

- If  $\alpha = 0$  for every  $k$  in some range then  $f(n) \in O(n^k)$  and we look up the tighter upper bound in table (1.40) for  $k$  in the range. Else...
- If  $\alpha > 0$  and finite then  $f(n) \in \Theta(n^k)$  and we look up solution in table (1.41).

recurrence type	$T(n) \leq aT(n/b) + f(n)$
case HAVE ONLY UPPER BOUND:	$f(n) \in O(n^k)$
if $a < b^k$ then	$T(n) \in O(n^k)$
if $a = b^k$ then	$T(n) \in O(n^k \log n)$
if $a > b^k$ then	$T(n) \in O(n^{\log_b a})$

(1.40)

recurrence type	$T(n) = aT(n/b) + f(n)$
case HAVE ORDER OF GROWTH:	$f(n) \in \Theta(n^k)$
if $a < b^k$ then	$T(n) \in \Theta(n^k)$
if $a = b^k$ then	$T(n) \in \Theta(n^k \log n)$
if $a > b^k$ then	$T(n) \in \Theta(n^{\log_b a})$

(1.41)

recurrence type	$T(n) \geq aT(n/b) + f(n)$
case HAVE ONLY LOWER BOUND:	$f(n) \in \Omega(n^k)$
if $a < b^k$ then	$T(n) \in \Omega(n^k)$
if $a = b^k$ then	$T(n) \in \Omega(n^k \log n)$
if $a > b^k$ then	$T(n) \in \Omega(n^{\log_b a})$

(1.42)

Note that the only difficult part is finding a polynomial bound on  $f(n)$ , i.e. a value of  $k$ .

We look-up table (1.42) if we are interested in lower bounds.

**Example 28.** Consider the following recurrence relation:

$$T(n) = 2T(n/2) + n$$

from which we read

$$a = b = 2 \text{ and } f(n) = n \in \Theta(n) \Rightarrow k = 1$$

In the table (1.41), corresponding to case  $f(n) \in \Theta(n^k)$ , row 4, corresponding to case  $a = b^k$ , we find the solution:

$$T(n) \in \Theta(n \log n)$$

This is the order of growth of the MergeSort!

**Example 29.** Consider the following recurrence relation:

$$T(n) = T(n/2) + e^n$$

from which we read

$$a = 1, b = 2 \text{ and } f(n) = e^n$$

since  $f(n)$  grows more than polynomially then from (1.38)

$$\lim_{n \rightarrow \infty} \frac{af(n/b)}{f(n)} = \lim_{n \rightarrow \infty} \frac{e^{n/2}}{e^n} = \lim_{n \rightarrow \infty} \left(\frac{1}{\sqrt{e}}\right)^n = \lim_{n \rightarrow \infty} (0.606\dots)^n = 0 < 1$$

we conclude that  $T(n) \in \Theta(e^n)$ .

**Example 30.** Consider the following recurrence relation:

$$T(n) = 3T(n/3) + n \log n$$

from which we read

$$a = b = 3 \text{ and } f(n) = n \log n$$

the best bounds we can put on  $f(x)$  are:  $f(n) \in O(n^2)$  and  $f(n) \in \Omega(n)$ . From them and tables (1.40) and (1.42) respectively we conclude:  $T(n) \in O(n^2)$  and  $T(n) \in \Omega(n \log n)$ . (The true solution is, in fact,  $T(n) \in \Theta(n \log^2 n)$  as stated in the one of the theorems below!)

**Theorem 31.** For  $m \geq 0$ ,  $p \geq 0$  and  $q > 1$ , from the Master Theorem (in its most general form) we conclude that:

$$\begin{aligned} T(n) &= aT(n/b) + \Theta(n^m) \text{ and } a < b^m \Rightarrow T(n) \in \Theta(n^m) \\ T(n) &= aT(n/b) + \Theta(n^m) \text{ and } a = b^m \Rightarrow T(n) \in \Theta(n^m \log n) \\ T(n) &= aT(n/b) + \Theta(n^m) \text{ and } a > b^m \Rightarrow T(n) \in \Theta(n^{\log_b a}) \\ T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a < b^m \Rightarrow T(n) \in \Theta(n^m \log^p n) \\ T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a = b^m \Rightarrow T(n) \in \Theta(n^m \log^{p+1} n) \\ T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a > b^m \Rightarrow T(n) \in \Theta(n^{\log_b a}) \\ T(n) &= aT(n/b) + \Theta(q^n) \Rightarrow T(n) \in \Theta(q^n) \end{aligned}$$

### Recurrence Relations solved by substitution (type III)

For example

$$T(n) = T(b) + T(n - b - a) + c$$

can be solved by guessing  $T(n) = n + a - c$ .

$$\begin{aligned} T(n) &= T(b) + T(n - b - a) + c \\ &= [b + a - c] + [n - b - a + a - c] + c \\ &= n + a - c \end{aligned}$$

from which we conclude  $T(n) \in \Theta(n)$ .

Sometimes our guess is only an inequality for example

$$T(n) = T(n/a - b) + c$$

with  $a > 1, b > 0$ , can be solved by guessing  $T(n) \leq c \log_a n$  in fact

$$\begin{aligned} T(n) &= T(n/a - b) + c \\ &\leq c \log_a(n/a - b) + c \\ &\leq c \log_a(n/a) + a \\ &\leq c \log_a n - c \log_a a + c \\ &= c \log_a n \end{aligned}$$

from which we conclude  $T(n) \in O(\log n)$ .

Note that when making a guess:

$$\begin{aligned} “=” &\rightarrow “\Theta” \\ “\leq” &\rightarrow “O” \\ “\geq” &\rightarrow “\Omega” \end{aligned}$$

### Recurrence Relations that can be reduced (type IV)

Recurrence relations can be very difficult and a general technique for solving them does not exist. Sometimes we try a **substitution method**.

**Example 32.** Consider the following recurrence relation:

$$T(n) = \begin{cases} c & \text{if } n \leq 1 \\ 2T(\sqrt{n}) + \log n & \text{if } n > 1 \end{cases} \quad (1.43)$$

we can replace  $n$  with  $e^k = n$  in eq.(1.43) and obtain:

$$T(e^k) = \begin{cases} c & \text{if } n \leq 1 \\ 2T(e^{k/2}) + k & \text{if } n > 1 \end{cases} \quad (1.44)$$

If we also replace  $T(e^k)$  with  $S(k) = T(e^k)$  we obtain:

$$\underbrace{S(k)}_{T(e^k)} = \begin{cases} c & \text{if } n \leq 1 \\ 2 \underbrace{S(k/2)}_{T(e^{k/2})} + k & \text{if } n > 1 \end{cases} \quad (1.45)$$

so that we can now apply the Master Theorem to  $S$ . We obtain that  $S(k) \in \Theta(k \log k)$ . Once we have the order of growth of  $S$  we can determine the order of growth of  $T(n)$  by substitution:

$$T(n) = S(\log n) \in \Theta(\underbrace{\log n}_k \log \underbrace{\log n}_k) \quad (1.46)$$

Note that there are recurrence relations that cannot be solved with any of the methods described above.

**Example 33.** Consider the following recurrence relation:

$$T(n) = \begin{cases} c & \text{if } n \leq 1 \\ T(n-b) + n^2 & \text{if } n > 1 \end{cases} \quad (1.47)$$

we can replace  $n$  with  $bk = n$  in eq.(1.43) and obtain:

$$T(bk) = \begin{cases} c & \text{if } n \leq 1 \\ T(bk-b) + b^2k^2 & \text{if } n > 1 \end{cases} \quad (1.48)$$

If we also replace  $T(bk)$  with  $S(k) = T(bk)$  we obtain:

$$\underbrace{S(k)}_{T(bk)} = \begin{cases} c & \text{if } n \leq 1 \\ \underbrace{S(k-1)}_{T(bk-b)} + b^2k^2 & \text{if } n > 1 \end{cases} \quad (1.49)$$

which has solution

$$S(k) = (c - b^2) + \sum_{i=1}^{i \leq k} b^2 i^2 = (c - b^2) + \frac{1}{6} b^2 k(k+1)(2k+1) \in \Theta(k^3)$$

therefore

$$T(n) = T(bk) = S(k) \in \Theta(k^3) = \Theta(n^3/b^3) = \Theta(n^3)$$

### Other types of recurrence relations (type V)

There are many types of recurrence relations and many of them do not fit in the categories listed above. They are not covered in this class.

## Useful theorems about recurrence relations

$$\begin{aligned} O(f(n)) + \Theta(g(n)) &= \Theta(g(n)) \text{ iff } f(n) \in O(g(n)) \\ \Theta(f(n)) + \Theta(g(n)) &= \Theta(g(n)) \text{ iff } f(n) \in O(g(n)) \\ \Omega(f(n)) + \Theta(g(n)) &= \Omega(f(n)) \text{ iff } f(n) \in \Omega(g(n)) \end{aligned}$$

How to apply:

$$T(n) = [\underbrace{n^2 + n + 3}_{\Theta(n^2)} + \underbrace{e^n - \log n}_{\Theta(e^n)}] \in \Theta(e^n) \text{ because } n^2 \in O(e^n)$$

### Example: factorial1

```
def factorial1(n):
    if n==0:
        return 1
    else:
        return n*factorial1(n-1)
```

$$T(n) = T(n-1) + 1 \Rightarrow T(n) \in \Theta(n) \Rightarrow \text{factorial1} \in \Theta(n)$$

### Example: recursive0

```
def recursive0(n):
    if n==0:
        return 1
    else:
        loop3(n)
        return n*n*recursive0(n-1)
```

$$T(n) = T(n-1) + P_2(n) \Rightarrow T(n) \in \Theta(n^2) \Rightarrow \text{recursive0} \in \Theta(n^3)$$

### Example: recursive1

```
def recursive1(n):
    if n==0:
        return 1
    else:
        loop3(n)
        return n*recursive1(n-1)*recursive1(n-1)
```

$$T(n) = 2T(n - 1) + P_2(n) \Rightarrow T(n) \in \Theta(2^n) \Rightarrow \text{recursive1} \in \Theta(2^n)$$

### Example: recursive2

```
def recursive2(n):
    if n==0:
        return 1
    else:
        a=factorial0(n)
        return a*recursive2(n/2)*recursive1(n/2)
```

$$T(n) = 2T(n/2) + P_1(n) \Rightarrow T(n) \in \Theta(n \log n) \Rightarrow \text{recursive2} \in \Theta(n \log n)$$

### Example: binarySearch

```
def binary_search(array,x):
    low=0
    high=len(array)-1
    while high>=low:
        mid=(high+low)/2
        if array[mid]<x:
            low=mid+1
        elif array[mid]>x:
            high=mid-1
        else:
            return mid
    return None
```

$$T(n) = T(n/2) + 1 \Rightarrow \text{binarySearch} \in O(\log n)$$

#### 1.4.4. MergeSort

Consider the following sorting algorithm:

```
def Merge(A,p,q,r):
    B=[]
    i=p
    j=q
    while true:
        if A[i]<=A[j]:
            B.append(A[i])
            i=i+1
        else:
            B.append(A[j])
            j=j+1
        if i==q:
            while j<r:
                B.append(A[j])
                j=j+1
            break
        if j==r:
            while i<q:
                B.append(A[i])
                i=i+1
            break
    A[p:r]=B

def MergeSort(A, p=0, r=-1):
    if r is -1:
        r=len(A)
    if p<r-1:
        q=int((p+r)/2)
        MergeSort(A,p,q)
        MergeSort(A,q,r)
        Merge(A,p,q,r)
```

The MergeSort follows a **Divide-and-Conquer** approach (see definition below). A Divide-and-conquer algorithm is a recursive algorithm and running time

of recursive algorithms is determined by solving the corresponding recurrence relation.

Let assume that the running time of the MergeSort is  $T(n)$  where  $n = \text{len}(A)$ . It is easy to prove that  $T_{\text{Merge}}(n) = cn$  since the Merge function contains only one loop. Therefore at each step

$$T(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2T(n/2) + T_{\text{Merge}}(n) & \text{if } n > 1 \end{cases} \quad (1.50)$$

#### 1.4.5. Order of growth of algorithms

Given an algorithm,  $A$ , we can determine three characteristics functions:

- $T_{\text{worst}}(n)$ : the running time in the worst case
- $T_{\text{best}}(n)$ : the running time in the best case
- $T_{\text{average}}(n)$ : the running time in the average case

If  $T_{\text{worst}}(n) \in O(f(n))$  or  $T_{\text{worst}}(n) \in \Theta(f(n))$  we say that the algorithms  $A$  is  $O(f(n))$ .

If  $T_{\text{best}}(n) \in \Omega(f(n))$  or  $T_{\text{best}}(n) \in \Theta(f(n))$  we say that the algorithms  $A$  is  $\Omega(f(n))$ .

If the algorithm  $A$  is  $O(f(n))$  and is  $\Omega(f(n))$  we say that  $A$  is  $\Theta(f(n))$ .

The order of growth in the average case is a very important measure of the efficiency of the algorithm but it does enter in the definitions above.

## 2. DATA STRUCTURES AND ALGORITHMS

### 2.1. CSC321 program

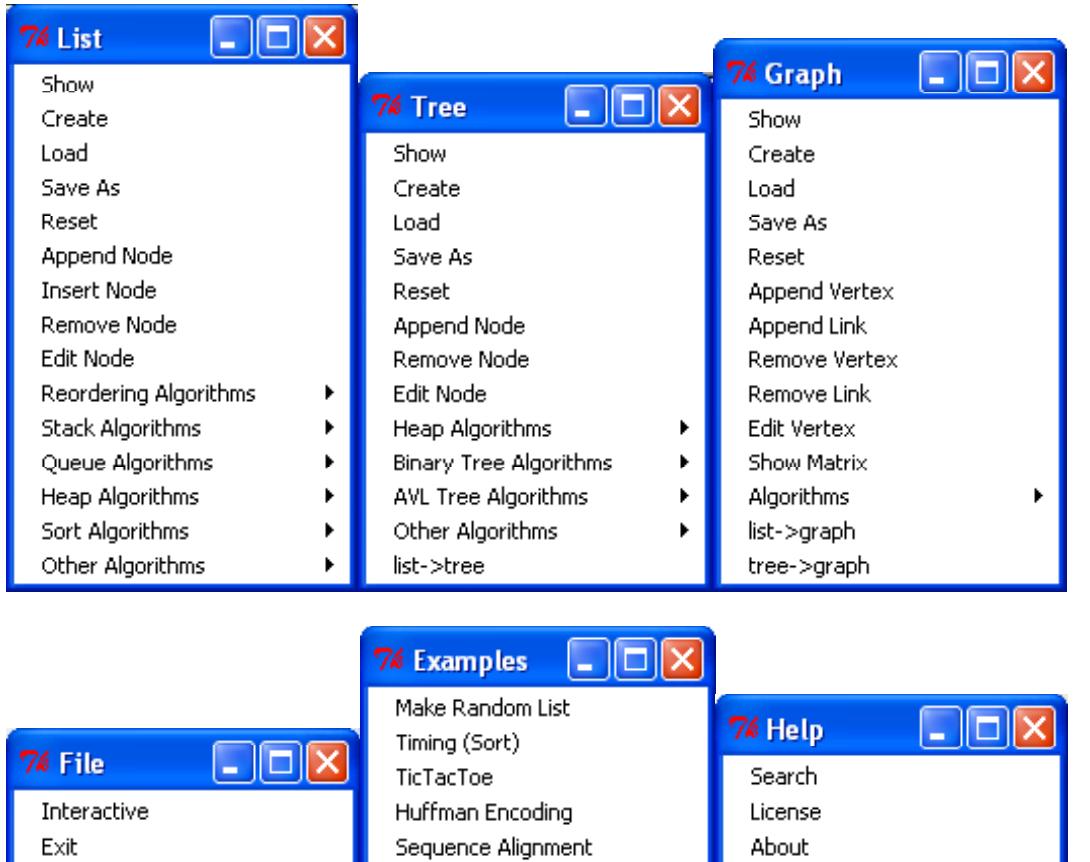
The CSC321 program can be downloaded from:

<http://www.phoenixcollective.org/mdp/csc321.exe>

The CSC321 program is implemented in Python and based on the Tkinter graphic library and the Python Mega Widgets (Pmw). Python is an interpreted language invented by Guido van Rossum. This language was chosen because its syntax closely resembles the syntax commonly used to write pseudo-codes. Moreover Python includes, as basic types, structures such as tuples, lists and dictionaries. The version of the program distributed with these notes is packaged for Windows. Versions for other platforms are available upon request.



The main menus are



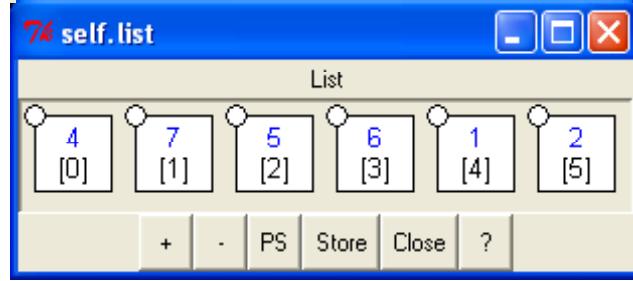
(2.1)

### 2.1.1. Lists

To start creating a list choose [List][Create] and type a Python list, for example:

```
[4, 7, 5, 6, 1, 2]
```

which is displayed as



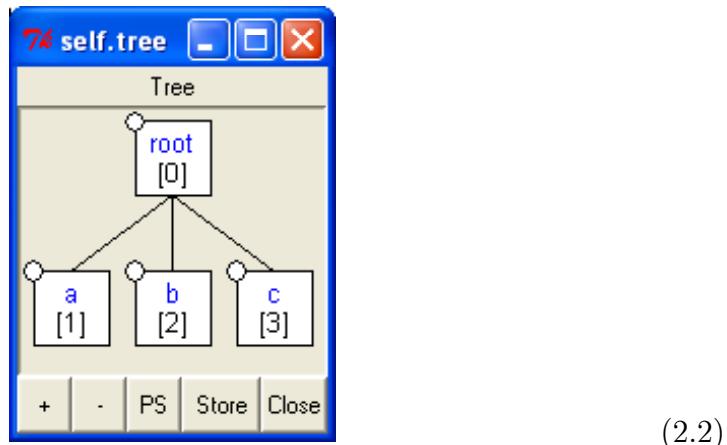
Note that each Node (in a list or a tree or a graph) can have different attributes: a **value**, a **name**, a **color**, etc. By default, when a list, a tree or a graph is created, only the value of the Nodes is specified ('root', 'a', 'b', 'c', 3, 4, 5). One should not confuse the value of a Node (for example 4 in the example) with its **location** ([0] in the example). The location is an index number which is automatically generated and is unique for each Node in a list, tree or graph. While the value of a node is not necessarily unique, the location index is unique.

### 2.1.2. Trees

In CSC321 a tree is implemented as a list of lists. For example, create the following list:

`['root', 'a', 'b', 'c']`

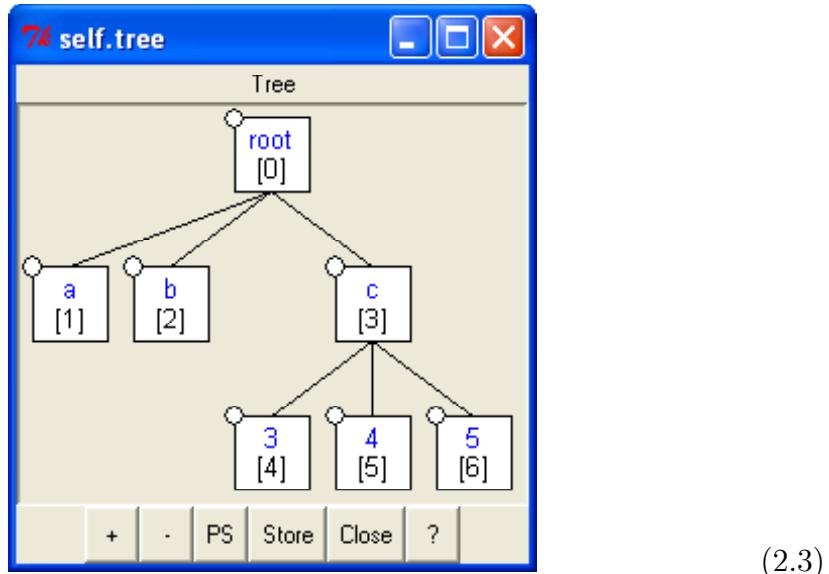
which is displayed as:



To add children Nodes to the Node c, one may create a new list where Node c itself is replaced by a subtree. For example, create the following list:

```
'root', 'a', 'b', ['c', 3, 4, 5]]
```

which is displayed as:



In general each Node in a tree can have an arbitrary number of children. It is possible to specialize a tree by imposing some constraints on it. In this course we examine three types of special trees:

- **Heaps**
- **Binary Trees**
- **AVL Tree**

### 2.1.3. Graphs

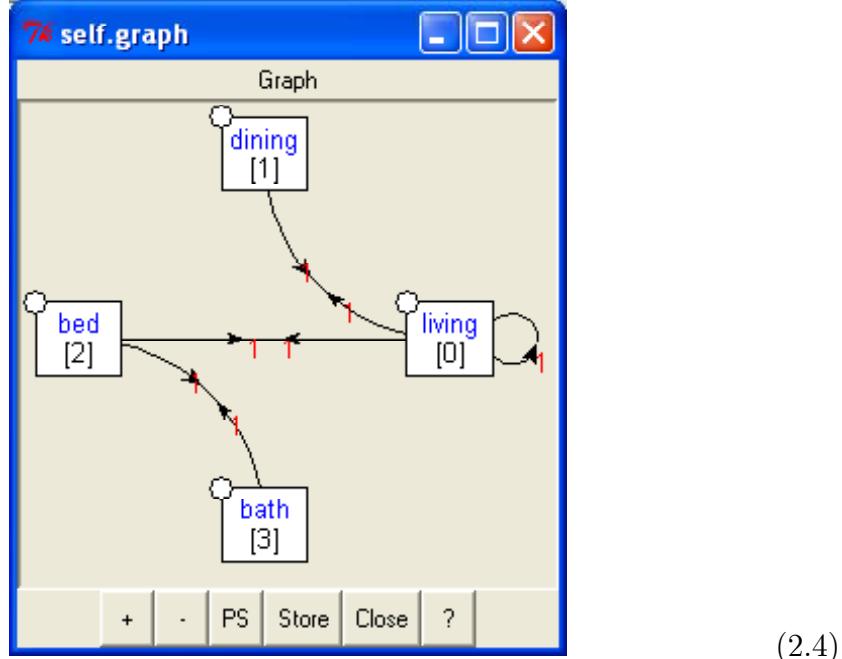
A graph is a collection of Nodes (with the same attributes as List Nodes and Tree Nodes) plus a collection of links between Nodes. In general links may have directions (for example Node [0] may be linked to Node [1] but not vice versa). Moreover each Node may be linked to itself (Node [0] may be linked to Node[0]).

In CSC321 a graph is implemented as a List of Lists where the element 0 of each sub-list is the value of the corresponding Node. The other elements of each sub-list are the location indices of the Nodes connected to the Node itself (**Adjacency List Representation**).

To start creating a graph choose [**Graph**][**Create**] and type a graph, for example:

```
[['living','dining','bed','bath'], [[0,0], [0,1], [1,0], [0,2], [2,0], [2,3], [3,2]]]
```

that represents the topology of a one bedroom apartment. This graph is displayed as:



A graphs is represented as two lists: a list of vertices/nodes (for example:  $V = ['living', 'dining', 'bed', 'bath']$ ) and a list of links.

- The living room (Node [0]) is connected to itself [0,0] (has an island kitchen?) to the dining room [0,1] and to the bedroom [0,2].
- The dining room (Node [1]) is connected only to the living room [1,0].

- The bedroom (Node [2]) is connected to the living room [2,0] and the bathroom [2,3].
- The bathroom (Node [3]) is connected to the bedroom only [3,2].

In the representation of the graph the number 1 attached to each link represents the length of the link (this is one by default).

## 2.2. Algorithms

### 2.2.1. Algorithm types: definitions

**Definition 34.** *The Divide-and-Conquer is a method of designing algorithms that (informally) proceeds as follows: Given an instance of the problem to be solved, split this into several, smaller, sub-instances (of the same problem) independently solve each of the sub-instances and then combine the sub-instance solutions so as to yield a solution for the original instance. This description raises the question: By what methods are the sub-instances to be independently solved? The answer to this question is central to the concept of the Divide-and-Conquer algorithm and is a key factor in gauging their efficiency. The solution depends on the problem.*

**Definition 35. Dynamic Programming** is a paradigm that is most often applied in the construction of algorithms to solve a certain class of optimization problems. That is problems which require the minimization or maximization of some measure. One disadvantage of using Divide-and-Conquer is that the process of recursively solving separate sub-instances can result in the same computations being performed repeatedly since identical sub-instances may arise. The idea behind dynamic programming is to avoid this pathology by obviating the requirement to calculate the same quantity twice. The method usually accomplishes this by maintaining a table of sub-instance results. We say that Dynamic Programming is a Bottom-Up technique in which the smallest sub-instances are explicitly solved first and the results of these used to construct solutions to progressively larger sub-instances. In contrast, we say that the Divide-and-Conquer is a Top-Down technique. One case of Top-Down approach, called **memoization**, that is normally included under the umbrella of Dynamic Programming. The reason is that the memoization approach uses tables to store intermediate results.

**Definition 36.** Greedy algorithms work in phases. In each phase, a decision is made that appears to be good, without regard for future consequences. Generally, this means that some local optimum is chosen. This ‘take what you can get now’ strategy is the source of the name for this class of algorithms. When the algorithm terminates, we hope that the local optimum is equal to the global optimum. If this is the case, then the algorithm is correct; otherwise, the algorithm has produced a suboptimal solution. If the best answer is not required, then simple greedy algorithms are sometimes used to generate approximate answers, rather than using the more complicated algorithms generally required to generate an exact answer. Even for problems which can be solved exactly by a greedy algorithm, establishing the correctness of the method may be a non-trivial process.

There are other types of algorithms that do not follow in any of the above categories. One is, for example, backtracking. Backtracking is not covered in this course.

### 2.2.2. Sorting algorithms

The  $T_{memory}$  in the following paragraph is the size of the extra memory required by the algorithm (i.e. space taken by local variables) and it does not include space for the input.

#### InsertionSort (divide and conquer)

$$\begin{aligned} T_{best} &\in \Theta(n) \\ T_{average} &\in \Theta(n^2) \\ T_{worst} &\in \Theta(n^2) \\ T_{memory} &\in \Theta(1) \end{aligned}$$

#### MergeSort (divide and conquer)

$$\begin{aligned} T_{best} &\in \Theta(n \log n) \\ T_{average} &\in \Theta(n \log n) \\ T_{worst} &\in \Theta(n \log n) \\ T_{memory} &\in \Theta(1) \end{aligned}$$

## MergeSortDP (dynamic programming)

```
def MergeSortDP(A):
    blocksize=1
    listszie=len(A)
    while blocksize<listszie:
        for p in range(0, listszie, 2*blocksize):
            q=p+blocksize
            r=min(q+blocksize, listszie)
            if r>q:
                Merge(A,p,q,r)
        blocksize=2*blocksize
```

$$\begin{aligned} T_{best} &\in \Theta(n \log n) \\ T_{average} &\in \Theta(n \log n) \\ T_{worst} &\in \Theta(n \log n) \\ T_{memory} &\in \Theta(1) \end{aligned}$$

## QuickSort (divide and conquer)

There are at least 2 possible implementations of partition and quicksort. The following:

```
def Partition(A,i,j):
    x=A[i]
    while true:
        j=j-1
        while A[j]>x:
            j=j-1
        while A[i]<x:
            i=i+1
        if i<j:
            (A[i],A[j])=(A[j],A[i])
        else:
            return j+1
```

```

def QuickSort(A,p=0,r=-1):
    if r is -1:
        r=len(A)
    if p<r-1:
        q=Partition(A,p,r)
        QuickSort(A,p,q)
        QuickSort(A,q,r)

```

and the following, based on the so called Lomuto partition:

```

def Partition(A,i,j):
    x=A[i]
    h=i
    for k in range(i+1,j):
        if A[k]<x:
            h=h+1
            A[h],A[k]=A[k],A[h]
    A[h],A[i]=A[i],A[h]
    return h

def QuickSort(A,p=0,r=-1):
    if r is -1:
        r=len(A)
    if p<r-1:
        q=Partition(A,p,r)
        QuickSort(A,p,q)
        QuickSort(A,q+1,r)

```

In both cases:

$$\begin{aligned}
T_{best} &\in \Theta(n \log n) \\
T_{average} &\in \Theta(n \log n) \\
T_{worst} &\in \Theta(n^2) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

## RandomizedQuickSort

The Randomized partition is similar to partition but uses a random element of the array  $A[p \leq i < r]$  as pivot.

```
def RandomizedPartition(A,p,r):
    i=randint(p,r-1)
    (A[p],A[i])=(A[i],A[p])
    return Partition(A,p,r)

def RandomizedQuickSort(A,p=0,r=-1):
    if r is -1:
        r=len(A)
    if p<r-1:
        q=RandomizedPartition(A,p,r)
        RandomizedQuickSort(A,p,q)
        RandomizedQuickSort(A,q+1,r)
```

$$\begin{aligned} T_{best} &\in \Theta(n \log n) \\ T_{average} &\in \Theta(n \log n) \\ T_{worst} &\in \Theta(n^2) \\ T_{memory} &\in \Theta(1) \end{aligned}$$

## HeapSort

**Definition 37.** A **leaf** is a node of a tree with no children nodes.

**Definition 38.** A **binary tree** is a tree where each node has no more than two children.

**Definition 39.** A **heap** is a special kind of binary tree. It has two properties that are not generally true for other trees:

**Completeness:** The tree is complete, which means that nodes are added from top to bottom, left to right, without leaving any spaces.

**Heapness:** The item in the tree with the highest value is at the top of the tree, and the same is true for every subtree.

A Heap is defined by the above relations:

```
def Parent(i):
    return int((i-1)/2)
```

```
def Left(i):
    return 2*i+1
```

```
def Right(i):
    return 2*i+2
```

Heaps have two important applications: HeapSort and priority queues (see later)

```
def Heapify(A,i,heapsize=-1):
    if heapsize<0:
        heapsize=len(A)
    left=2*i+1
    right=2*i+2
    if left<heapsize and A[left]>A[i]:
        largest=left
    else:
        largest=i
    if right<heapsize and A[right]>A[largest]:
        largest=right
    if largest!=i:
        (A[i], A[largest])=(A[largest], A[i])
        Heapify(A,largest,heapsize)
```

```
def BuildHeap(A):
    heapsize=len(A)
    i=int(heapsize/2)
    while i>0:
        i=i-1
        Heapify(A,i)
```

```
def HeapSort(A):
    BuildHeap(A)
```

```

heapsize=len(A)
i=heapsize
while i>1:
    i=i-1
    (A[0],A[i])=(A[i],A[0])
    heapsize=heapsize-1
    Heapify(A,0,heapsize)

```

$$\begin{aligned}
T_{best} &\in \Theta(n) \\
T_{average} &\in \Theta(n \log n) \\
T_{worst} &\in \Theta(n \log n) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

## CountingSort

The CountingSort algorithm is the only algorithm, among the ones considered here, that uses the array values  $A[i]$  as index for another array  $C[A[i]]$ . For this reason this algorithm only works if the elements to be sorted are integers but it presents the advantage of running in linear time in the range spanned by the array/list elements  $k$ . The implementation below assumes elements are positive integers.

```

def CountingSortSmall(A):
    if Minimum(A)<0:
        raise 'CountingSort List Unbound'
    n=len(A)
    C=[]
    k=Maximum(A)+1
    for j in range(k):
        C.append(0)
    for j in range(n):
        C[A[j]]=C[A[j]]+1
    i=0
    for j in range(k):
        while C[j]>0:
            A[i]=j
            C[j]-=1

```

```
C[j]=C[j]-1
i=i+1
```

Another alternative implementation is the following:

```
def CountingSort(A):
    if Minimum(A)<0:
        raise 'CountingSort List Unbound'
    B=[]
    C=[]
    k=Maximum(A)+1
    for j in range(k):
        C.append(0)
    for j in range(len(A)):
        B.append(0)
        C[A[j]]=C[A[j]]+1
    for i in range(1,k):
        C[i]=C[i]+C[i-1]
    j=len(A)-1
    while j>=0:
        B[C[A[j]]-1]=A[j]
        C[A[j]]=C[A[j]]-1
        j=j-1
    return B
```

If we define  $k = \max(A) - \min(A) + 1$  and  $n = \text{len}(A)$  we see:

$$\begin{aligned} T_{best} &\in \Theta(k + n) \\ T_{average} &\in \Theta(k + n) \\ T_{worst} &\in \Theta(k + n) \\ T_{memory} &\in \Theta(k + n) \end{aligned}$$

### 2.2.3. Searching algorithms

#### Priorities queues

A priority queue is a container implemented as a Heap. We can put items in the priority queue and we can easily extract the item with maximum value (the top of the heap).

```

def HeapExtractMax(A):
    if len(A)<1:
        raise 'Heap Underflow'
    max=A[0]
    A[0]=A[len(A)-1]
    del A[len(A)-1]
    Heapify(A,0)
    return max

```

$$\begin{aligned}
T_{best} &\in \Theta(1) \\
T_{average} &\in \Theta(\log n) \\
T_{worst} &\in \Theta(\log n) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

```

def HeapInsert(A,node):
    A.append(node)
    i=len(A)-1
    while i>0 and A[Parent(i)]<A[i]:
        (A[i],A[Parent(i)])=(A[Parent(i)],A[i])
        i=Parent(i)

```

$$\begin{aligned}
T_{best} &\in \Theta(1) \\
T_{average} &\in \Theta(\log n) \\
T_{worst} &\in \Theta(\log n) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

## Binary search trees

A binary search tree is a binary tree and, for each node, the values of left descendant nodes are lower than the value of parent node and the values of the right descendant nodes are higher than the value of the parent node. A binary search tree is a container that allows us to insert items and seach them rapidly,  $O(\log n)$ , accordingly with their values.

```

def InorderTreeWalk(tree,list=[]):
    ifisNullTree(tree):
        return
    InorderTreeWalk(tree[leftchild],list)
    list.append(tree[rootnode])
    InorderTreeWalk(tree[rightchild],list)
    return list

def TreeSearch(tree, k):
    ifisNullTree(tree):
        return
    if k < tree[rootnode]:
        return TreeSearch(tree[leftchild],k)
    elif k > tree[rootnode]:
        return TreeSearch(tree[rightchild],k)
    else:
        return tree[rootnode]

def TreeMinimum(tree):
    while notisNullTree(tree[leftchild]):
        tree=tree[leftchild]
    return tree

def TreeMaximum(tree):
    while notisNullTree(tree[rightchild]):
        tree=tree[rightchild]
    return tree

```

Each of these algorithms goes like:

$$\begin{aligned}
 T_{best} &\in \Theta(1) \\
 T_{average} &\in \Theta(\log n) \\
 T_{worst} &\in \Theta(n) \\
 T_{memory} &\in \Theta(1)
 \end{aligned}$$

```

def TreeInsertLeaf(x, k):
    ifisNullTree(x):

```

```

        k.parent=None
        x[:]=BinaryTree(k)
    else:
        k.parent=x
        if k<x[rootnode]:
            x[leftchild]=BinaryTree(k)
        else:
            x[rightchild]=BinaryTree(k)

def TreeInsert(tree, k):
    ifisNullTree(tree):
        TreeInsertLeaf(tree,k)
        return tree
    x=tree
    while not isNullTree(x):
        y=x
        if k < x[rootnode]:
            x=x[leftchild]
        elif k > x[rootnode]:
            x=x[rightchild]
        else:
            return tree
        x=y
    TreeInsertLeaf(x, k)
    return tree

```

The TreeInsert algorithms grows like:

$$\begin{aligned}
 T_{best} &\in \Theta(1) \\
 T_{average} &\in \Theta(\log n) \\
 T_{worst} &\in \Theta(n) \\
 T_{memory} &\in \Theta(1)
 \end{aligned}$$

```

def TreeDelete(z):
    if isNullTree(z[leftchild]) and isNullTree(z[rightchild]):
        TreeLeafDelete(z)
    elif isNullTree(z[leftchild]):

```

```

x=z[rightchild]
TreeNodeReplaceWithChildren(z,x)
elifisNullTree(z[rightchild]):
    x=z[leftchild]
    TreeNodeReplaceWithChildren(z,x)
else:
    x=TreeMinimum(z[rightchild])
    TreeNodeReplaceWithoutChildren(z,x)

```

$$\begin{aligned}
T_{best} &\in \Theta(1) \\
T_{average} &\in \Theta(\log n) \\
T_{worst} &\in \Theta(n) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

### AVL trees (balanced binary trees)

AVL trees are binary search trees that are rebalanced after each insertion/deletion. They are rebalanced in such a way that for each node the height of the left subtree minus height of the right subtree is -1,0 or +1. The rebalance operation can be done in  $O(\log n)$ .

$$\begin{aligned}
T_{best} &\in \Theta(1) \\
T_{average} &\in \Theta(\log n) \\
T_{worst} &\in \Theta(\log n) \\
T_{memory} &\in \Theta(1)
\end{aligned}$$

### k-trees, B-trees and Red-black trees

Until now we have considered binary trees (each node has 2 children and store 1 value). One can generalize this to  $k$ trees where each node has  $k$  children and store more than one value.

#### 2.2.4. Graph algorithms

**Definition 40.** A **graph**  $G$  is a pair of sets  $(V, E)$  where  $V$  is a set of vertices (or **Nodes**) and  $E$  a set of **links** between the couple of vertices.

**Definition 41.** In general a **link**, indicated with the notation  $e_{ij}$ , connecting vertex  $i$  with vertex  $j$  is called a **directed link**. If the link has no direction  $e_{ij} = e_{ji}$  it is called an **undirected link**. A graph that contains only undirected links is an **undirected graph**, otherwise it is a **directed graph**.

**Definition 42.** A **walk** is an alternating sequence of vertices and links, with each link being incident to the vertices immediately preceding and succeeding it in the sequence. A **trail** is a walk with no repeated links.

**Definition 43.** A **path** is a walk with no repeated vertices. A walk is **closed** if the initial vertex is also the terminal vertex.

**Definition 44.** A **cycle** is a closed trail with at least one edge and with no repeated vertices except that the initial vertex is the terminal vertex.

**Definition 45.** A graph that contains no cycles is an **acyclic graph**. Any connected acyclic undirected graph is also a **tree**.

**Definition 46.** A **loop** is a one link path connecting a vertex with itself.

**Definition 47.** A non-null graph is **connected** if, for every pair of vertices, there is a walk whose ends are the given vertices. Let us write  $i \sim j$  if there is path from  $i$  to  $j$ . Then  $\sim$  is an equivalence relation. The equivalence classes under  $\sim$  are the vertex sets of the connected components of  $G$ . A connected graph is therefore a graph with exactly one connected component.

**Definition 48.** A graph is called **complete** when every pair of vertices is connected by a link (or edge).

**Definition 49.** A **clique** of a graph is a subset of vertices in which every pair is an edge.

**Definition 50.** The **degree** of a vertex of a graph is the number of edges incident to it.

**Definition 51.** If  $i$  and  $j$  are vertices, the **distance** from  $i$  to  $j$ , written  $d_{ij}$ , is the minimum length of any path from  $i$  to  $j$ . In an undirected graph, this induces a metric.

**Definition 52.** The **eccentricity**,  $e(i)$ , of the vertex  $i$  is the maximum value of  $d_{ij}$ , where  $j$  is allowed to range over all of the vertices of the graph.

**Definition 53.** The **subgraph** of  $G$  induced by a subset  $W$  of its vertices  $V$  ( $W \subseteq V$ ) is the graph formed by the vertices in  $W$  and all edges whose two endpoints are in  $W$ .

### Breadth first search

```
def Adjacents(u,graph):
    vertices=graph[0]
    links=graph[1]
    a=[]
    for link in links:
        if vertices[link.source] is u:
            a.append(link)
    return a

def BreadthFirstSearch(graph,start):
    vertices=graph[0]
    links=graph[1]
    blacknodes=[]
    graynodes=[start]
    t=0
    vertices[start].time_discovered=t
    while len(graynodes)>0:
        k=Dequeue(graynodes)
        for link in Adjacents(vertices[k],graph):
            j=link.destination
            if not j in blacknodes and not j in graynodes:
                Enqueue(graynodes,j)
                vertices[j].time_discovered=t
        blacknodes.append(k)
```

```

    vertices[k].time_found=
    t=t+1
    return blacknodes

```

The function `Adjacents(vertex,graph)` returns a list of links starting from vertex. The present implementation of the `Adjacents` function is not very efficient (sorry!). The BFS algorithm goes like:

$$\begin{aligned}
 T_{best} &\in \Theta(n_E + n_V) \\
 T_{average} &\in \Theta(n_E + n_V) \\
 T_{worst} &\in \Theta(n_E + n_V) \\
 T_{memory} &\in \Theta(1)
 \end{aligned}$$

### Depth first search

```

def DepthFirstSearch(graph,start):
    vertices=graph[0]
    links=graph[1]
    blacknodes=[]
    graynodes=[start]
    t=0
    vertices[start].time_discovered=t
    while len(graynodes)>0:
        k=Pop(graynodes)
        for link in Adjacents(vertices[k],graph):
            j=link.destination
            if not j in blacknodes and not j in graynodes:
                Push(graynodes,j)
                vertices[j].time_discovered=t
        blacknodes.append(k)
        vertices[k].time_found=t
        t=t+1
    return blacknodes

```

The DFS algorithm goes like:

$$T_{best} \in \Theta(n_E + n_V)$$

$$\begin{aligned}
 T_{average} &\in \Theta(n_E + n_V) \\
 T_{worst} &\in \Theta(n_E + n_V) \\
 T_{memory} &\in \Theta(1)
 \end{aligned}$$

### Minimum spanning tree: Kruskal

```

def FindSet(S,i):
    for j in range(len(S)):
        if i in S[j]:
            return j
    return None

def UnionSets(S,i,j):
    for k in range(len(S)):
        if i in S[k]:
            A=S[k]
            break
    del S[k]
    for k in range(len(S)):
        if j in S[k]:
            S[k]=S[k]+A
            break
    return

def MSTKruskal(graph):
    vertices=graph[0]
    links=graph[1]
    A=[]
    S=[]
    for i in range(len(vertices)):
        S.append([i])
    links.sort()
    for link in links:
        i=link.source
        j=link.destination
        if FindSet(S,i)!=FindSet(S,j):

```

```

A=A+[(i,j)]
UnionSets(S,i,j)
return A

```

Functions FindSet and UnionsSets are described in the book (section 21.1)  
The Kruskal algorithm goes like:

$$\begin{aligned}
T_{worst} &\in \Theta(n_E \log n_V) \\
T_{memory} &\in \Theta(n_E)
\end{aligned}$$

### Minimum spanning tree: Prim

```

def ExtractMin(Q):
    i=0
    for j in range(1,len(Q)):
        if Q[j].d<Q[i].d or Q[i].d==Infinity:
            i=j
    u=Q[i]
    del Q[i]
    return u

def MSTPrim(graph, r):
    vertices=graph[0]
    links=graph[1]
    Q=[]
    for i in range(len(vertices)):
        vertices[i].d=Infinity
        Q.append(vertices[i])
    Q[r].parent=None
    Q[r].d=0

    while len(Q)>0:
        u=ExtractMin(Q)
        for link in Adjacents(u,graph):
            v=vertices[link.destination]
            w=link.length
            if v in Q and w<v.d:

```

```

v.parent=u
v.d=w

```

For maximum speed Q should be implemented as a priority queue or, better, as a Fibonacci heap (not covered). Here adopted a slower implementation.

The Prim algorithm, when using a priority queue for Q, goes like:

$$\begin{aligned}
T_{worst} &\in \Theta(n_E + n_V \log n_V) \\
T_{memory} &\in \Theta(n_E)
\end{aligned}$$

### Single source shortest path: Dijkstra

```

def Dijkstra(graph, r):
    vertices=graph[0]
    links=graph[1]
    Q=[]
    for i in range(len(vertices)):
        vertices[i].d=Infinity
        Q.append(vertices[i])
    Q[r].parent=None
    Q[r].d=0

    while len(Q)>0:
        u=ExtractMin(Q)
        for link in Adjacents(u,graph):
            v=vertices[link.destination]
            w=link.length
            if v in Q and w+u.d<v.d:
                v.parent=u
                v.d=w+u.d

```

The Dijkstra algorithm goes like:

$$\begin{aligned}
T_{worst} &\in \Theta(n_E + n_V \log n_V) \\
T_{memory} &\in \Theta(n_E)
\end{aligned}$$

### Single source shortest path: Bellman-Ford

```
def InitializeSingleSource(graph,s):
    vertices=graph[0]
    for vertex in vertices:
        vertex.d=Infinity
        vertex.parent=None
    vertices[s].d=0

def Relax(u,v,graph):
    vertices=graph[0]
    links=graph[1]
    for link in Adjacents(u,graph):
        if v is vertices[link.destination]:
            w=link.length
            if not u.d is Infinity:
                if v.d>u.d+w:
                    v.d=u.d+w
                    v.parent=u
    return

def BellmanFord(graph, s):
    vertices=graph[0]
    links=graph[1]
    InitializeSingleSource(graph,s)
    for i in range(1,len(vertices)-1):
        for link in links:
            u=vertices[link.source]
            v=vertices[link.destination]
            Relax(u,v,graph)
    for link in links:
        u=vertices[link.source]
        v=vertices[link.destination]
        w=link.length
        if u.d is Infinity:
            d=1e10
        else:
```

```

d=u.d+w
if v.d>d:
    return false
return true

```

The Bellman-Ford algorithm goes like:

$$\begin{aligned} T_{worst} &\in \Theta(n_E n_V) \\ T_{memory} &\in \Theta(1) \end{aligned}$$

### 2.2.5. More examples

#### Fibonacci number and memoization

Let's consider a divide-and-conquer approach (recursive, top-down) to the problem of determining fibonacci numbers:

```

def FibonacciRecursive(n):
    print 'calling FibonacciRecursive(%i)' % (n)
    if n is 0 or n is 1:
        return 1
    return FibonacciRecursive(n-1)+FibonacciRecursive(n-2)

```

This solution works but it is very slow because the function `FibonacciRecursive` calls itself twice and the two branches contain common subproblems. Therefore the same subproblems are solved multiple times. The fact that a problem, for example `FibonacccyRecursive(3)`, contains solution of subproblems, for example `FibonacciRecursive(2)`, is referred to as **optimal substructure**. If a problem has optimal substructure then we can use a dynamic programming approach:

```

def FibonacciDynamicProgramming(n):
    print 'calling FibonacciDynamicProgramming(%i)' % (n)
    if n is 0 or n is 1:
        return 1
    a=1
    b=1;
    for i in range(1,n):
        (a,b) = (b, a+b)
        print '    fibonacci(',i+1,')=', b
    return b

```

Sometime it is convenient, easier, to mantain a top-down recursive approach and store in a table the subproblems that have been solved already. So that if they appear again one can look them up in the table instead of recomputing their solution. This approach is called **memoization**:

```
def FibonacciMemoize(n, table={0:1, 1:1}):
    print 'calling FibonacciMemoize(%i)' % (n)
    if table.has_key(n):
        print '    value is in table'
        return table[n]
    b=FibonacciMemoize(n-1,table)+FibonacciMemoize(n-2,table)
    table[n]=b
    print '    computing fibonacci(',n,')=', b, ' and storing in table'
    return b
```

## Huffman encoding

**Definition 54. Shannon-Fano encoding** (also known as **minimal prefix code**). In this encoding each character in a string is mapped into a sequence of bits so characters that appear with less frequency are encoded with in a longer sequence of bits while characters that appear with more frequency are encoded with a shorter sequence.

**Definition 55. Huffman Encoding.** The optimal choice for a minimal prefix code). The choice is built in the following way. We associate a tree to each character in the string to compress. Each tree is a trivial tree containing only one node: the root node. We then associate to the root node the frequency of the character representing the tree. We then extract from the list of trees the two trees with lowest frequency:  $t_1$  and  $t_2$ . We form a new tree  $t_3$ , we attach  $t_1$  and  $t_2$  to  $t_3$  and we associate a frequency to  $t_3$  equal to the sum of the frequencies of  $t_1$  and  $t_2$ . We repeat this operation until the list of trees containing only one tree. At this point we associate a sequence of bits to each node of the tree. Each bit corresponds to one level on the tree. The more frequent characters end up being closer to the root and are encoded with few bits, while rare characters are far from the root and encoded with more bits.

PKZIP, ARJ, ARC, JPEG, MPEG3 (mp3), MPEG4 and a other programs and compressed file formats all use the Huffman coding algorithm for compressing

strings. Note that Huffman is a compression algorithm with no-information-loss. In the JPEG and MPEG compression algorithms Huffman if combined with some form of cut of the Fourier specturm (for example MP3 is an audio compression format where frequencies below 2KHz are dumped and not compressed because they are not audible). Therefore the JPEG and MPEG formats are referred to as compression with information-loss.

### Longest common subsequence

Given two sequences of characters,  $S_1$  and  $S_2$ , this is the problem of determining the length of the longest common sub-sequence,  $LCS$ , that is a sub-sequence of both  $S_1$  and  $S_2$ .

Why might we want to solve the longest common subsequence problem? There are several motivating applications.

- **Molecular biology.** DNA sequences (genes) can be represented as sequences of four letters ACGT, corresponding to the four submolecules forming DNA. When biologists find a new sequences, they typically want to know what other sequences it is most similar to. One way of computing how similar two sequences are is to find the length of their longest common subsequence.
- **File comparison.** The Unix program `diff` is used to compare two different versions of the same file, to determine what changes have been made to the file. It works by finding a longest common subsequence of the lines of the two files and displays the set of lines that have changed. In this instance of the problem we should think of each line of a file as being a single complicated character.
- **Spelling Correction.** If a text contains a word,  $w$ , that is not in the dictionary, a ‘close’ word, i.e. one with a small edit distance to  $w$ , may be suggested as a correction. Transposition errors are common in written text. A transposition can be treated as a deletion plus an insertion, but a simple variation on the algorithm can treat a transposition as a single point mutation.
- **Speech Recognition.** Algorithms similar to the LCS are used in some speech recognition systems: find a close match between a new utterance

and one in a library of classified utterances.

Let's start with some simple observations about the LCS problem. If we have two strings, say "ATGGCACTACGAT" and "ATCGAGC", we can represent a subsequence as a way of writing the two so that certain letters line up:

```
ATGGCACTACGAT
|| | |
ATCG AG C
```

From this we can observe the following simple fact: if the two strings start with the same letter, it's always safe to choose that starting letter as the first character of the subsequence. This is because, if you have some other subsequence, represented as a collection of lines as drawn above, you can "push" the leftmost line to the start of the two strings, without causing any other crossings, and get a representation of an equally-long subsequence that does start this way.

On the other hand, suppose that, like the example above, the two first characters differ. Then it is not possible for both of them to be part of a common subsequence - one or the other (or maybe both) will have to be removed.

Finally, observe that once we've decided what to do with the first characters of the strings, the remaining subproblem is again a longest common subsequence problem, on two shorter strings. Therefore we can solve it recursively.

Rather than finding the subsequence itself, it turns out to be more efficient to find directly the length of the longest subsequence. Then in the case where the first characters differ, we can determine which subproblem gives the correct solution by solving both and taking the max of the resulting subsequence lengths. Once we turn this into a dynamic programming algorithm we get the following:

```
def LCS(X, Y)
    m=len(X)
    n=len(Y)
    c=[]
    for i in range(0,m):
        c.append([])
    for j in range(0,n):
        c[i].append(0)
    if X[i] == Y[j]:
        if i==0 or j==0:
            c[i][j]=1
        else:
            c[i][j]=c[i-1][j-1]+1
    else:
        c[i][j]=max(c[i-1][j], c[i][j-1])
```

```

        c[i][j]=1
    else:
        c[i][j] = c[i-1][j-1] + 1
    else:
        if i==0 or j==0:
            c[i][j]=0
        else:
            c[i][j] = max(c[i][j-1],c[i-1][j])
    return c[m-1][n-1]

```

The algorithms can be shown to be  $O(nm)$  (where  $m = \text{len}(X)$  and  $n = \text{len}(Y)$ ).

## Continuum Knapsack

The continuum Knapsack problem can be formulated as the problem of maximizing:

$$f(x) = a_0x_0 + a_1x_1 + \dots + a_nx_n$$

given the constraint

$$b_0x_0 + b_1x_1 + \dots + b_nx_n \leq c$$

where coefficients  $a_i$ ,  $b_i$  and  $c$  are provided and  $x_i \in [0, 1]$  are to be determined.

Using financial terms we can say that

- The set  $\{x_0, x_1, \dots, x_n\}$  forms a portfolio
- $b_i$  is the cost of investment  $i$
- $c$  is the total investment capital available
- $a_i$  is the expected return of investment for investment  $i$
- $f(x)$  is the expected value of our portfolio  $\{x_0, x_1, \dots, x_n\}$

Here is the solving algorithm:

```

def ContinuumKnapsack(a,b,c):
    n=len(a)
    table=[]
    f=0.0

```

```

for i in range(n):
    y=a[i]/b[i]
    table.append((y,i))
table.sort()
table.reverse()
for (y,i) in table:
    quantity=min(c/b[i],1)
    x.append((i,quantity))
    c=c-b[i]*quantity
    f=f+a[i]*quantity
return (f,x)

```

This algorithm is dominated by the sort therefore

$$T_{worst}(x) \in O(n \log n)$$

## Discrete Knapsack

The discrete Knapsack problem is very similar to the continuum knapsack problem but  $x_i \in \{0, 1\}$  (can only zero or one).

In this case a greedy approach does not apply and the problem is, in general NP complete. If we assume that  $c$  and  $b_i$  are all multiple of a finite factor  $\varepsilon$  then it is possible to solve the problem in  $O(c/\varepsilon)$ . Even when there is not a finite factor  $\varepsilon$ , we can always round  $c$  and  $b_i$  to some finite precision  $\varepsilon$  and we can conclude that, for any finite precision  $\varepsilon$ , we can solve the problem in linear time. The algorithm that solves this problem follows a dynamic programming approach.

We can re-formulate the problem in terms of simple capital budgeting problem. We have to invest \$5M. We assume  $\varepsilon = \$1M$ . We are in contact with 3 investment firms. Each of them offers a number of investment opportunities characterized by an investment cost  $c[i, j]$  and an expected return of investment  $r[i, j]$ . The index  $i$  labels the investment firm and the index  $j$  label the different investment opportunities offered by the firm. We have to build a portfolio that maximizes the return of investment. We cannot select more than one investment for each firm and we cannot select fractions of investments.

Without loss of generality we will assume that

$$c[i, j] \leq c[i, j + 1] \text{ and } r[i, j] \leq r[i, j + 1]$$

which means that investment opportunities for each firm are sorted according with their cost.

Let's consider the following explicit case:

	Firm	$i = 0$	Firm	$i = 1$	Firm	$i = 2$
proposal	$c[0, j]$	$r[0, j]$	$c[1, j]$	$r[1, j]$	$c[2, j]$	$r[2, j]$
$j = 0$	0	0	0	0	0	0
$j = 1$	1	5	2	8	1	4
$j = 2$	2	6	3	9	-	-
$j = 3$	-	-	4	12	-	-

(Table 1)

(table values are always multiple of  $\varepsilon = \$1M$ ).

Notice that we can label each possible portfolio by a triplet  $\{j_0, j_1, j_2\}$ .

A straightforward way to solve this is to try all possibilities and choose the best. In this case, there are only  $3 \times 4 \times 2 = 24$  possible portfolios. Many of these are infeasible (for instance, portfolio  $\{2, 3, 0\}$  costs \$6M and we cannot afford it). Other portfolios are feasible, but very poor (like portfolio  $\{0, 0, 1\}$  which is feasible but returns only \$4M)

Here are some disadvantages of total enumeration:

- For larger problems the enumeration of all possible solutions may not be computationally feasible.
- Infeasible combinations may not be detectable a priori, leading to inefficiency.
- Information about previously investigated combinations is not used to eliminate inferior, or infeasible, combinations (unless we use memoization but in this case the algorithm would not grow polynomially in memory space).

We can, instead, use a dynamic programming approach.

We break the problem into three stages and, at each stage we fill a table of optimal investments for each discrete amount on money. At each stage  $i$  we only consider investments from firm  $i$  and the table during the previous stage.

So stage 0 represents the money allocated to firm 0, stage 1 the money to firm 1, and stage 2 the money to firm 2.

STAGE ZERO: we maximize the return of investment considering only offers from firm 0. We fill a table  $f[0, k]$  with the maximum return of investment if we invest  $k$  million dollars on firm 0:

$$f[0, k] = \max_{j|c[0,j] < k} r[0, j] \quad (2.5)$$

$k$	$f[0, k]$
0	0
1	5
2*	6*
3	6
4	6
5	6

(2.6)

STAGE TWO: we maximize the retun of investment considering offers from firm 1 and the above table. We fill a table  $f[1, k]$  with the maximum return of investment if we invest  $k$  million dollars on firm 0 and firm 1:

$$f[1, k] = \max_{j|c[1,j] < k} r[1, j] + f[0, k - c[0, j]] \quad (2.7)$$

$k$	$c[2, j]$	$f[0, k - c[0, j]]$	$f[1, k]$
0	0	0	0
1	0	1	5
2	2	0	8
3	2	1	9
4	3	1	13
5*	4*	1*	18*

(2.8)

STAGE THREE: we maximize the retun of investment considering offers from firm 2 and the above table. We fill a table  $f[2, k]$  with the maximum return of investment if we invest  $k$  million dollars on firm 0, firm 1 and firm 2:

$$f[2, k] = \max_{j|c[2,j] < k} r[2, j] + f[1, k - c[1, j]] \quad (2.9)$$

$k$	$c[2, j]$	$f[1, k - c[1, j]]$	$f[2, k]$
0	0	0	0
1	0	1	5
2	2	0	8
3	2	1	9
4	1	3	13
5*	2*	3*	18*

(2.10)

The maximum return of investment with \$5M can is therefore \$18M. It can be achieved by investing \$2M on firm 2 and \$3M on firms 0 and 1. The optimal choice is marked with a star in each table. Note that in order to determine how much money have to be allocated in order to maximize the return of investment requires storing past tables in order to be able to lookup the solution to subproblems.

We can generalize eq.(2.7) and eq.(2.9) for any number of investment firms (decision stages):

$$f[i, k] = \max_{j | c[i, j] < k} r[i, j] + f[i - 1, k - c[i - 1, j]]$$

### 2.3. NP and NPC

**Definition 56.** We say a problem is in  $P$  if it can be solved in polynomial time:  $T_{worst} \in O(n^\alpha)$  for some  $\alpha$ .

**Definition 57.** We say a problem is in  $NP$  if an input string can be verified to be a solution in polynomial time:  $T_{worst} \in O(n^\alpha)$  for some  $\alpha$ .

**Definition 58.** We say a problem is in  $co\text{-}NP$  if an input string can be verified not to be a solution in polynomial time:  $T_{worst} \in O(n^\alpha)$  for some  $\alpha$ .

**Definition 59.** We say a problem is in  $NPH$  ( $NP$  Hard) if it is harder than any other problem in  $NP$ .

**Definition 60.** We say a problem is in  $NPC$  ( $NP$  Complete) if it is in  $NP$  and in  $NPH$ . Consequences:

$$\text{if } \exists x \mid x \in NPC \text{ and } x \in P \Rightarrow \forall y \in NP, y \in P$$

Open problems:

- $\text{co-NP} \stackrel{?}{=} \text{NP}$
- $\text{co-NP} \cap \text{NP} \stackrel{?}{=} \text{P}$
- $\text{NPC} \stackrel{?}{=} \text{NP}$

# **Part I**

## **Appendices**

## 2.4. Programs in Java

### Example: loop0

```
public static void loop0(int n) {  
    int i;  
    for(i=0; i<n; i++) {  
        System.out.println("i="+i);  
    }  
}
```

### Example: loop1

```
public static void loop1(int n) {  
    int i;  
    for(i=0; i<n*n; i++) {  
        System.out.println("i="+i);  
    }  
}
```

### Example: loop2

```
public static void loop2(int n) {  
    int i,j;  
    for(i=0; i<n; i++) {  
        System.out.println("i="+i);  
        for(j=0; j<n; j++) {  
            System.out.println("j="+j);  
        }  
    }  
}
```

### Example: loop3

```
public static void loop3(int n) {  
    int i,j;  
    for(i=0; i<n; i++) {  
        System.out.println("i="+i);  
    }
```

```

        for(j=0; j<i; j++) {
            System.out.println("j="+j);
        }
    }
}

```

**Example: loop4**

```

public static void loop4(int n) {
    int i,j;
    for(i=0; i<n; i++) {
        System.out.println("i="+i);
        for(j=0; j<i*i; j++) {
            System.out.println("j="+j);
        }
    }
}

```

**Example: factorial0**

```

public static int factorial0(int n) {
    int i, prod=1;
    for(i=1; i<=n; i++) prod=prod*n;
    return n;
}

```

**Example: concatenate0**

```

public static void concatenate0(int n) {
    for(int i=0; i<n*n; i++) System.out.println(""+i);
    for(int i=0; i<n*n*n; i++) System.out.println(""+i);
}

```

**Example: concatenate1**

```

public static void concatenate1(int n, int a) {
    if(a<0)

```

```
        for(int i=0; i<n*n; i++) System.out.println(""+i);
    else
        for(int i=0; i<n*n*n; i++) System.out.println(""+i);
}
```

#### **Example: factorial1**

```
public static int factorial1(int n) {
    if(n==0) return 1;
    else return n*factorial1(n-1);
}
```

#### **Example: recursive0**

```
public static int recursive0(int n) {
    if(n==0) return 1;
    else {
        loop3(n)
        return n*n*recursive0(n-1);
    }
}
```

#### **Example: recursive1**

```
public static int recursive1(int n) {
    if(n==0) return 1;
    else {
        loop3(n)
        return n*recursive1(n-1)*recursive1(n-1);
    }
}
```

#### **Example: recursive2**

```
public static int recursive2(int n) {
    if(n==0) return 1;
    else {
```

```
    int a;
    a=factorial0(n)
    return a*recursive2(n/2)*recursive2(n/2);
}
}
```

**Example: binarySearch**

```
public static int binarySearch(Comparable[] a, Comparable x) {
    int low=0, high=a.length-1;
    while(high>=low) {
        int mid=(high-low)/2;
        if(a[mid].compareTo(x)<0) low=mid+1;
        else if(a[mid].compareTo(x)>0) high=mid-1;
        else return mid;
    }
    return -1;
}
```

### 3. PRACTICE EXERCISES

#### 3.1. Limits

$$\begin{aligned}\lim_{n \rightarrow \infty} \frac{\sqrt{n^2 + 2}}{\sqrt{2n^2 - 1}} &= \frac{1}{\sqrt{2}} \\ \lim_{n \rightarrow \infty} \frac{2n^3 + n^5}{5n^7 + n^2} &= 0 \\ \lim_{n \rightarrow \infty} \frac{n^{1/3} + n^{5/9}}{2n^{1/2} - n^{1/10}} &= \infty \\ \lim_{n \rightarrow \infty} \frac{3n + \sqrt{n} \log n}{2n + \sqrt{n} e^{-n}} &= \frac{3}{2} \\ \lim_{n \rightarrow \infty} \frac{\sqrt{n+1} - 1}{\sqrt{n+2}} &= 1 \\ \lim_{n \rightarrow \infty} \frac{n}{3^n - 2^n} &= 0 \\ \lim_{n \rightarrow \infty} \frac{n^2}{n!} &= 0 \\ \lim_{n \rightarrow \infty} \frac{n^2 + 2^n + 1}{n^2 + 5^n + 2^n} &= 0 \\ \lim_{n \rightarrow \infty} \frac{n!}{n^{n/2}} &= \infty \\ \lim_{n \rightarrow \infty} \frac{\log(n^2 + 1) + \log(n^3 + 2)}{3 \log(n + 1) + \log(n)} &= \frac{5}{4} \\ \lim_{n \rightarrow \infty} \frac{2^n \sqrt{n!}}{n} &= 0 \\ \lim_{n \rightarrow \infty} \frac{2^{n+8}}{\log(n) + 2^n} &= 256\end{aligned}$$

For some of these limits you may need

$$n! = \sqrt{2\pi n} n^n e^{-n} (1 + O(1/n))$$

# CSC416. Class Notes (DRAFT)

Massimo Di Pierro

School of Computer Science, Telecommunications and Information Systems  
DePaul University, 243 S. Wabash Av, Chicago, IL 60604, USA

September 2, 2003

## Abstract

These notes provide a concise review to some of the material covered in the csc416 course. These notes are not intended as a substitution for the textbook. Attention: these notes are in a draft stage and may contain errors. Please report errors to mdipierro@cs.depaul.edu

## Contents

<b>1 Syllabus</b>	<b>4</b>
1.1 Approximate weekly schedule . . . . .	4
1.2 Grading Policy . . . . .	4
<b>2 Java review</b>	<b>5</b>
2.1 Primitive types . . . . .	5
2.2 Packages . . . . .	5
2.2.1 Required packages . . . . .	5
2.3 Input/Output . . . . .	5
2.3.1 To read from the console . . . . .	5
2.3.2 To write to the console . . . . .	5
2.3.3 To read from a file . . . . .	5
2.3.4 To write to a file . . . . .	6
2.3.5 How to break a string into tokens . . . . .	6
2.4 int, float, double vs Integer, Float, Double . . . . .	6
2.4.1 Store an int into an Integer . . . . .	6
2.4.2 Store an Integer into an int . . . . .	6
2.4.3 What is the difference then? . . . . .	6
2.4.4 Comparable interface . . . . .	7
2.5 More on Objects . . . . .	8
2.6 Exceptions . . . . .	8
2.6.1 try ... catch . . . . .	8
2.7 Conditionals . . . . .	9
2.7.1 if ... else . . . . .	9

2.7.2	<code>switch { case ... break ... otherwise }</code>	10
2.8	<code>Loops</code>	10
2.8.1	<code>for(...)</code>	10
2.8.2	<code>while(...)</code>	10
<b>3</b>	<b>Math review</b>	<b>11</b>
3.1	<code>Symbols</code>	11
3.2	<code>Set Theory</code>	11
3.2.1	<code>Important Sets</code>	11
3.2.2	<code>Set operations</code>	11
3.3	<code>Finite sums</code>	11
3.3.1	<code>Definition</code>	11
3.3.2	<code>Properties</code>	12
3.4	<code>Order of growth of functions</code>	13
3.4.1	<code>Definitions</code>	14
3.4.2	<code>Intuitive meaning</code>	14
3.4.3	<code>Practical rules</code>	14
3.4.4	<code>Example: loop0</code>	15
3.4.5	<code>Example: loop1</code>	15
3.4.6	<code>Example: loop2</code>	15
3.4.7	<code>Example: loop3</code>	15
3.4.8	<code>Example: loop4</code>	16
3.4.9	<code>Example: factorial0</code>	16
3.4.10	<code>Example: concatenate0</code>	16
3.4.11	<code>Example: concatenate1</code>	17
3.5	<code>Recursive alorithms and recurrence relations</code>	17
3.5.1	<code>Example: factorial1</code>	18
3.5.2	<code>Example: recursive0</code>	18
3.5.3	<code>Example: recursive1</code>	18
3.5.4	<code>Example: recursive2</code>	19
3.5.5	<code>Example: binarySearch</code>	19
3.5.6	<code>Example: Euclid Algorithm for the greatest common divisor</code>	19
<b>4</b>	<b>Algorithms Animator and Data Structures</b>	<b>20</b>
4.1	<code>Lists</code>	21
4.2	<code>Trees</code>	21
4.3	<code>Graphs</code>	23
<b>5</b>	<b>Algorithms</b>	<b>25</b>
5.1	<code>Defintions</code>	25
<b>6</b>	<b>Arrays and Lists</b>	<b>26</b>
6.1	<code>Arrays</code>	26
6.2	<code>Vectors/Lists using arrays</code>	26
6.3	<code>Lists</code>	28

<b>7 Book Programs</b>	<b>31</b>
7.1 AANode.java . . . . .	32
7.2 AATree.java . . . . .	33
7.3 AvlNode.java . . . . .	39
7.4 AvlTree.java . . . . .	40
7.5 BinaryHeap.java . . . . .	46
7.6 BinaryNode.java . . . . .	50
7.7 BinarySearchTree.java . . . . .	51
7.8 BinomialNode.java . . . . .	56
7.9 BinomialQueue.java . . . . .	57
7.10 Comparable.java . . . . .	62
7.11 CursorList.java . . . . .	63
7.12 CursorListItr.java . . . . .	67
7.13 CursorNode.java . . . . .	69
7.14 DisjSets.java . . . . .	70
7.15 DisjSetsFast.java . . . . .	72
7.16 DSL.java . . . . .	74
7.17 Hashable.java . . . . .	78
7.18 HashEntry.java . . . . .	79
7.19 LeftHeapNode.java . . . . .	80
7.20 LeftistHeap.java . . . . .	81
7.21 LinkedList.java . . . . .	84
7.22 LinkedListItr.java . . . . .	88
7.23 ListNode.java . . . . .	90
7.24 MyInteger.java . . . . .	91
7.25 Overflow.java . . . . .	93
7.26 PairHeap.java . . . . .	94
7.27 PairNode.java . . . . .	99
7.28 QuadraticProbingHashTable.java . . . . .	100
7.29 QueueAr.java . . . . .	105
7.30 Random.java . . . . .	108
7.31 RedBlackNode.java . . . . .	111
7.32 RedBlackTree.java . . . . .	112
7.33 Rotations.java . . . . .	117
7.34 SeparateChainingHashTable.java . . . . .	118
7.35 SkipNode.java . . . . .	122
7.36 Sort.java . . . . .	123
7.37 SplayTree.java . . . . .	130
7.38 StackAr.java . . . . .	136
7.39 StackLi.java . . . . .	139
7.40 Treap.java . . . . .	142
7.41 TreapNode.java . . . . .	147
7.42 Underflow.java . . . . .	148
<b>8 Finite Automata and Formal Grammars</b>	<b>149</b>

# 1 Syllabus

## 1.1 Approximate weekly schedule

1. Java and Math review. Algorithms Analysis. (book chapters 1 and 2)
2. Arrays, Lists, Stacks and Queues (book chapters 3)
3. Sorting (book chapter 7)
4. Hashing (book chapter 5)
5. Priority Queues and Trees (book chapter 6 and 4)
6. Trees, AVL Trees and Red-Black Trees (book chapter 4)
7. Graphs (book chapter 9)
8. Graph Algorithms (book chapters 8 and 9)
9. Finite Automata and Formal Grammars
10. Finite Automata and Formal Grammars

## 1.2 Grading Policy

$$FG = 0.50 \cdot HA + 0.25 \cdot ME + 0.25 \cdot FE$$

where  $FG$  is the final grade,  $HA$  is the average of the homework assignments excluding the lowest grade,  $ME$  is the midterm exam,  $FE$  is the final exam. The grade is converted to a letter according to the following table:

92–100	A
89–91	A–
86–88	B+
82–85	B
79–81	B–
76–78	C+
73–75	C
70–72	C–
67–69	D+
60–66	D
0–59	F

## 2 Java review

### 2.1 Primitive types

```
boolean  
char  
byte  
short  
int  
long  
float  
double
```

Note that `boolean` is either `true` or `false`, not equivalent to `int` as in C++

### 2.2 Packages

#### 2.2.1 Required packages

```
import java.io.*;      // for input/output  
import java.util.*;    // various stuff  
import java.awt.*;    // for graphics  
import java.applet.*; // for applets
```

### 2.3 Input/Output

#### 2.3.1 To read from the console

```
BufferedReader input=new BufferedReader(  
                    new InputStreamReader(System.in));  
String s=input.readLine(); // returns null if no input  
Integer s=Integer.valueOf(input.readLine());  
Double s=Double.valueOf(input.readLine());  
int s=Integer.valueOf(input.readLine()).intValue();  
double s=Double.valueOf(input.readLine()).doubleValue();
```

#### 2.3.2 To write to the console

```
System.out.print(""+whatever); // do not go to newline  
System.out.println(""+whatever); // do to newline
```

#### 2.3.3 To read from a file

```
BufferedReader input=new BufferedReader(  
                    new FileReader("filename"));  
String s=input.readLine(); // s==null at EOF
```

### 2.3.4 To write to a file

```
PrintWrtter output=new PrintWriter(
    new BufferedWriter(
        new FileWriter("filename")));
output.print(""+whatever);
output.println(""+whatever);
```

### 2.3.5 How to break a string into tokens

```
s="1.34, 2.45";
 StringTokenizer st=new StringTokenizer(s);
for(i=0; i<st.countTokens(); i++)
    x=Double.parseDouble(st.nextToken());
```

## 2.4 int, float, double vs Integer, Float, Double

`Integer` is an object wrapper of the primitive `int` type. `Float` is an object wrapper to the primitive `float` type. `Double` is an object wrapper to the primitive `Double` type.

By using objects instead of primitive types we can:

### 2.4.1 Store an int into an Integer

```
int i=5;
Integer j=new Integer(i);
```

### 2.4.2 Store an Integer into an int

```
Ingeter j=new integer(45);
int i=j.intValue();
```

### 2.4.3 What is the diffenerece then?

An `Integer` is a class therefore it has methods. For example:

```
String s=j.toString();
    (convert to String)
int i=j.intValue();
    (convert to int)
float a=j.floatValue();
    (convert to float)
double x=j.doubleValue();
    (convert to double)
```

```

j.parseInt("34");
        (read from a String)

i=j.compareTo(new Integer(45));
        (compare with another Integer)

int h=j.hashCode();
        (hash it!)

What is more important we can cast an Integer into an Object and vice versa:

Object obj=new Integer(45);
Integer j=(Integer) obj;

```

Since the same is true for any object we can, for example, build an array of `Object`(s) and store `Integer`(s), `Float`(s), `Double`(s) and `String`(s) into it:

```

Object array[4];
array[0]=new Integer(3);
array[1]=new Float(3.14);
array[2]=new Double(3.14);
array[3]=new String("Hello World");
System.out.println("array[0]="+ (Integer) array[0]);
System.out.println("array[1]="+ (Float) array[1]);
System.out.println("array[2]="+ (Double) array[2]);
System.out.println("array[3]="+ (String) array[3]);

```

We will use this trick to build generic containers.

#### 2.4.4 Comparable interface

Note that all numeric classes (`Integer`, `Float` and `Double`) and the `String` class implement the `Comparable` interface. This means they have a method (`compareTo`) that can be used to compare an object with another object. For example we can define our own integer class:

```

public class MyInteger implements Comparable {
    private int value;
    public MyInteger(int x) { value=x; }
    public int intValue() { return value; }
    public int compareTo(Objects other) {
        if(value<((MyInteger) other).value) return -1;
        else if(value>((MyInteger) other).value) return +1;
        else return 0;
    }
}

```

So that we can do the following (as with Integer):

```
MyInteger x=new MyInteger(3);
MyInteger y=new MyInteger(5);
boolean comp=x.compareTo(y);
```

(comp is -1 in this case because x<y).

We will often assume our objects implement comparable.

## 2.5 More on Objects

In Java any variable is either a primitive type or a reference type (a pointer to a memory location where an object is stored). Consider the following example:

```
Integer a=new Integer(3);
Integer b=new Integer(3);
if(a==b) System.out.println("yes");
else System.out.println("no");
```

This program prints "no" since, even if a stores 3 and b stores 3, a and b refer to different instances of Integer(3), meaning they refer to different copies of the object stored in different memory locations. The condition a==b compares the memory locations of the references, not the values stored in the objects.

The following programs would, instead, print "yes".

```
Integer a=new Integer(3);
Integer b=a;
if(a==b) System.out.println("yes");
else System.out.println("no");

int a=3;
int b=3;
if(a==b) System.out.println("yes");
else System.out.println("no");
```

## 2.6 Exceptions

### 2.6.1 try ... catch ...

Many built-in methods may throw exceptions in case of failure. We should enclose these methods in try ... catch ... . For example:

```
Integer i;
String s="Hello World";
try {
    i=Integer.parseInt(s);
} catch(Exception e) {
    System.out.println("s does not contain an integer");
}
```

Our methods should also throw exception when there is a possibility of failure of the method itself. We throw the exception to inform the caller of the method that the method is unable to complete its task.

Note the following program:

```
public class MyException extends Exception {};
public class Test01 {
    public static void f(String[] args) {
        try {
            if(args.length==0) throw new MyException();
            return 1;
        } catch(Excetion e) {
            System.out.println("error in f: "+e);
            return 0;
        }
    }
    public static main(String[] args) {
        try {
            int i=f(args);
        } catch(Excetion e) {
            System.out.println("error in main: "+e);
        }
    }
}
```

Note: exception is caught in `f`, not in `main`; `args.length` returns `int` and not `Integer`. It is possible to throw `Throwable` rather than `Exception` but the former cannot be caught and will cause the program to abort.

## 2.7 Conditionals

### 2.7.1 if ... else

```
boolean condition=true;
if(condition) {
    System.out.println("do this");
} else {
    System.out.println("do that");
}
```

Condition has to be boolean type, not int, Integer or else. Conditions can be combined using logical operators:

```
cond1 && cond2 (AND)
cond1 || cond2 (OR)
!cond1          (NOT)
```

### 2.7.2 switch { case ... break ... otherwise }

```
int i=2;
switch(i) {
    case 0: System.out.println("i=0"); break;
    case 1: System.out.println("i=1"); break;
    case 2: System.out.println("i=2"); break;
    default: System.out.println("i>2");
}
```

Note that switch works well with int and String but not with other types.

## 2.8 Loops

### 2.8.1 for(...)

```
int i;
for(i=0; i<100; i++) {
    System.out.println("i="+i);
}
```

In principle we could loop with Integer instead of int but it would be ugly:

```
Integer j;
for(j=new Integer(0);
    j.compareTo(new Integer(100))<0;
    j=new Integer(j.intValue()+1)) {
    System.out.println("i="+i);
}
```

In fact if j is Integer we cannot do j=j+1 or j++. This is why we do not do math with objects but only with primitive types. Java does not support operator overloading and this is one of its limitations.

### 2.8.2 while(...)

```
boolean cond=true;
while(cond) {
    System.out.println("cond="+cond);
    cond=false;
}
```

### 3 Math review

#### 3.1 Symbols

$\infty$	infinity
$\wedge$	and
$\vee$	or
$\cap$	intersection
$\cup$	union
$\in$	element or In
$\forall$	for each
$\exists$	exists
$\Rightarrow$	implies
:	such that
$\iff$	if and only if

(1)

#### 3.2 Set Theory

##### 3.2.1 Important Sets

$\emptyset$	empty set
$\mathbb{N}$	natural numbers $\{0,1,2,3,\dots\}$
$\mathbb{N}^+$	positive natural numbers $\{1,2,3,\dots\}$
$\mathbb{Z}$	all integers $\{\dots,-3,-2,-1,0,1,2,3,\dots\}$
$\mathbb{R}$	all real numbers
$\mathbb{R}^+$	positive real numbers (not including 0)
$\mathbb{R}^*$	positive numbers including 0

(2)

##### 3.2.2 Set operations

$\mathcal{A}$ ,  $\mathcal{B}$  and  $\mathcal{C}$  are some generic sets.

- Intersection

$$\mathcal{A} \cap \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ and } x \in \mathcal{B}\} \quad (3)$$

- Union

$$\mathcal{A} \cup \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ or } x \in \mathcal{B}\} \quad (4)$$

- Difference

$$\mathcal{A} - \mathcal{B} \stackrel{\text{def}}{=} \{x : x \in \mathcal{A} \text{ and } x \notin \mathcal{B}\} \quad (5)$$

### 3.3 Finite sums

#### 3.3.1 Definition

$$\sum_{i=0}^{i \leq n} f(i) \stackrel{\text{def}}{=} f(0) + f(1) + \dots + f(n-1) + f(n) \quad (6)$$

Corresponding Python functions:

```

public Interface Summable {
    public static double f(int i);
}
public class MyFunction implements Summable {
    public static double f(int i) {
        return i*i; // example
    }
}
public class Adder {
    public static double sum(Summable a, int imin, int imax) {
        int i;
        for(i=min; i<=imax; i++) {
            sum=sum+a.f(i);
        }
        return sum;
    }
    public static test() {
        double y=sum(new MyFunction(),0,10);
        System.out.println("sum="+y);
    }
}

```

### 3.3.2 Properties

- Linearity

$$\sum_{i=0}^{i \leq n} af(i) + bg(i) = a \left( \sum_{i=0}^{i \leq n} f(i) \right) + b \left( \sum_{i=0}^{i \leq n} g(i) \right) \quad (7)$$

Proof:

$$\begin{aligned}
\sum_{i=0}^{i \leq n} af(i) + bg(i) &= (af(0) + bg(0)) + (af(1) + bg(1)) + \dots + (af(n) + bg(n)) \\
&= af(0) + af(1) + \dots + af(n) + bg(0) + bg(1) + \dots + bg(n) \\
&= a(f(0) + f(1) + \dots + f(n)) + b(g(0) + g(1) + \dots + g(n)) \\
&= a \left( \sum_{i=0}^{i \leq n} f(i) \right) + b \left( \sum_{i=0}^{i \leq n} g(i) \right)
\end{aligned} \quad (8)$$

Examples:

$$\sum_{i=0}^{i \leq n} i = \frac{1}{2}n(n+1)$$

$$\begin{aligned}
\sum_{i=0}^{i \leq n} i^2 &= \frac{1}{6}n(n+1)(2n+1) \\
\sum_{i=0}^{i \leq n} i^3 &= \frac{1}{4}n^2(n+1)^2 \\
\sum_{i=0}^{i \leq n} x^i &= \frac{x^{n+1} - 1}{x - 1} \text{ (geometric sum)}
\end{aligned}$$

### 3.4 Order of growth of functions

Let  $f(x)$  be the running time of an algorithm P1 as function of the input size  $x$ . Let  $g(x)$  be the running time of another algorithm P2 as function of the input size  $x$ . The limit

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \quad (9)$$

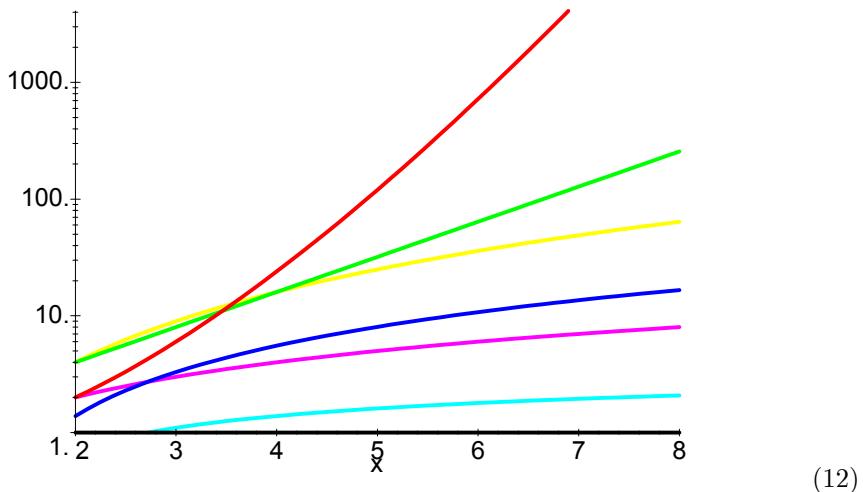
means that  $g(x)$  grows more than  $f(x)$  when  $x \rightarrow \infty$ . This induces a relation between  $f(x)$  and  $g(x)$ :

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0 \implies f(x) \text{ "grows less than" } g(x) \quad (10)$$

We can then sort functions according with their behavior at infinity for example:

$$1 \prec \log x \prec x \prec x \log x \prec x^2 \prec 2^x \prec x! \prec x^x \quad (11)$$

where the symbol  $\prec$  in this context reads "grow less than".



### 3.4.1 Definitions

$$\begin{aligned}
O(g(x)) &\stackrel{\text{def}}{=} \{f(x) : \exists x_0, c_0, \forall x > x_0, 0 \leq f(x) < c_0 g(x)\} \\
\Omega(g(x)) &\stackrel{\text{def}}{=} \{f(x) : \exists x_0, c_0, \forall x > x_0, 0 \leq c_0 g(x) < f(x)\} \\
\Theta(g(x)) &\stackrel{\text{def}}{=} O(g(x)) \cap \Omega(g(x)) \\
o(g(x)) &\stackrel{\text{def}}{=} O(g(x)) - \Omega(g(x)) \\
\omega(g(x)) &\stackrel{\text{def}}{=} \Omega(g(x)) - O(g(x))
\end{aligned}$$

### 3.4.2 Intuitive meaning

- $O(g(x))$  = Functions that grow no faster than  $g(x)$  when  $x \rightarrow \infty$
- $\Omega(g(x))$  = Functions that grow no slower than  $g(x)$  when  $x \rightarrow \infty$
- $\Theta(g(x))$  = Functions that grow at the same rate as  $g(x)$  when  $x \rightarrow \infty$
- $o(g(x))$  = Functions that grow slower than  $g(x)$  when  $x \rightarrow \infty$
- $\omega(g(x))$  = Functions that grow faster than  $g(x)$  when  $x \rightarrow \infty$

### 3.4.3 Practical rules

1. Compute the limit

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = a \quad (13)$$

2. Then look it up on the table

$$\begin{array}{lll}
a \text{ is positive or zero} & \implies f(x) \in O(g(x)) \Leftrightarrow f \preceq g \\
a \text{ is positive or infinity} & \implies f(x) \in \Omega(g(x)) \Leftrightarrow f \succeq g \\
a \text{ is positive} & \implies f(x) \in \Theta(g(x)) \Leftrightarrow f \sim g \\
a \text{ is zero} & \implies f(x) \in o(g(x)) \Leftrightarrow f \prec g \\
a \text{ is infinity} & \implies f(x) \in \omega(g(x)) \Leftrightarrow f \succ g
\end{array} \quad (14)$$

The rules assume the limits exist. The inverse is not true. For example:

$$f(x) \in \Theta(g(x)) \not\Rightarrow \lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} \text{ is positive} \quad (15)$$

**Theorem 1** Any polynomial  $P_m(n)$  of degree  $m$  has  $T(n) \in \Theta(n^m), \in O(n^m)$

**Theorem 2**  $T_1(n) \in O(f(n))$  and  $T_2 \in O(g(n)) \Rightarrow T_1(n)+T_2(n) \in O(\max(f(n), g(n)))$

**Theorem 3**  $T_1(n) \in O(f(n))$  and  $T_2 \in O(g(n)) \Rightarrow T_1(n)T_2(n) \in O(f(n)g(n))$

#### 3.4.4 Example: loop0

```
public static void loop0(int n) {  
    int i;  
    for(i=0; i<n; i++) {  
        System.out.println("i="+i);  
    }  
}
```

$$T(n) = \sum_{i=0}^{i \leq n} 1 = n \in \Theta(n) \Rightarrow \text{loop0} \in \Theta(n)$$

#### 3.4.5 Example: loop1

```
public static void loop1(int n) {  
    int i;  
    for(i=0; i<n*n; i++) {  
        System.out.println("i="+i);  
    }  
}
```

$$T(n) = \sum_{i=0}^{i \leq n^2} 1 = n^2 \in \Theta(n) \Rightarrow \text{loop1} \in \Theta(n^2)$$

#### 3.4.6 Example: loop2

```
public static void loop2(int n) {  
    int i,j;  
    for(i=0; i<n; i++) {  
        System.out.println("i="+i);  
        for(j=0; j<n; j++) {  
            System.out.println("j="+j);  
        }  
    }  
}
```

$$T(n) = \sum_{i=0}^{i \leq n} \sum_{j=0}^{j \leq n} 1 = \sum_{i=0}^{i \leq n} n = n^2 \in \Theta(n^2) \Rightarrow \text{loop2} \in \Theta(n^2)$$

#### 3.4.7 Example: loop3

```
public static void loop3(int n) {  
    int i,j;  
    for(i=0; i<n; i++) {
```

```

        System.out.println("i="+i);
        for(j=0; j<i; j++) {
            System.out.println("j="+j);
        }
    }
}

```

$$T(n) = \sum_{i=0}^{i \leq n} \sum_{j=0}^{j \leq i} 1 = \sum_{i=0}^{i \leq n} i = \frac{1}{2}n(n+1) \in \Theta(n^2) \Rightarrow \text{loop3} \in \Theta(n^2)$$

### 3.4.8 Example: loop4

```

public static void loop4(int n) {
    int i,j;
    for(i=0; i<n; i++) {
        System.out.println("i="+i);
        for(j=0; j<i*i; j++) {
            System.out.println("j="+j);
        }
    }
}

```

$$T(n) = \sum_{i=0}^{i \leq n} \sum_{j=0}^{j \leq i^2} 1 = \sum_{i=0}^{i \leq n} i^2 = \frac{1}{6}n(n+1)(2n+1) \in \Theta(n^3) \Rightarrow \text{loop4} \in \Theta(n^3)$$

### 3.4.9 Example: factorial0

```

public static int factorial0(int n) {
    int i, prod=1;
    for(i=1; i<=n; i++) prod=prod*n;
    return n;
}

```

$$T(n) = \sum_{i=0}^{i \leq n} 1 = n \in \Theta(n) \Rightarrow \text{factorial0} \in \Theta(n)$$

### 3.4.10 Example: concatenate0

```

public static void concatenate0(int n) {
    for(int i=0; i<n*n; i++) System.out.println(""+i);
    for(int i=0; i<n*n*n; i++) System.out.println(""+i);
}

```

$$T(n) = \Theta(\max(n^2, n^3)) \Rightarrow \text{concatenate0} \in \Theta(n^3)$$

### 3.4.11 Example: concatenate1

```
public static void concatenate1(int n, int a) {
    if(a<0)
        for(int i=0; i<n*n; i++) System.out.println(""+i);
    else
        for(int i=0; i<n*n*n; i++) System.out.println(""+i);
}
```

$$T(n) = \Theta(\max(n^2, n^3)) \Rightarrow \text{concatenate1} \in \Theta(n^3)$$

## 3.5 Recursive algorithms and recurrence relations

The running time  $T(n)$  of recursive algorithms is determined by:

1. Writing the recurrence relation in  $T(n)$
2. Solving the recurrence relation or, at least, put a bound on  $T(n)$ .

**In these notes we assume  $f(n)$  is a positive monotonic increasing function,  $a \geq 1$  and  $b \geq 2$ .**

For example:

```
public static int factorial1(int n) {
    if(n==0) return 1;
    else return n*factorial1(n-1);
}
```

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ T(n-1) + 1 & \text{if } n > 1 \end{cases} \Rightarrow T(n) = ?, T(n) \in ?$$

**Theorem 4** If  $P_m(n)$  is a polynomial of degree  $m$

$$T(n) = T(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(n^{m+1})$$

**Theorem 5** If  $P_m(n)$  is a polynomial of degree  $m$ ,  $a > 1$

$$T(n) = aT(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(a^n)$$

**Theorem 6** If  $P_m(n)$  is a polynomial of degree  $m$  and  $a < 1$

$$T(n) = aT(n-1) + P_m(n) \Rightarrow T(n) \in \Theta(n^m)$$

**Theorem 7** For  $m \geq 0$ ,  $p \geq 0$  and  $q > 1$ , from the Master Theorem (in its most general form) we conclude that:

$$T(n) = aT(n/b) + \Theta(n^m) \text{ and } a < b^m \Rightarrow T(n) \in \Theta(n^m)$$

$$T(n) = aT(n/b) + \Theta(n^m) \text{ and } a = b^m \Rightarrow T(n) \in \Theta(n^m \log n)$$

$$\begin{aligned}
T(n) &= aT(n/b) + \Theta(n^m) \text{ and } a > b^m \Rightarrow T(n) \in \Theta(n^{\log_m a}) \\
T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a < b^m \Rightarrow T(n) \in \Theta(n^m \log^p n) \\
T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a = b^m \Rightarrow T(n) \in \Theta(n^m \log^{p+1} n) \\
T(n) &= aT(n/b) + \Theta(n^m \log^p n) \text{ and } a > b^m \Rightarrow T(n) \in \Theta(n^{\log_b a}) \\
T(n) &= aT(n/b) + \Theta(q^n) \Rightarrow T(n) \in \Theta(q^n)
\end{aligned}$$

**Theorem 8** For any constant  $a > 0, b > 0$  and  $c > 0$

$$T(n) = T(a) + T(n - a - b) + c \Rightarrow T(n) \in \Theta(n)$$

### 3.5.1 Example: factorial1

```
public static int factorial1(int n) {
    if(n==0) return 1;
    else return n*factorial1(n-1);
}
```

$$T(n) = T(n - 1) + 1 \Rightarrow T(n) \in \Theta(n) \Rightarrow \text{factorial1} \in \Theta(n)$$

### 3.5.2 Example: recursive0

```
public static int recursive0(int n) {
    if(n==0) return 1;
    else {
        loop3(n)
        return n*n*recursive0(n-1);
    }
}
```

$$T(n) = T(n - 1) + P_3(n) \Rightarrow T(n) \in \Theta(n^4) \Rightarrow \text{recursive0} \in \Theta(n^4)$$

### 3.5.3 Example: recursive1

```
public static int recursive1(int n) {
    if(n==0) return 1;
    else {
        loop3(n)
        return n*recursive1(n-1)*recursive1(n-1);
    }
}
```

$$T(n) = 2T(n - 1) + P_3(n) \Rightarrow T(n) \in \Theta(2^n) \Rightarrow \text{recursive1} \in \Theta(2^n)$$

### 3.5.4 Example: recursive2

```
public static int recursive2(int n) {
    if(n==0) return 1;
    else {
        int a;
        a=factorial0(n)
        return a*recursive2(n/2)*recursive2(n/2);
    }
}
```

$$T(n) = 2T(n/2) + P_1(n) \Rightarrow T(n) \in \Theta(n \log n) \Rightarrow \text{recursive2} \in \Theta(n \log n)$$

### 3.5.5 Example: binarySearch

```
public static int binarySearch(Comparable[] a, Comparable x) {
    int low=0, high=a.length-1;
    while(high>=low) {
        int mid=(high-low)/2;
        if(a[mid].compareTo(x)<0) low=mid+1;
        else if(a[mid].compareTo(x)>0) high=mid-1;
        else return mid;
    }
    return -1;
}
```

$$T(n) = T(n/2) + 1 \Rightarrow \text{binarySearch} \in O(\log n)$$

### 3.5.6 Example: Euclid Algorithm for the greatest common divisor

```
public static long gcd(long m, long n) {
    while(n>0) {
        long tmp=m % n;
        m=n;
        n=tmp;
    }
    return m;
}
```

$$\text{gcd} \in O(\log n)$$

Note that this algorithm is the first known algorithm and was invented by Euclid in 400b.c. This algorithm plays a crucial role in the RSA encryption scheme used in public key cryptography (secure internet transactions).

# PSIM: Parallel Algorithms in Python

Massimo Di Pierro

School of Computer Science, Telecommunications and Informtaion Systems

DePaul University, 243 S. Wabash Av., Chicago, IL 60604, USA

February 1, 2005

## 1. Introduction

With the growing need for computing power, parallel architectures such as PC clusters are becoming more and more common. Many parallel algorithms are being developed to solve problems in various fields (from engineering to biology to finance) [1]. MPI [2] is becoming the most common message passing protocol for point-to-point parallel communication. It is an efficient and reliable protocol that is available for many programming languages (C/C++ in particular) and many different computer architectures.

Unfortunately the skills to develop parallel algorithms remain relegated to a small groups of experts. The problem is that teaching and learning parallel algorithms and parallel programming is difficult. It requires dedicated hardware (for example a PC cluster) as well as specialized software (a batch queue system, the MPI libraries, monitoring tools, etc.). Moreover, it requires extremely good programming skillss and detailed understanding of machine architecture.

In this notes we present PSIM (= Parallel SIMulator). This is a Python module that allows to develop and debug parallel algorithms using the Python language. The Python language [3] was invented by Guido van Rossum mainly as a teaching language and its syntax closely resemble typical pseudo-codes. Python is an interpreted language available for Windows, Unix/Linux and Mac systems. Python + PSIM allow to develop, implement and test parallel algorithms without any dedicated hardware. In this notes we briefly discuss the implementation of PSIM and its usage. We will also provide examples of parallel algorithms written using PSIM.

Note that PSIM is designed primarily as a teaching instrument. In fact Python, being an interpreted language, is not fast enough to justify its usage in real-life parallel applications. The Psim module itself does not implement any kind of speed optimization, does not rely on MPI and is written 100% in Python. Still, on a multiprocessor machine the different PSIM processes would be executed by different processors and may lead to a speed up in the execution.

## 2. Overview

`Psim` is a Python module. It consists of a single byte-code compiled file: `psim.pyc`, attached. In order to use `psim` one need to have the Python interpreter. The `psim` file has to be in the same directory as the parallel source file or in the Python `\libs` directory. Here is a typical parallel program that uses `psim`:

```
0: from psim import *
1: c=psim(2)
2: if c.process_id()==0: c.send(1,'Hello')
3: if c.process_id()==1: print c.receive(0)
```

Where:

- Line 0 imports the `psim` module
- Line 1 creates 2 parallel processes. The `psim` command is a constructor for a `psim` object (assigned to the variable `c`) that contains information associated to the created parallel processes. Each parallel process has a unique process id that is an integer number ranging from 0 to  $n - 1$  (where  $n$  is the total number of parallel processes).
- Line 2 checks if the present process has process id equal to 0. If true it sends the string 'Hello' to process number 1 (`process_id==1`).
- Line 3 checks if the present process has process id equal to 1. If true it receives data from process number 0 (`process_id==0`). The received data is returned.

This program prints 'Hello'.

Internally the `psim` constructor forks  $n$  copies of the present process (in the example  $n = 2$ ) so then whatever follows in the program is executed in parallel

according with the MIMD paradigm. Before returning, `psim` opens FIFO pipes between each couple of parallel processes. These pipes remain open as long as the `psim` object is referenced and they can be used to communicate.

When process  $i$  sends data,  $a$ , to process  $j$ , the data is serialized and written into the pipe that connects  $i$  with  $j$ . When process  $i$  receives data,  $a$ , from process  $j$ , the data is read from the pipe that connects  $i$  with  $j$ , de-serialized and returned. Every serializable Python object, including numbers, strings, tuples, lists and dictionaries, can be send/recieved. The serialization is implemented through the `pickle` module.

The `psim` module supports point-to-point communication (send/received) and some global communication functions (barrier, broadcast, global sum)

Other global communication functions can be implemented by the user using the following method `psim_obj.list`.

### 3. Syntax

- Create parallel processes:

```
psim_obj=psim(n [, fd])
```

This command forks the present process into  $n$  parallel processes and opens the communication channels (FIFO pipes). The second optional argument is a file descriptor,  $fd$ , open write-only. The file associated to the file descriptor is used as logfile by the `psim` communication functions to be used as a debugging tool. Returns a `psim` object containg information about the parallel processes.

- Get the number id of the present process

```
psim_obj.process_id()
```

Returns an integer type.

- Get the number of parallel processes

```
psim_obj.number_of_processes()
```

Returns an integer type.

- Asynchronous send:

```
psim_obj.send(i,data)
```

Sends data to process number  $i$ .  $data$  must be a serializable type. Returns `None`.

- Synchronous receive:

```
psim_obj.receive(i)
```

Sends data to process number  $i$ .  $data$  must be a serializable type. Returns the received data.

- Barrier\*:

```
psim_obj.barrier()
```

All parallel processes pause at barrier until all processes have reached the barrier. Returns `None`.

- Broadcast\*:

```
psim_obj.broadcast(i, data)
```

Process number  $i$  broadcast  $data$  to all the other processes. The received data is returned.

- Global sum\*:

```
psim_obj.sum(data)
```

All parallel processes sum their  $data$  in parallel. The sum is returned.

- Global list\*:

```
psim_obj.list(data)
```

All parallel processes construct a list of the  $data$  passed by each process. The list is broadcasted and returned by each processor. This method is used to implement global sum and some other global operations.

Note that `psim` only implements the asynchronous send and the synchronous receive. This is because `psim` is not thread based. We believe this is the most natural way to perform message passing and the easiest to teach. Moreover it does not create problems with buffer's de-allocation. Note also that the `psim` communication functions, differently from the corresponding MPI ones, do not support communication tags. This is because many message passing protocols, for example gm, do not support tags since they are inefficient. We believe it is

good programming practice not to use communication tags therefore we do not support them.

The \* accompanying some of the above methods indicates global communication functions. Global functions must be called by all processes at once because they involve all processes. For example if process  $i$  want to broadcast some *data* all processes must call `broadcast(i,data)` because all processes are involved in the communication of the data broadcasted by process  $i$ . This means that if the broadcast method is called in an if statement, the if condition must have the same logical value (true or false) for all processes. Some explicit examples are listed below.

## 4. Examples

In the above examples we assume the following required modules have been imported

```
from math    import *
from psim    import *
from sys     import *
from random  import *
from time   import *
```

### 4.1. Use of `psim_obj.send` and `psim_obj.receive` (ping-pong)

The following program creates two parallel processes. Process 0 sets a variable  $i$  to 0, prints  $i$  and sends it to process 1. It then receives a new value for  $i$  prints it and keeps looping. Process 1 receives the number  $i$  sent by process 0, increases it by one and sends it back. Each process loops  $m = 3$  times.

```
file=open('file.log', 'w')
c=psim(2,file)
i=0
m=3
print c.logfile
if c.process_id()==0:
    while i<m:
        print i
        c.send(1,i)      # send
```

```

    i=c.receive(1)  # receive
else:
    while i<m:
        i=c.receive(0)  # receive
        i=i+1
        c.send(0,i)      # send

```

The program produces the following output:

012

The program produces a log file (file.log) like the following

```

START: creating 2 parallel processes
START: done.
process 1: receive(0) starting...
process 1: receive(0) received 0
process 1: receive(0) done.
process 1: send(0,1) starting...
process 1: send(0,1) success.
process 1: receive(0) starting...
process 1: receive(0) received 1
process 1: receive(0) done.
process 1: send(0,2) starting...
process 1: send(0,2) success.
process 1: receive(0) starting...
process 1: receive(0) received 2
process 1: receive(0) done.
process 1: send(0,3) starting...
process 1: send(0,3) success.
START: creating 2 parallel processes
START: done.
process 0: send(1,0) starting...
process 0: send(1,0) success.
process 0: receive(1) starting...
process 0: receive(1) received 1
process 0: receive(1) done.
process 0: send(1,1) starting...
process 0: send(1,1) success.

```

```

process 0: receive(1) starting...
process 0: receive(1) received 2
process 0: receive(1) done.
process 0: send(1,2) starting...
process 0: send(1,2) success.
process 0: receive(1) starting...
process 0: receive(1) received 3
process 0: receive(1) done.

```

Note that output is buffered separately by the different parallel processes and flushed in an unpredictable order.

#### **4.2. Use of *psim\_obj.broadcast***

In the following program process 0 takes a string as input, *a*, and broadcasts it to *n* parallel processes. Each process prints its own process id and the string. Finally all processes, except process 0, exit, while process 0 returns.

```

def PsimTestBroadcast(n):
    c=psim(n)
    a=0
    i=c.process_id()
    if i==0:
        a=input('Type a number:')
    a=c.broadcast(0,a)
    print i, a
    if i!=0:
        exit(0)

```

The argument of the function is the number of desired processes. It can be tested with the following code:

```
PsimTestBroadcast(4)
```

#### **4.3. Use of *psim\_obj.list***

In the following program each process, among *n* parallel processes, creates a variable *j* and add it to a common list. The list is retrurned.

```

def PsimTestList(n):
    c=psim(n)
    i=c.process_id()
    j=i*i
    a=c.list(j)
    if i!=0:
        exit(0)
    return a

```

The argument of the function is the number of desired processes. It can be tested with the following code:

```
print PsimTestList(4)
```

The program outputs [0,1,4,9]. The `list` method can be used to implement other global communication functions.

#### 4.4. Use of `psim_obj.sum`

The following program sums the elements of an input list A in parallel (without using global communication functions):

```

def PsimSum1(A,n):
    c=psim(n)
    i=c.process_id()
    step=len(A)/n
    j,k=i*step,(i+1)*step
    sum=0
    for x in A[j:k]:
        sum=sum+x
    if i!=0:
        c.send(0,sum)          # send
        exit(0)                  # terminate process
    for i in range(1,n):
        sum=sum+c.receive(i) # receive
    return sum

```

It can be tested with the following code

```
print PsimSum1([3,6,4,7,8,9], 2)
```

Each parallel process  $i$  sums the sublist  $A[j:k]$  and sends the partial sum to process 0. Process 0 receives the partial sums and adds them all. The algorithm assumes that  $\text{len}(A)$  is divisible by  $n$ , the number of processes. The case  $\text{len}(A)$  not divisible by  $n$  is left as an exercise to the reader.

The same algorithm can be implemented using the global communication method `psim_obj.sum` as follows:

```
def PsimSum2(A,n):
    c=psim(n)
    i=c.process_id()
    step=len(A)/n
    j,k=i*step,(i+1)*step
    sum=0
    for x in A[j:k]:
        sum=sum+x
    sum=c.sum(sum) # global sum
    if i!=0:
        exit(0)      # terminate process
    return sum
```

#### 4.5. Parallel search

The following is a parallel program that searches if a value  $x$  is in a list  $A$ . The search function runs in parallel over  $n$  processes

```
def PsimSearch(x,A,n=1):
    c=psim(n)
    i=c.process_id()
    p=int(i*len(A)/n)
    q=int((i+1)*len(A)/n)
    check=0
    for k in range(p,q):
        if A[k]==x:
            check=1
    check=c.sum(check)
    if c.process_id()!=0:
        exit(0)
    return check>0
```

It can be tested with the following code

```
x=4
A=[5,6,7,8,9,3,4,6,7]
n=3
print PsimSearch(x, A, n)
```

#### 4.6. Parallel MergeSort

The following is a parallel implementation of the non-recursive mergesort. It sorts the elements of the input list  $A$ . The program assumes  $\text{len}(A)$  to be an integer multiple of  $n$ , the number of processes.

```
def Merge(A,p,q,r):
    B=[]
    i=p
    j=q
    while 1:
        if A[i]<=A[j]:
            B.append(A[i])
            i=i+1
        else:
            B.append(A[j])
            j=j+1
        if i==q:
            while j<r:
                B.append(A[j])
                j=j+1
            break
        if j==r:
            while i<q:
                B.append(A[i])
                i=i+1
            break
    A[p:r]=B

def MergeSortAux(A,j,k,blocksize):
    while blocksize<k:
```

```

        for p in range(j, k, 2*blocksize):
            q=p+blocksize
            r=min(q+blocksize, k)
            if r>q:
                Merge(A,p,q,r)
            blocksize=2*blocksize

def PsimMergeSort(A,n=1):
    listsize=len(A)
    step=int(listsize/n)
    if float(listsize/n)!=step:
        raise Exception
    c=psim(n)
    i=c.process_id()
    j,k=i*step,(i+1)*step
    MergeSortAux(A,j,k,1)
    if i>0:
        c.send(0,A[j:k])           # send
        exit(0)                   # terminate process
    for i in range(1,n):
        j,k=i*step,(i+1)*step
        A[j:k]=c.receive(i)      # receive
    MergeSortAux(A,0,len(A),step)
    return A

```

It can be tested with the following code

```

A=[7,8,4,5,6,2,9,0]
n=4
print PsimMergeSort(A,n)

```

Each process *i* sorts the sublist  $A[j:k]$  using the ordinary non-recursive `MergeSortAux`. When this operation is completed, in the line

```
c.send(0,A[j:k])
```

process number *i* sends the sorted sublist  $A[j:k]$  to process 0 and terminates. Process 0 receives all sublists in place, replaces the unsorted elements, then calls `MergeSortAux` to merge the sorted sublists.

## 5. Advanced issues

Our specific implementation of `psim` allows for individual processes in a parallel application to iteratively parallelize themselves. For example consider the following program:

```
def TestPsim2(a,b):
    c0=psim(a)
    if c0.process_id()<a/2:
        c1=psim(b)
        print '(',c0.process_id(),',',c1.process_id(),') '
    else:
        print '(',c0.process_id(),') '
```

It can be tested with the following code

```
TestPsim2(4,3)
```

The first call to `psim` forks into four processes 0,1,2 and 3. Then the first two of these processes call `psim` a second time and fork into three processes each. This program produces the following output:

```
(0,0) (0,1) (0,2) (1,0) (1,1) (1,2) (2) (3)
```

## 6. Conclusions

We believe `psim`, in combination with the Python language, can be a useful tool for teaching and testing parallel algorithms. It can be useful to instructors as well as developers. In fact developers will have the ability to perform rapid development of parallel algorithms in Python without need for expensive parallel architectures and without getting lost in implementation details that are typical of compiled languages such as C/C++. Once the algorithm is developed and tested it is an easy programming exercise to implement it in C/C++ with MPI.

`Psim` can be downloaded from: [www.phoenixcollective.org/mdp/psim.py](http://www.phoenixcollective.org/mdp/psim.py)

## References

- [1] Ian Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley Pub Co. (1995) ISBN: 0201575949

- [2] Peter Pacheco, *Parallel Programming With MPI*, Morgan Kaufmann. (1997)  
ISBN: 1558603395
- [3] David M. Beazley and Guido Van Rossum, *Python Essential Reference*, New  
Riders Publishing. (2001) ISBN: 0735710910

**Topic:**

# **ECT582**

**Secure  
Electronic  
Commerce**



## **Tentative Program Overview**

- Week 1 - Introduction, Examples and Definitions**
- Week 2 - Types of Attack and Threat Risk Modeling**
- Week 3 - Authentication, Authorization and Session Management**
- Week 4 - Data Validation, Injections and Cross Site Scripting**
- Week 5 - Handling e-Commerce Payments and Logging**
- Week 6 - Web Services Security**
- Week 7 - Encryption Algorithms and Hashing**
- Week 8 - Asymmetric Encryption and SSL**
- Week 9 - PKI, Digital Certificates, CRL**
- Week 10 - Securing Apache, mod\_ssl**

## Textbook and other resources

OWAP 2.0

A Guide to Building Secure Web Applications and Web Services

[www.owasp.org](http://www.owasp.org)

Stallings, William

*Network Security Essentials - Applications and Standards, 2nd ed.*

Upper Saddle River, New Jersey: Prentice Hall.

## Introduction



## Introduction / Definitions

**Vulnerability:** weakness in the security of a system

**Threat:** a set of circumstances that can cause harm

**Attack:** an intentional exploitation of a vulnerability

**Interception:** unauthorized party has control of an asset (information or resource)

**Interruption:** an asset (information or resource) becomes lost or unavailable

**Modification:** an asset (information or resource) is changed

**Fabrication:** new fake information or resources are created and made available

**Confidentiality + Integrity + Availability = Security**

## Introduction / Vulnerabilities (web)

### **Vulnerabilities:**

- **Interception**

- information is stolen from the source
- information is stolen during transmission

- **Interruption**

- web server down or slow
- network connection down or slow

- **Modification**

- data on server is compromised
- data is compromised during transmission

- **Fabrication**

- fake data is inserted in the server (by un-authorized party)
- fake data is inserted during transmission

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

Which users is authorized to access a certain resource?

Which operations is user XYZ permitted to do with resource ABC?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What information should be considered public?

What should be accessed only by authorized users?

What should be considered private of each user?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What are the steps required to authorize a certain operation?

Should the administrators have access to everything?

How do we prevent it?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

Can a software bug cause expose confidential data?

Can a software bug cause unauthorized access?

How do we prevent it?

## Why Security?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What if an authorized user of the system commits a mistake?

What if an impatient user submits twice the same form? Can this result in data corruption? How do we prevent it?

## Introduction / Defense Systems

### Methods of Defense:

**Prevent it** by blocking attacks, closing vulnerabilities  
(security settings, good coding, antivirus, firewall)

ECT582

**Deter it** by making the attack hard and not worthwhile  
(strong encryption, firewall configuration, security policies)

**Deflect it** by making another target more attractive  
(network configuration, decoy programs)

**Detect it** by monitoring your system/network  
(use port scans, monitor network traffic and check logs)

**Recover** from its effect  
(make regular backups and check backups and logs)

## Laws, Regulations and Standards

	USA	AU	EU	
National Legislation	Sarbanes-Oxley (SOX)	TPA	Data Protection	Legal Legal
Industry Regulations	VISA PCI UCCC	VISA PCI UCCC	VISA PCI	
National Standards	NIST Sp800* ISO 17799	ISO 17799	ISO 17799	Security Reviewer
Industry Standards	OWASP, ITIL, COBIT			Business
Organization	Information Security Policy			CSO/CIO

## Sarbanes-Oxley

Sarbanes-Oxley Act is the single most important piece of legislation affecting corporate governance, financial disclosure and the practice of public accounting since the US securities laws of the early 1930s.

### Section 404: Management Assessment Of Internal Controls.

Requires each annual report of an issuer to contain an "internal control report", which shall:

- (1) state the responsibility of management for establishing and maintaining an adequate internal control structure and procedures for financial reporting; and
- (2) contain an assessment, as of the end of the issuer's fiscal year, of the effectiveness of the internal control structure and procedures of the issuer for financial reporting.

[http://www.aicpa.org/info/sarbanes\\_oxley\\_summary.htm](http://www.aicpa.org/info/sarbanes_oxley_summary.htm)

## VISA PCI/CISP

**PCI=Payment Card Industry**

**CISP=Cardholder Information Security Program**

**Mandatory since June 2001**

- CISP compliance is required for all merchants and service providers that store, process, or transmit Visa cardholder data.
- The program [applies to all payment channels, including retail](#).
- To achieve CISP compliance merchants and service providers must implement PCI
- PCI/CISP is a result of a collaboration between Visa and MasterCard and is
- Other card companies operating in the U.S. have also endorsed the PCI Data Security Standard within their respective programs.
- If a member, merchant or service provider does not comply with the security requirements or fails to rectify a security issue, Visa may fine the responsible member and/or impose restrictions on the merchant or its agent.
- Members receive protection from fines for merchants or service providers that have been compromised but found to be CISP-compliant at the time of the security breach.

## VISA PCI/CISP

### **PCI Guidelines:**

1. Install and maintain a firewall configuration to protect data
2. Do not use vendor-supplied defaults for system passwords and other security parameters
3. Protect stored data
4. Encrypt transmission of cardholder data and sensitive information across public networks
5. Use and regularly update anti-virus software
6. Develop and maintain secure systems and applications
7. Restrict access to data by business need-to-know
8. Assign a unique ID to each person with computer access
9. Restrict physical access to cardholder data
10. Track and monitor all access to network resources and cardholder data
11. Regularly test security systems and processes
12. Maintain a policy that addresses information security

## ISO 17799 – The International Security Standard

First Published in 2002, revised in 2005

A comprehensive set of controls comprising best practices in Information Security

- Security Policy
- System Access Control
- Computer & Operations Management
- System Development and Maintenance
- Physical and Environmental Security
- Compliance
- Personnel Security
- Security Organization
- Asset Classification and Control
- Business Continuity Management (BCM)

<http://www.iso.org/iso/en/commcentre/pressreleases/2005/Ref963.html>

## ITIL

Originally developed by the British government,

ITIL ®, is a series of documents that are used to aid the implementation of a framework for IT Service Management (ITSM).

**12 main ITIL processes:** Availability Management; Capacity Management; Configuration Management; Change Management; Financial Management; **Incident Management; IT Service Continuity Management;** Problem Management; Release Management; **Security Management;** Service Desk; Service Level Management.

This is not free, they charge for the documentation!

<http://www.itil-toolkit.com/>

## COBIT

### Now version 4.0

COBIT is an IT governance framework and supporting toolset that allows managers to bridge the gap between control requirements, technical issues and business risks.

4 domains:

- Planning and Organization
- Acquisition and Implementation
- Delivery and Support
- Monitoring

COBIT is typically used as a SOX control framework  
Some COBIT objectives can be achieved via OWASP controls

## OWASP

### Guide “Black hat 2.0” published in 2005

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software.

It is an open source project that produces free, open-source documentation, tools, and standards.

The OWASP community also organizes conferences, local chapters, articles, papers, and message forums.

The OWASP Foundation is a not-for-profit charitable organization that ensures the ongoing availability and support for the work.

Participation in OWASP is free and open to all.

<http://www.owasp.org/index.jsp>

## General Secure (good) Practices

- 1) Adopt a versioning system (for example CVS)
- 2) Adopt a coding methodology (design / test / document)
- 3) Adopt a coding standard
- 4) Classify the assets (data) to be protected
- 5) Minimize Attack Surface Area
- 6) Keep it simple
- 7) Enforce authentication of all users
- 8) Perform separation of duties
- 9) Apply “Principle of Least Privilege”
- 10) Apply “Principle of Defense in Depth”
- 11) Secure the default behavior
- 12) Make sure you fail securely
- 13) Assume external systems are insecure
- 14) Do not use Security through Obscurity

## General Secure (good) Practices

### 1) Adopt a versioning system (for example CVS)

A versioning system is an application that records changes you make to your code.

- The code is “checked out” from a repository
- The code is modified by the programmers
- The new code is “checked in” together with a description of the changes

The versioning system stores all past changes with relative dates and descriptions. It also notifies the users about potential conflicts due to forks.

The simplest versioning system is RCS. RCS has two commands

```
co [filename]  
ci [filename]
```

CVS is an extension of RCS. It can check in/out entire folders.

## General Secure (good) Practices

### 2) Adopt a coding methodology (design / test / document)

Examples:

- *Waterfall*
- *Incremental*
- *Spiral*
- *Agile*
- *Extreme-Programming*

Suggestions:

- divide your application in “modules” where each module comprises a minimal but complete set of functionalities.
- Write the specifications of the module before coding each module then,
- have at least two people code each module.
- Have somebody else test each module.
- Integrate the modules
- For web application it is important to be able to modify specs dynamically

## General Secure (good) Practices

### 3) Adopt a coding standard

- Describe the directory structure (what is where)
- Document each function / minimize in-line comments
- Code a test function/method for each module/class
- Adopt a naming convention for classes/methods/functions/objects/variables
- Adopt a convention for how functions/methods should report errors
- Adopt a convention for how errors should be reported to the user
- Adopt a convention for the default behaviour of your system
- ...
- Isolate authentication code from rest of code
- Minimize use of cookies
- Use session variables
- Validate all user input
- Escape all user input that is displayed in a web page
- ...

## General Secure (good) Practices

### 4) Classify the assets (data) to be protected

- Make a list of your database tables and relative fields
- For each field, for each table, document who has access to that file

Example:

Table	Users	
	Field Name	only user can read it
	Field Username	only authentication function can read it
	Field Password	only user can write it, only authentication func can read
	Field Balance	only user can read it, sell func can read/write

Table	Items	
	Field Name	everybody can read
	Field InStock	everybody can read, sell func can read/write
	Field Cost	everybody can read

## General Secure (good) Practices

### 5) Minimize Attack Surface Area

- Assume each exposed functionality is a potential security risk  
(for example SQL injections, XSS, etc.)
- Limit access to functionalities as much as compatible with requirements
- Make sure all user input is validated

## **General Secure (good) Practices**

### **6) Keep it simple**

- Keep both the code design and the user interface as simple as possible
- It may be convenient to draw a graph where each vertex is a possible web page and arrows represents links (including forms) connecting one page to another.
- Each page can have multiple arrows in and multiple arrows out but should not have multi arrows in and multiple arrows out (bad design).
- Make sure each page validates its input and checks for the referrer page.
- The code each page separately by calling common functions/methods.

## **General Secure (good) Practices**

### **7) Enforce authentication of all users**

- Any web application should have a public side (which exposes text but no functionalities and/or forms) and restricted side that requires a one-time authentication (login/password/token/etc)
- Authentication must be done via an encrypted channel (ssl)
- Authentication information should not be sent back to the user (with web or email)
- If public access is required force a free registration and send a one-time password to the registrant by email or SMS. Then record the email address.
- If anonymous access is required store the remote ip address
- Log all login attempts (success and failure)

## **General Secure (good) Practices**

### **8) Perform separation of duties**

- Administrators should not be users of the application (this does not mean that the same person cannot have two logins, one as user and one as administrator).
- Administrators should not be able to buy good or communicate on behalf of other users.
- For more complex web application where users can have various roles (buyers/sellers/etc.) create different users types to prevent the same user to assume conflicting roles (for example order an item and certify that a payment has been received).

## **General Secure (good) Practices**

### **9) Apply “Principle of Least Privilege”**

- Design your application around a hierarchy of privileges. The user with more privileges can do more than the user with less privileges.
- Assign each user the least privilege required to access the functionality/data he is supposed to access. This prevents the user from accidentally/maliciously accessing functionalities/data he/she is not supposed to access.
- Often a single number is sufficient to identify the privilege level of a user and the least required privilege level for each asset. For each user the number is stored in the TABLE USERS together with the username and hashed password.
- For more sophisticated systems (example: a web site to access classified info from the department of defense), a classification systems may be required in which privileges are typically identified by a vector of numbers.

## **General Secure (good) Practices**

### **10) Apply “Principle of Defense in Depth”**

- Even if a single control is better, more controls are better.

Example, to buy an item...

- The user must be logged in.
- The user must have logged in no longer than 15mins before
- Check that remote IP address is the same used at login
- Check the referrer page is valid
- If item is very expensive, ask the user to login again and verify purchase by phone or email

## **General Secure (good) Practices**

### **11) Secure the default behavior**

- Force use of SSL for all sensitive information, this should not be an “optional” feature.
- Force users to adopt complex passwords (long, mixed cases with numbers)
- Force users to change passwords
- Disable users than don’t change passwords when requested and/or don’t use the system for long time
- Validate all fields
- Escape all output
- Log all important successful and unsuccessful transactions.

## General Secure (good) Practices

### 12) Make sure you fail securely

Avery operation can fail in two ways:

- *Invalid use action (invalid field for example)*
  - *Error in code*
- 
- Notify the user of any invalid action (invalid field) and revert to a state equal to the state before the action was initiated
  - In case of any error in code, log the error, email the administrators, perhaps release a ticket to the user, logout the user. Make sure all code is in a **try... catch...** in order to achieve this.

## General Secure (good) Practices

### 13) Assume external systems are insecure

- If your application talks to a database or to another application don't assume they are secure.

Examples:

- Your application passes a user's field to a second application that queries an SQL database; don't assume the second application will check the field for SQL injections. Do the check before calling the second application.
- Your application receives data from a database filled by a second application and displays the data. Don't assume the data in a format valid for display in text/html, always escape data.

## **General Secure (good) Practices**

### **14) Do not use Security through Obscurity**

Don't assume that, if your code has a vulnerability, if you don't tell anybody, nobody will find out. Please accidentally or maliciously do find out.

Consider the case of vulnerabilities in Microsoft Explorer and/or IIS. New vulnerabilities are found weekly even if they are closed source projects.

Consider the use of well-established open source systems. Even if it is easier to find vulnerabilities there, they are usually fixed in a short time.

## **Topic:**

# **ECT582**

**Secure  
Electronic  
Commerce**



## Tentative Program Overview

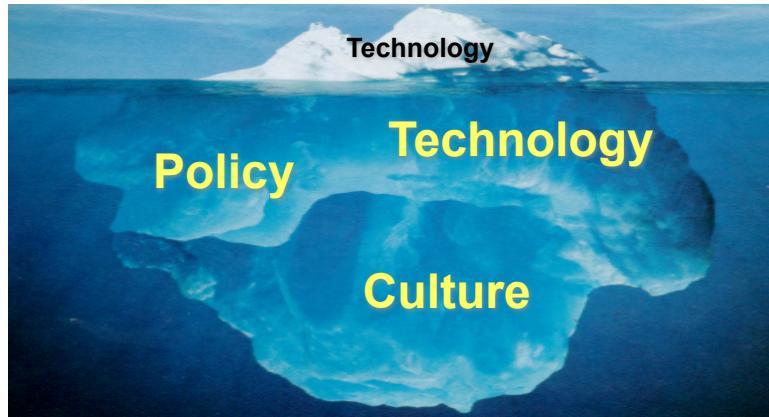
- Week 1 - Introduction, Examples and Definitions**
- Week 2 - Types of Attack and Threat Risk Modeling**
- Week 3 - Authentication, Authorization and Session Management**
- Week 4 - Data Validation, Injections and Cross Site Scripting**
- Week 5 - Handling e-Commerce Payments and Logging**
- Week 6 - Web Services Security**
- Week 7 - Encryption Algorithms and Hashing**
- Week 8 - Asymmetric Encryption and SSL**
- Week 9 - PKI, Digital Certificates, CRL**
- Week 10 - Securing Apache, mod\_ssl**

## Textbook and other resources

**OWAP 2.0**  
**A Guide to Building Secure Web Applications and Web Services**  
[www.owasp.org](http://www.owasp.org)

**Stallings, William**  
***Network Security Essentials - Applications and Standards, 2nd ed.***  
Upper Saddle River, New Jersey: Prentice Hall.

## Introduction



## Introduction / Definitions

**Vulnerability:** weakness in the security of a system

**Threat:** a set of circumstances that can cause harm

**Attack:** an intentional exploitation of a vulnerability

**Interception:** unauthorized party has control of an asset (information or resource)

**Interruption:** an asset (information or resource) becomes lost or unavailable

**Modification:** an asset (information or resource) is changed

**Fabrication:** new fake information or resources are created and made available

$$\text{Confidentiality} + \text{Integrity} + \text{Availability} = \text{Security}$$

## Introduction / Vulnerabilities (web)

### Vulnerabilities:

- **Interception**

- information is stolen from the source
- information is stolen during transmission

- **Interruption**

- web server down or slow
- network connection down or slow

- **Modification**

- data on server is compromised
- data is compromised during transmission

- **Fabrication**

- fake data is inserted in the server (by un-authorized party)
- fake data is inserted during transmission

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

Which users is authorized to access a certain resource?

Which operations is user XYZ permitted to do with resource ABC?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What information should be considered public?

What should be accessed only by authorized users?

What should be considered private of each user?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What are the steps required to authorize a certain operation?

Should the administrators have access to everything?

How do we prevent it?

## Why Security ?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

Can a software bug cause expose confidential data?

Can a software bug cause unauthorized access?

How do we prevent it?

## Why Security?

Protecting Information and Resources

Ensuring Privacy

Facilitating Workflow

Software bugs

Compensating for human error or neglect

What if an authorized user of the system commits a mistake?

What if an impatient user submits twice the same form? Can this result in data corruption? How do we prevent it?

## Introduction / Defense Systems

### Methods of Defense:

**Prevent it** by blocking attacks, closing vulnerabilities  
(security settings, good coding, antivirus, firewall)

ECT582

**Deter it** by making the attack hard and not worthwhile  
(strong encryption, firewall configuration, security policies)

**Deflect it** by making another target more attractive  
(network configuration, decoy programs)

**Detect it** by monitoring your system/network  
(use port scans. monitor network traffic and check logs)

**Recover** from its effect  
(make regular backups and check backups and logs)

## Laws, Regulations and Standards

	USA	AU	EU	
National Legislation	Sarbanes-Oxley (SOX)	TPA	Data Protection	Legal
Industry Regulations	VISA PCI UCCC	VISA PCI UCCC	VISA PCI	Legal
National Standards	NIST Sp800* ISO 17799	ISO 17799	ISO 17799	Security Reviewer
Industry Standards	OWASP, ITIL, COBIT			Business
Organization	Information Security Policy			CSO/CIO

## Sarbanes-Oxley

**Sarbanes-Oxley Act is the single most important piece of legislation affecting corporate governance, financial disclosure and the practice of public accounting since the US securities laws of the early 1930s.**

### **Section 404: Management Assessment Of Internal Controls.**

Requires each annual report of an issuer to contain an "internal control report", which shall:

- (1) state the responsibility of management for establishing and maintaining an adequate internal control structure and procedures for financial reporting; and
- (2) contain an assessment, as of the end of the issuer's fiscal year, of the effectiveness of the internal control structure and procedures of the issuer for financial reporting.

[http://www.aicpa.org/info/sarbanes\\_oxley\\_summary.htm](http://www.aicpa.org/info/sarbanes_oxley_summary.htm)

## VISA PCI/CISP

**PCI=Payment Card Industry**

**CISP=Cardholder Information Security Program**

**Mandatory since June 2001**

- CISP compliance is required for all merchants and service providers that store, process, or transmit Visa cardholder data.
- The program **applies to all payment channels, including retail.**
- To achieve CISP compliance merchants and service providers must implement PCI
- PCI/CISP is a result of a collaboration between Visa and MasterCard and is
- Other card companies operating in the U.S. have also endorsed the PCI Data Security Standard within their respective programs.
- If a member, merchant or service provider does not comply with the security requirements or fails to rectify a security issue, Visa may fine the responsible member and/or impose restrictions on the merchant or its agent.
- Members receive protection from fines for merchants or service providers that have been compromised but found to be CISP-compliant at the time of the security breach.

## VISA PCI/CISP

### PCI Guidelines:

1. Install and maintain a firewall configuration to protect data
2. Do not use vendor-supplied defaults for system passwords and other security parameters
3. Protect stored data
4. Encrypt transmission of cardholder data and sensitive information across public networks
5. Use and regularly update anti-virus software
6. Develop and maintain secure systems and applications
7. Restrict access to data by business need-to-know
8. Assign a unique ID to each person with computer access
9. Restrict physical access to cardholder data
10. Track and monitor all access to network resources and cardholder data
11. Regularly test security systems and processes
12. Maintain a policy that addresses information security

## ISO 17799 – The International Security Standard

First Published in 2002, revised in 2005

A comprehensive set of controls comprising best practices in Information Security

- Security Policy
- System Access Control
- Computer & Operations Management
- System Development and Maintenance
- Physical and Environmental Security
- Compliance
- Personnel Security
- Security Organization
- Asset Classification and Control
- Business Continuity Management (BCM)

<http://www.iso.org/iso/en/commcentre/pressreleases/2005/Ref963.html>

## ITIL

Originally developed by the British government,

ITIL ®, is a series of documents that are used to aid the implementation of a framework for IT Service Management (ITSM).

**12 main ITIL processes:** Availability Management; Capacity Management; Configuration Management; Change Management; Financial Management; **Incident Management; IT Service Continuity Management;** Problem Management; Release Management; **Security Management;** Service Desk; Service Level Management.

This is not free, they charge for the documentation!

<http://www.itil-toolkit.com/>

## COBIT

Now version 4.0

COBIT is an IT governance framework and supporting toolset that allows managers to bridge the gap between control requirements, technical issues and business risks.

4 domains:      Planning and Organization  
                    Acquisition and Implementation  
                    Delivery and Support  
                    Monitoring

COBIT is typically used as a SOX control framework  
Some COBIT objectives can be achieved via OWASP controls

## OWASP

### Guide “Black hat 2.0” published in 2005

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software.

It is an open source project that produces free, open-source documentation, tools, and standards.

The OWASP community also organizes conferences, local chapters, articles, papers, and message forums.

The OWASP Foundation is a not-for-profit charitable organization that ensures the ongoing availability and support for the work.

Participation in OWASP is free and open to all.



<http://www.owasp.org/index.jsp>

## General Secure (good) Practices

- 1) Adopt a versioning system (for example CVS)
- 2) Adopt a coding methodology (design / test / document)
- 3) Adopt a coding standard
- 4) Classify the assets (data) to be protected
- 5) Minimize Attack Surface Area
- 6) Keep it simple
- 7) Enforce authentication of all users
- 8) Perform separation of duties
- 9) Apply “Principle of Least Privilege”
- 10) Apply “Principle of Defense in Depth”
- 11) Secure the default behavior
- 12) Make sure you fail securely
- 13) Assume external systems are insecure
- 14) Do not use Security through Obscurity

## General Secure (good) Practices

### 1) Adopt a versioning system (for example CVS)

A versioning system is an application that records changes you make to your code.

- The code is “**checked out**” from a repository
- The code is **modified** by the programmers
- The new code is “**checked in**” together with a description of the changes

The versioning system stores all past changes with relative dates and descriptions  
It also notifies the users about potential conflicts due to forks.

The simplest versioning system is RCS. RCS has two commands

```
co [filename]  
ci [filename]
```

CVS is an extension of RCS. It can check in/out entire folders.

## General Secure (good) Practices

### 2) Adopt a coding methodology (design / test / document)

Examples:

- *Waterfall*
- *Incremental*
- *Spiral*
- *Agile*
- *Extreme-Programming*

Suggestions:

- divide your application in “modules” where each module comprises a minimal but complete set of functionalities.
- Write the specifications of the module before coding each module then,
- have at least two people code each module.
- Have somebody else test each module.
- Integrate the modules
- For web application it is important to be able to modify specs dynamically

## General Secure (good) Practices

### 3) Adopt a coding standard

- Describe the directory structure (what is where)
- Document each function / minimize in-line comments
- Code a test function/method for each module/class
- Adopt a naming convention for classes/methods/functions/objects/variables
- Adopt a convention for how functions/methods should report errors
- Adopt a convention for how errors should be reported to the user
- Adopt a convention for the default behaviour of your system
- ...
- Isolate authentication code from rest of code
- Minimize use of cookies
- Use session variables
- Validate all user input
- Escape all user input that is displayed in a web page
- ...

## General Secure (good) Practices

### 4) Classify the assets (data) to be protected

- Make a list of your database tables and relative fields
- For each field, for each table, document who has access to that file

Example:

Table Users	
Field Name	only user can read it
Field Username	only authentication function can read it
Field Password	only user can write it, only authentication func can read
Field Balance	user can read it, sell func can read/write

Table Items	
Field Name	everybody can read
Field InStock	everybody can read, sell func can read/write
Field Cost	everybody can read

## **General Secure (good) Practices**

### **5) Minimize Attack Surface Area**

- Assume each exposed functionality is a potential security risk (for example SQL injections, XSS, etc.)
- Limit access to functionalities as much as compatible with requirements
- Make sure all user input is validated

## **General Secure (good) Practices**

### **6) Keep it simple**

- Keep both the code design and the user interface as simple as possible
- It may be convenient to draw a graph where each vertex is a possible web page and arrows represents links (including forms) connecting one page to another.
- Each page can have multiple arrows in and multiple arrows out but should not have multi arrows in and multiple arrows out (bad design).
- Make sure each page validates its input and checks for the referrer page.
- The code each page separately by calling common functions/methods.

## **General Secure (good) Practices**

### **7) Enforce authentication of all users**

- Any web application should have a public side (which exposes text but no functionalities and/or forms) and restricted side that requires a one-time authentication (login/password/token/etc)
- Authentication must be done via an encrypted channel (ssl)
- Authentication information should not be sent back to the user (with web or email)
- If public access is required force a free registration and send a one-time password to the registrant by email or SMS. Then record the email address.
- If anonymous access is required store the remote ip address
- Log all login attempts (success and failure)

## **General Secure (good) Practices**

### **8) Perform separation of duties**

- Administrators should not be users of the application (this does not mean that the same person cannot have two logins, one as user and one as administrator).
- Administrators should not be able to buy good or communicate on behalf of other users.
- For more complex web application where users can have various roles (buyers/sellers/etc.) create different users types to prevent the same user to assume conflicting roles (for example order an item and certify that a payment has been received).

## **General Secure (good) Practices**

### **9) Apply “Principle of Least Privilege”**

- Design your application around a hierarchy of privileges. The user with more privileges can do more than the user with less privileges.
- Assign each user the least privilege required to access the functionality/data he is supposed to access. This prevents the user from accidentally/maliciously accessing functionalities/data he/she is not supposed to access.
- Often a single number is sufficient to identify the privilege level of a user and the least required privilege level for each asset. For each user the number is stored in the TABLE USERS together with the username and hashed password.
- For more sophisticated systems (example: a web site to access classified info from the department of defense), a classification systems may be required in which privileges are typically identified by a vector of numbers.

## **General Secure (good) Practices**

### **10) Apply “Principle of Defense in Depth”**

- Even if a single control is better, more controls are better.

Example, to buy an item...

- The user must be logged in.
- The user must have logged in no longer than 15mins before
- Check that remote IP address is the same used at login
- Check the referrer page is valid
- If item is very expensive, ask the user to login again and verify purchase by phone or email

## General Secure (good) Practices

### 11) Secure the default behavior

- Force use of SSL for all sensitive information, this should not be an “optional” feature.
- Force users to adopt complex passwords (long, mixed cases with numbers)
- Force users to change passwords
- Disable users than don’t change passwords when requested and/or don’t use the system for long time
- Validate all fields
- Escape all output
- Log all important successful and unsuccessful transactions.

## General Secure (good) Practices

### 12) Make sure you fail securely

Avery operation can fail in two ways:

- *Invalid use action (invalid field for example)*
  - *Error in code*
- Notify the user of any invalid action (invalid field) and revert to a state equal to the state before the action was initiated
  - In case of any error in code, log the error, email the administrators, perhaps release a ticket to the user, logout the user. Make sure all code is in a **try... catch...** in order to achieve this.

## **General Secure (good) Practices**

### **13) Assume external systems are insecure**

- If your application talks to a database or to another application don't assume they are secure.

Examples:

- Your application passes a user's field to a second application that queries an SQL database; don't assume the second application will check the field for SQL injections. Do the check before calling the second application.
- Your application receives data from a database filled by a second application and displays the data. Don't assume the data is in a format valid for display in text/html, always escape data.

## **General Secure (good) Practices**

### **14) Do not use Security through Obscurity**

Don't assume that, if your code has a vulnerability, if you don't tell anybody, nobody will find out. Please accidentally or maliciously do find out.

Consider the case of vulnerabilities in Microsoft Explorer and/or IIS. New vulnerabilities are found weekly even if they are closed source projects.

Consider the use of well-established open source systems. Even if it is easier to find vulnerabilities there, they are usually fixed in a short time.

## **Topic: Threat Risk Modeling**

**ECT582**

**Secure  
Electronic  
Commerce**



## **Threat Risk Modeling**

Goals:

- Determine Controls
- Produce effective countermeasures

Thread Model Flow:

1. Identify Security Objectives
2. Application Overview
3. Decompose Application
4. Identify Threats
5. Identify Vulnerabilities
6. Go to 2.

## Threat Risk Modeling

### 1) Example of Security Objectives

- Protect identity of the users
- Protect reputation in case of successful attack
- Minimize financial loss in case of successful attack
- Enforce applicable regulations (Bank? Credit cards?)
- Enforce applicable standards (ISO17799)

## Threat Risk Modeling

### 2) Decompose Application, 3) Identify Threats, 4) Identify Vulnerabilities

- Is data validation implemented?
- Is it correct?
- Is it sufficient?
- Is the workflow enforced?
- What kind of authorization checks are performed?
- What information is stored client side?
- What information is stored server-side (session)? And who can retrieve it?
- What information is transmitted in the clear (GET fields)?
- ...
- Who are the potential attackers?
- What are their motivations?

## Threat Risk Modeling: STRIDE/DREAD

### Microsoft Threat's Modeling Process

#### STRIDE COMPONENT:

- Users should not be able to act as other users, nor become administrators
- Applications should not send data to user, only keys
- Application should minimize data sent to browser (cached)
- The application should have repudiation controls (log at every tier)
- Application should prevent DoS attack by limiting complex operations

## Threat Risk Modeling: STRIDE/DREAD

#### DREAD COMPONENT: $\text{Risk} = ([1] + [2] + [3] + [4] + [5]) / 5$

- [1] Damage Potential (0=nothing, 10=complete disruption)
- [2] Reproducibility of Threat (0=impossible, 10=just a browser and the URL)
- [3] What is needed for Exploitability (0=advanced programming skills, 10=browser)
- [4] How many users would be affected? (0=none, 10=all)
- [5] How easy is it to discover this threat (0=impossible, 9=using Google, 10=from address bar).

Note: Since Google stores pages and URLs it is possible to use Google, to some extent, to search for web site exposing some known vulnerabilities.

## STRIDE/DREAD Example

“An online auction has a vulnerability that allows a seller to buy the very same item he is selling at ½ the price he will receive for the sale. This vulnerability is only known to the coders of the system and requires using a backdoor via the url.”

- [1] = 5 Damage Potential (0=nothing, 10=complete disruption)  
(a user who knows about it can make a profit without actually selling the item)
- [2] = 0 Reproducibility of Threat (0=impossible, 10=just a browser and the URL)
- [3] = 10 What is needed for Exploitability (0=advanced programming skills, 10=browser)
- [4] = 0 How many users would be affected? (0=none, 10=all)
- [5] = 0 How easy is it to discover this threat (0=impossible, 9=using Google, 10=from address bar).

**DREAD COMPONENT:** Risk = ( [1] + [2] + [3] + [4] + [5] ) / 5 = 3

## STRIDE/DREAD Example

“An online bank allows any logged user to mingle the URL and obtain information about other accounts not owned by the user”

- [1] = 0 Damage Potential (0=nothing, 10=complete disruption)  
(there is no damage to the data even if once public this will have consequences)
- [2] = 10 Reproducibility of Threat (0=impossible, 10=just a browser and the URL)
- [3] = 10 What is needed for Exploitability (0=advanced programming skills, 10=browser)
- [4] = 10 How many users would be affected? (0=none, 10=all)
- [5] = 10 How easy is it to discover this threat (0=impossible, 9=using Google, 10=from address bar).

**DREAD COMPONENT:** Risk = ( [1] + [2] + [3] + [4] + [5] ) / 5 = 8

## Threat Risk Modeling

### Alternative Threat Modeling schemes

- Trike (similar to STRIDE/DREAD)
- AS/NZS 4360 (Australian standard)
- CVSS (Common Vulnerability Scoring System) created by the US Department of Homeland Security and endorsed by Cisco, Symantec, Microsoft, eBay, etc.

Required if you work with Government and preferred by researchers.

Complex and not designed to avoid design flaws (ranking only)

- Octave from CMU. Design to evaluate business risk, not technical risk.

## Top Ten Vulnerabilities

- 1. Unvalidated Input:** Information from web requests is not validated before being used by a web application. Attackers can use these flaws to attack backend components through a web application.
- 2. Broken Access Control:** Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions.
- 3. Broken Authentication and Session Management:** Account credentials and session tokens are not properly protected. Attackers that can compromise passwords, keys, session cookies, or other tokens can defeat authentication restrictions and assume other users' identities.
- 4. Cross Site Scripting (XSS) Flaws:** The web application can be used as a mechanism to transport an attack to an end user's browser. A successful attack can disclose the end user's session token, attack the local machine, or spoof content to fool the user.

## Top Ten Vulnerabilities (continued...)

5. **Buffer Overflows:** Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of a process. These components can include CGI, libraries, drivers, and web application server components.
6. **Injection Flaws:** Web applications pass parameters when they access external systems or the local operating system. If an attacker can embed malicious commands in these parameters, the external system may execute those commands on behalf of the web application.
7. **Improper Error Handling:** Error conditions that occur during normal operation are not handled properly. If an attacker can cause errors to occur that the web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail, or crash the server.

## Top Ten Vulnerabilities (continued...)

8. **Insecure Storage:** Web applications frequently use cryptographic functions to protect information and credentials. These functions and the code to integrate them have proven difficult to code properly, frequently resulting in weak protection.
9. **Denial of Service:** Attackers can consume web application resources to a point where other legitimate users can no longer access or use the application. Attackers can also lock users out of their accounts or even cause the entire application to fail.
10. **Insecure Configuration Management:** Having a strong server configuration standard is critical to a secure web application. These servers have many configuration options that affect security and are not secure out of the box.

## **Topic: Data Validation and Escaping**

**ECT582**

**Secure  
Electronic  
Commerce**



## **Data Validation**

- 1) **Integrity Checks:** Ensure that data is the same as before and has not been tampered with
- 2) **Validation:** Ensure that data is strongly typed, correct syntax, within legal boundaries, contains only permitted characters or, if numeric, the sign is right.
- 3) **Business rules:** Ensure that data is not only validated but business rules are correct. For example interest rate falls within permitted boundaries.
- 4) **Enforce navigation/workflow and prevent double submission.** User should not be able to submit the same form twice and skip forms by tampering with URL.
- 5) **Minimize the amount of information stored client-side (cookies)**

## Data Validation, where?

### Where to put data validation?

Integrity checks should be placed every time data passes from a trusted to a less trusted boundary or vice versa, for example in multi tier system.

Validation and business rules must be performed at every tier. For example, every time user data is stored or used.

Always validate cookies. Cookies should be used only (optionally) for storing session IDs and for tracking customers (eventually). Cookies should not be used for anything else!

## Data Validation

1) Check that the input make sense ([validation](#))

2) If input is displayed back to the user, [escape](#) input appropriately

- a) input may be displayed in HTML
- b) input may be used to build an URL (different type of escaping)
- c) input may be rendered inside an HTML tag
- d) input may be used inside Javascript
- e) input may be used to access filesystem (directory traversal attacks)
- f) ...

3) If input is used to build an SQL or other type query, remove control characters that may lead to injection ([avoid SQL injection](#))

## Data Validation

### Example From:

```
insert your SSN#:<input type=text name=ssn>
```

### Target:

```
<?validate_ssn(request.ssn)?>
```

```
def validate_ssn(text):
    if re.match("^\d{3}\-\d{2}\-\d{4}$", text)==false:
        return false
    return true

^      starting at the beginning of line
\d{3} exactly {3} digits \d
\-
      followed by a dash
\d{2} followed by exactly {2} digits
\-
      followed by a dash
\d{4} followed by exactly {4} digits
$      followed by end of line
```

## Data Validation

### Example From:

```
insert your phone n.:<input type=text name=phone>
```

### Target:

```
<?validate_phone(request.phone)?>
```

```
def validate_phone(text):
    if re.match(
        "^(1\-)?\d{3}\-(\d{3})\-(\d{4})$",
        text)==false:
        return false
    return true

^(1\-)?      an optional 1-
\d{3}\-      followed by area code and dash (ddd-)
\d{3}\-\d{4} followed by ddd-dddd
$      followed by end of line
```

## Data Validation

### Example From:

```
insert your zip code:<input type=text name=zip>
```

### Target:

```
<?validate_zip(request.zip)?>
```

```
def validate_zip(text):
    if re.match("^\d{5}(\-\d{4})?",text)==false:
        return false
    return true

^      starting at the beginning of line
\d{5}  exactly {5} digits
(      (
\-      followed by a dash
\d{4}  and {4} digits
)?     ) optionally,
$      followed by the end of line
```

## Data Validation

### Example From:

```
insert your emial addr.:<input type=text name=email>
```

### Target:

```
<?validate_email(request.email)?>
```

```
def validate_email(text):
    if re.match(
        "^\[\w-]+\(:\.\[\w-]+\)*@(\?:[\w-]+\.)+[\a-zA-Z]{2,7}\.\?$/,
        text)==false:
        return false
    return true
```

```
massimo.dipierro@cs.depaul.com
any repetition of characters, digits and underscores [\w-]+
followed by any repetition of \.\[\w-]+\+
followed by @
followed by any repetition of [\w-]+\.
followed by [\a-zA-Z]{2-7} (top level domain "com")
```

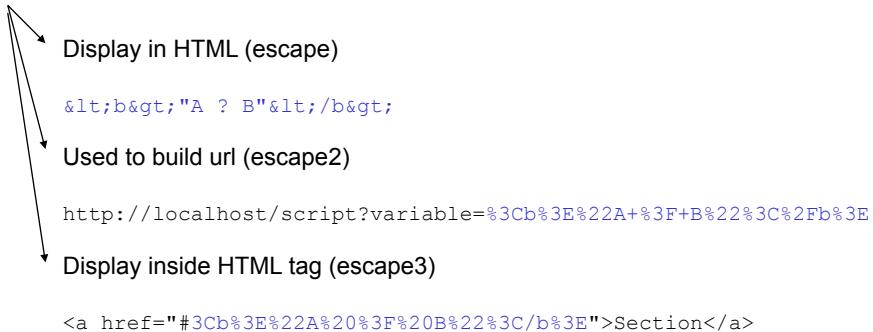
## Data Validation

```
URL      ^((https?://)([0-9A-Za-z]{2}|[-()_.!~*'@:&=+\W,A-Za-z0-9]+)([]!.!';/?:[[:blank:]])?)$  
EMAIL    ^[\w-]+(?:\.\w-+)*@[^\w-]+\.\w{2,7}$  
TEXT     ^[a-zA-Z0-9\s.-]+$  
ZIP      ^\d{5}(\-\d{4})?$_  
PHONE    ^(\(1\-\)?)\d{3}\-\(\d{3}\)\-\(\d{4}\)$  
C.CARD   ^((4\d{3})|(5[1-5]\d{2})|(6011))-?\d{4}-?\d{4}?\d{4}|3[4,7]\d{13}$  
PASSWD   ^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{4,8}$  
SSN      ^\d{3}\-\d{2}\-\d{4}$  
FL.PNT.  ^[-+]?[0-9]+[.]*[0-9]*([eE][-+]?[0-9]+)?$_  
NAME     ^[a-zA-Z]+(([\\',\\.\\- ]|[a-zA-Z ]*)*[a-zA-Z]*)$_
```

## Escaping

### User input

text:<b>"A ? B"</b>



## Data Validation

User input

text:<b>"A ? B"</b>

→ Display in HTML (escape)

&lt;b&gt;"A ? B"&lt;/b&gt;

<        &lt;  
>        &gt;  
&        &amp;

[http://www.webmonkey.com/webmonkey/reference/special\\_characters/](http://www.webmonkey.com/webmonkey/reference/special_characters/)

In PHP: htmlspecialchars() and/or htmlentities()

In ASP: HTMLEncode()

In JSP: HTMLEncode()

In Python: cgi.escape()

## Data Validation

User input

text:<b>"A ? B"</b>

→ Used to build url (escape2)

http://localhost/script?variable=%3Cb%3E%22A+%3F+B%22%3C%2Fb%3E

space +  
any % [ASCII]  
< %Cb

<http://www.december.com/html/spec/escrcodes.html>

In PHP: urlencode()

In ASP: URLEncode()

In JSP: URLDecoder.encode()

In Python: urllib.quote\_plus()

## Data Validation

User input

```
text:<b>"A ? B"</b>
```

→ Display inside HTML tag (escape3)

```
<a href="#3Cb%3E%22A%20%3F%20B%22%3C/b%3E">Section</a>
```

```
any    % [ASCII]  
<     %Cb  
space  %20
```

<http://www.blooberry.com/indexdot/html/topics/urlencoding.htm>  
RFC1738

## Itemized lists

### The bad

```
<select name=color>  
<option value=1>red</option>  
<option value=2>blue</option>  
<option value=3>green</option>  
</select>
```

### Target:

```
<?colors=['red', 'blue', 'green']?>  
<?i=int(request.color)?>  
<?if i<1 or i>3: error()?>  
<font color=<?(colors[i])?>XXXXX</font>
```

## Itemized lists

### The ugly

```
<select name=color>
<option value=1>red</option>
<option value=2>blue</option>
<option value=3>green</option>
</select>
```

### Target:

```
<?colors=['red', 'blue', 'green']?>
<?try:
    i=int(request.color)
    if i<1 or i>3: error()
except SyntaxError:
    error()
pass?>
<font color=<?(colors[i])?>XXXXX</font>
```

## Itemized lists

### The bad

```
<select name=color>
<option value=red>red</option>
<option value=blue>blue</option>
<option value=green>green</option>
</select>
```

### Target:

```
<font color=<?(request.color)?>XXXXX</font>
```

## Itemized lists

### The ugly (not so ugly but error prone)

```
<select name=color>
<option value=red>red</option>
<option value=blue>blue</option>
<option value=green>green</option>
</select>
```

### Target:

```
<?colors=['red', 'blue', 'green']?>
<?if not request.color in colors: error()?>
<font color=<?(request.color)?>XXXXXX</font>
```

## Itemized lists

### The good

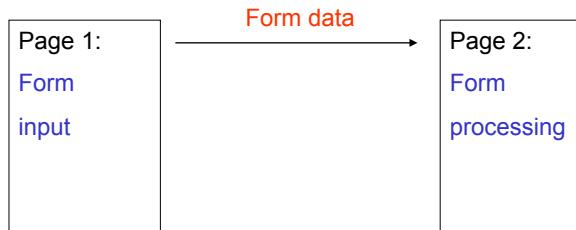
```
<select name=color>
<option value=1>red</option>
<option value=2>blue</option>
<option value=3>green</option>
</select>
```

### Target:

```
<?colors=['red', 'blue', 'green']?>
<?if not request.color in ['1','2','3']: error()?>
<font color=<?(colors[int(request.color)])?>XXXXXX</font>
```

## Enforce Navigation/Workflow

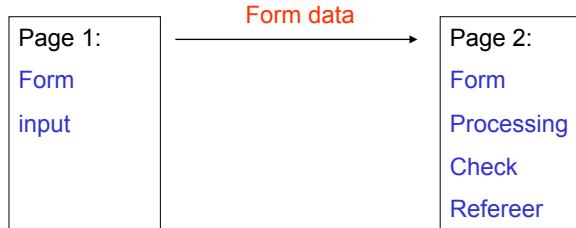
Naïve:



Problem: page 2 can be reached without going through page 1 first.  
This allows for brute force and other types of attacks.

## Enforce Navigation/Workflow

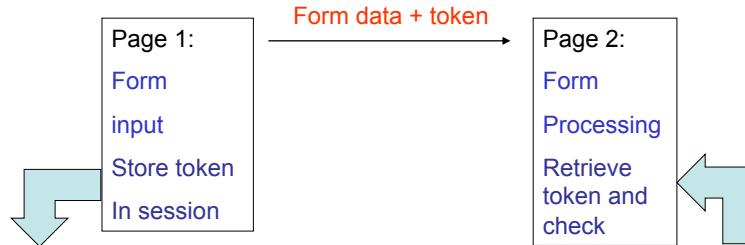
Better:



Bad solution: check refereer page. The refereer info is passed by the browser  
And therefore it is not reliable information.

## Enforce Navigation/Workflow

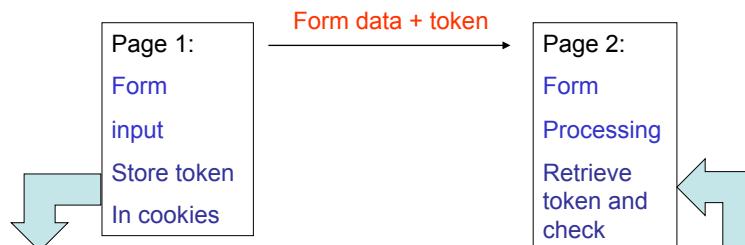
Best:



Solution, if token is random, page 2 will not be able to validate form  
Unless page 1 has been visited.

## Enforce Navigation/Workflow

Looking for trouble:



Another bad solution because attacker can edit its own cookies.

## Itemized lists

### Form (page 1):

```
<?session.token=str(random())?>
<form action="...page2">
<input type=hidden name=token value=<?(session.token)?>>
... rest of form
</form>
```

### Target (page 2):

```
<?is not request.token==session.token: error('invalid form')?>
<?session.token=str(random()) #reset session.token?>
... process form
```

## View – Controller - Model

Organize your application by dividing its components in three categories

### View: the front-end rendering code

This is the code that produces the actual HTML/Javascript/AJAX returned to client

### Controller: application logic

This is the code that validates the input, the workflow, and escapes the output

### Model: the functionality

A set of functions/methods encoded in a library called by the controller. They should be high level functions that encapsulate all the dirty work and testable independently on the view and the controller.

## Topic: Vulnerabilities and Attacks

**ECT582**

**Secure  
Electronic  
Commerce**



## Vulnerabilities and fixes of Web Apps.

1. **Brute force attacks (for example loop and try different passwords):**  
firewall + enforce password complexity + log and block attempts
2. **Injection (for example SQL injection):**  
validation
3. **XSS (for example web site defacement):**  
escaping + no frames
4. **Directory traversal:**  
validation
5. **Remote Command Execution:**  
do it, you have to, use strong validation
6. **Buffer Overflow:**  
don't validate string size + intrusion detection system
7. **Session hijacking:**  
make id unpredictable + validate IP + remove session at logout +  
do not store information in cookies

## Brute force attack 1/2

### Automate a POST request

```
def POST(hostname,file,**ka):
    fields=''
    for key in ka.keys():
        fields=fields+escape2(key) +'=' +escape2(ka[key]) + '& '
    pass
    if ka: fields=fields[:-1]
    request='POST /%s HTTP/1.1\nContent-Length: %i\n\n%s' % \
        (file, restofheader,len(fields),fields)
    print request
    s=socket.socket()
    s.connect((hostname,80))
    s.send(request)
    return s.recv(100000)
pass

print POST('www.google.com', 'search',hl='en',q='fermiqcd')
```

## Brute force attack 2/2

### Loop and try different passwords from a dictionary

```
fail_html=POST('www.targetbank.com', 'login',
               username='target',password='')

for word in dictionary:
    response=POST('www.targetbank.com', 'login',
                  username='target',password=word)
    if response!=fail_html:
        print word
        break
    pass
pass
```

## SQL Injection 1/3

### Example From

```
username: <input type=text name=username>
password: <input type=password name=password autocomplete=off>
```

### Target

```
query=SELECT FROM USERS WHERE USERNAME='%s' AND PASSWORD='%s'
```

### Attack

```
username:mdp password:test
query=SELECT FROM USERS WHERE USERNAME='mdp' AND
    PASSWORD='test'
(query is always true)
```

### Solution:

check for invalid characters (for example spaces from password) and  
in both username and password `replace("\\\", "\\\\"")`  
(specific syntax depends on the language)

## SQL Injection 2/3

### Example From

```
username: <input type=text name=username>
password: <input type=password name=password autocomplete=off>
```

### Target

```
query=SELECT FROM USERS WHERE USERNAME='%s' AND PASSWORD='%s'
```

### Attack

```
username:mdp password:' OR PASSWORD<>'
query=SELECT FROM USERS WHERE USERNAME='mdp' AND
    PASSWORD=''' OR PASSWORD<>'
(query is always true)
```

### Solution:

check for invalid characters (for example spaces from password) and  
in both username and password `replace("\\\", "\\\\"")`  
(specific syntax depends on the language)

## SQL Injection 3/3 (PHP)

### Example From

```
username: <input type=text name=username>
password: <input type=password name=password autocomplete=off>
```

### Target in PHP (example from Olawale)

```
<?php
$link = mysql_connect('mysql_host', 'mysql_username',
'mysql_password') OR die(mysql_error());

$query = sprintf("SELECT * FROM Users WHERE username='%s' AND
password='%s'",
mysql_real_escape_string($username),
mysql_real_escape_string($password));
?>
```

## XSS web face defacement

### Attacker steps

1. Create a copy of the page to be defaced
2. Modify the copy
3. Post the copy on some server (other then the one hosting the original)
4. Insert one of the following strings in the site to be defaced

### Pure HTML XSS

```
<META HTTP-EQUIV="Refresh" CONTENT="0;URL=http://defaced/">
```

### Javascript HTML XSS

```
<SCRIPT>self.location=http://defaced/</SCRIPT>
```

They force the browser of a visitor to reload the defaced page in place of the original.

If the original used frames, the URL will show the original and not the defaced URL.

## Directory traversal 1/2

`http://www.target_host.com/path/name.html`

### Reads

Return the file name.html located at Path on [www.target\\_host.com](http://www.target_host.com)

### Attack

`http://www.target_host.com/./etc/passwords`

Return the file passwords located on the system directory ./etc/

Directory traversal strings depend of the web server, operating system, and configuration

## Directory traversal 2/2

`http://www.target_host.com/cgi-bin/script?r=name`

### May mean

Return the file name located on [www.target\\_host.com](http://www.target_host.com)

### Attack

`http://www.target_host.com/cgi-bin/script?r=.%2Fetc%2Fpasswords`

Returns the file ./etc/passwords

Where `urlescape("./etc/passwords")` is `.".%2Fetc%2Fpasswords"`

## Remote command execution 1/2

### Example From

```
formula: <input type=input name=formula>
```

### Target Page

```
You formula = <?print eval(request.formula)?>
```

The target page executes any command passed in the request including things like:

- open('./etc/passwords','r').read()
- open('./etc/passwords','a').write('phantom:asfasdfgasdg')
- system('format c:')

### Solution

Don't ever call the interpreter in a script

## Remote command execution 2/2

### More typical examples

#### From

```
filename: <input type=input name=filename>
```

#### Target Page

```
<?system("rm "+request.filename)?>  
File <?(request.filename)?> has been deleted
```

The target page executes any command passed in the request including things like:

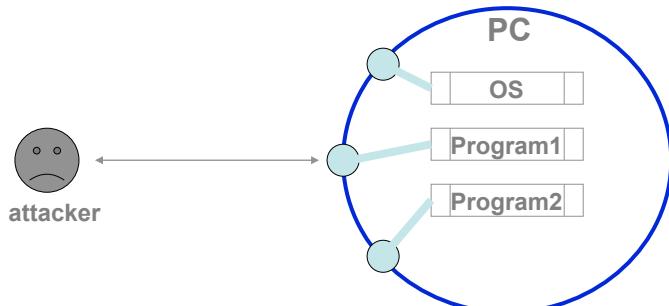
- -help; less otherfile

#### Solution

- Avoid calling the shell from a script
- Validate the input

## Buffer Overflow

How does the attacker or a malicious program enters in a computer without a password  
And/or without using email attachments?



It exploits vulnerabilities in programs (or the OS), such as buffer overflow.

## Buffer Overflow

In computer programming, a **buffer overflow** is an anomalous condition where a program somehow writes data beyond the allocated end of a buffer in memory. Buffer overflows usually arise as a consequence of a bug, and are a commonly exploited computer security risk — since program code often sits in the memory areas adjacent to data buffers, by means of a buffer overflow condition the computer can be made to execute arbitrary (and potentially malicious) code that is fed to the buggy program as data.

[http://www.windowsecurity.com/articles/Analysis\\_of\\_Buffer\\_Overflow\\_Attacks.html](http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html)

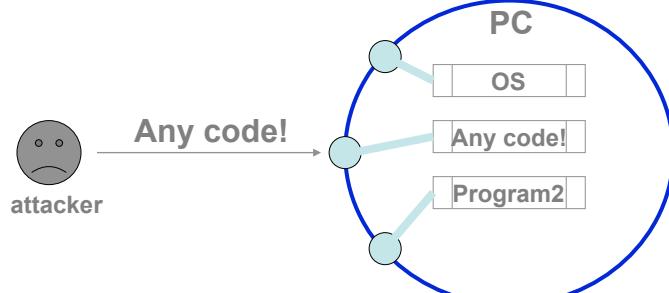
## Buffer Overflow

Buffer overflow is the **single most dangerous source of danger**.

Buffer overflow is **consequence of sloppy programming**.

If there is a buffer overflow vulnerability an attacker can execute any code he wants with the privileges of the program that has the vulnerability.

A buffer overflow vulnerability in a SUID program would allow the attacker to become root.



## Buffer Overflow

emacs

```
• void foo(char *s) {  
•     char buf[10];  
•     strcpy(buf,s);  
•     printf("buf is %s\n",s);  
• }  
• ...  
• foo("thisstringistolongforfoo");
```

memory

allocated for buf

thisstringistolongforfoo

## Buffer Overflow

emacs

```
• FILE* fstream;
• ...
• char buf[10];
• fscanf(fstream,"%[^\\0]s",buf)
```

memory

allocated for buf

thisstringistolongforfoo

thisstringistolongforfoo

## Topic: Authentication/authorized

# ECT582

Secure  
Electronic  
Commerce



## **Authentication/authorization**

### **Authentication**

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. This is done by verifying user's credentials.

### **Authorization**

Authorization is the process of giving someone permission to do or have something. In multi-user computer systems, a system administrator defines for the system which users are allowed access to the system and what privileges of use (such as access to which file directories, hours of access, amount of allocated storage space, and so forth).

## **Authentication/authorization**

No authentication technique should be used without SSL

### **Authentication techniques**

- Form based authentication: weaknesses: replay attacks, man in the middle attacks, clear text credentials, weak password controls, luring attacks
- Integrated authentication (IIS + ASP.NET on intranets)
- Certificate based authentication. OK for business2business, not business2customer

### **Authentication means**

- Something you know (for example a password)
- Something you have (for example a token/certificate/cryptocard)
- Something you are (for example biometrics)

Strong authentication = password + token/certificate

## Authentication/authorization

### Replay attack

A replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed. This is carried out either by the originator or by an adversary who intercepts the data and retransmits it, possibly as part of a masquerade attack.

All authentication techniques except those that require a one-time token or a one-time password are vulnerable to replay attacks.

### Solutions

Use Challenge response systems (crypto card or SMS-challenge-response)

## Authentication methods

### Example Form

```
username: <input type=text name=username>
password: <input type=password name=password autocomplete=off>
```

### A wrong login script

```
<?
session.authorized=true
if not in_database(request.username, request.password) :
    print "Invalid Login"
    session.authorized=false
else:
    print "you are logged in"
pass
?>
```

Error: in case of error session.authorized is true by default

## Authentication methods

### Another wrong login script

```
<?
if not in_database(request.username, request.password) :
    print "Invalid Login"
    session.authorized=false
else:
    print "you are logged in"
    session.autjorized=true
pass
?>
```

Error: in case of error session.authorized is undefined

## Authentication methods

### Better login script

```
<?
session.authorized=false
if not in_database(request.username, request.password) :
    print "Invalid Login"
else:
    print "you are logged in"
    session.authorized=true
pass
?>
```

A bug could expose authentication code

## Authentication methods

### Best login script

```
<?
session.authorized=false
try:
    if not in_database(request.username, request.password):
        raise Exception
    else:
        print "you are logged in"
        session.authorized=true
    pass
except:
    print "Invalid Login"
    session.authorized=false
pass
?>
```

## Password complexity

- Enforce password complexity (8-10 characters, mixed case, numbers, symbols)
- Force users to change password frequently (every 3 months)
- Notify users by email that they need to change password
- Disable accounts of users that do not change passwords
- Never store passwords in the clear, encrypt (hash) passwords before storage
- Do not implement a password retrieval mechanism but only a password reset mechanism
- If you implement self-registration, consider using CAPTCHA (NOTE: any web site that is mandated or legally required to be accessible must not use CAPTCHA).
- If you implement self-registration, verify identity by SSN or other ID.

## Authorization objectives and requirements

### Objectives

- Ensure only privileged users can perform allowed actions within their privilege level
- To control access to protected resources using decisions based upon privilege level
- To prevent privilege escalation

### Requirements

- Web-Applications should run with the lowest possible privilege (user nobody or www-data)
- On Unix systems use chroot
- Ideally, run the application inside a Virtual Machine
- Database access only through stored parameterized functions
- Change default settings of your components

## Session management

- Don't use databases for session variables but use an out-of-the-box session manager (most frameworks have one)
- Use unpredictable session IDs (to prevent session hijacking)
- Only store session ID into cookies, all session variables must be stored on server
- Always validate the session ID upon retrieval of session data
- Close session at logout and do not implement "remember me" mechanisms
- Implement a session expiration mechanism (timeout)
- Store information about the owner of the session in the session (such as client IP address) and verify the information at each request within the same session.

**Topic: Other Threads not App. Specific**

# ECT582

**Secure  
Electronic  
Commerce**



**Other Threats not App. related**

1. **Interception of traffic (Ethereal, tcpdump)**  
Use HTTPS
2. **Buffer Overflow:**  
validate string size + intrusion detection (IDS)
3. **Phishing:**  
?
4. **Viruses and Worms**  
Install antivirus and keep rules updated
5. **Spyware**  
Install anti-spyware and keep rules updated
6. **Zombie networks**  
(a way to do DoS) Monitor your network with IDS
7. **DNS cache poisoning**  
Consider static cache, monitor local traffic

## **Firewall**

A firewall is a set of related programs, located at a network gateway server, that protects the resources of a private network from users from other networks.

An enterprise with an intranet that allows its workers access to the wider Internet installs a firewall to prevent outsiders from accessing its own private data resources and for controlling what outside resources its own users have access to.

Basically, a firewall, working closely with a router program, examines each network packet to determine whether to forward it toward its destination. A firewall also includes or works with a proxy server that makes network requests on behalf of workstation users.

## **Intrusion Detection Software**

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

IDS differs from a firewall in that a firewall looks out for intrusions in order to stop them from happening. The firewall limits the access between networks in order to prevent intrusion and does not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system.

<http://www.snort.org/pub-bin/downloads.cgi>

## Anti-virus Software

An anti-virus software program is a computer program that attempts to identify and eliminate computer viruses and other malicious software (malware).

Anti-virus software typically uses two different techniques to accomplish this:

Examining (scanning) files to look for known viruses matching definitions in a virus dictionary

Identifying suspicious behavior from any computer program which might indicate infection

Most commercial anti-virus software uses both of these approaches, with an emphasis on the virus dictionary approach.

A good virus database is maintained by **Symantec**:

<http://www.symantec.com/avcenter/vinfodb.html>

<http://clamav-du.securesites.net/cgi-bin/clamgrok>

## VBS Viruses and Worms

Basically, think about **VBScript as a super batch language**. VBScript is an interpreted language (so scripts are really the source code for whatever needs to be done). Scripts can be embedded into such things as web pages or can be standalone files (with the extension .VBS usually).

If you've got Microsoft's Internet Explorer 5 browser on your system it's likely you also have the Windows Scripting Host (WSH) which is the program used to interpret and run VBS scripts.

Even though VBScript is a scaled down language it is quite capable and can be used to, connect to Microsoft's Outlook mail routines and send files to anyone in your address book. This, of course, makes it possible for VBScript to be a language used by worms to spread themselves.

**VBH can be disabled on your system**

## How to secure a host

- 1) **Implement strong policies** (make backups, force users to use strong passwords and change them often, forbid users to install programs, regulate and minimize access to resources, keep system updated)
- 2) **Only run trusted programs** (do not use unnecessary programs, only get programs from trusted sources, check digital signatures)
- 3) **Minimize network access** (uninstall unwanted services, close unused ports)
- 4) **Monitor and audit** suspicious activities
- 5) **Do not trust** software vendors (easy of use != secure)

## Topic: Logging and Auditing

**ECT582**

**Secure  
Electronic  
Commerce**



## Logging requirements

Many industries are required by law to be:

**Auditable:**

All activities that affect user state and balances are tracked

- Application must record all accesses (success and failure), all transactions, all errors (application must fail safe, log the error but not expose error info to the user other than a ticket number).

**Traceable:**

It must be able to trace the execution of any activities at all tiers

- Example: Log system logins, application logins, database logins, etc.
- Other example: Log web server requests, application requests, database requests, etc.

**Highly Integrity:**

Logs must be protected against tampering and accidents.

- Burn a CD-R with all logs in a daily basis. The CDs must be kept for 6 months.

## Logging tips

**Important:**

- Logfiles may have a maximum size. Make sure the size is large enough to protect against DoS attacks.
- Only Audit important events.
- Use a centralized logging facility and encrypt data during communication
- Hash the logs to prevent tampering with them.
- Never review the actual logs, always act on a copy of the logs.

## **Topic: Handling Payments**

**ECT582**

**Secure  
Electronic  
Commerce**



## **VISA PCI/CISP**

### **PCI Guidelines:**

1. Install and maintain a firewall configuration to protect data
2. Do not use vendor-supplied defaults for system passwords and other security parameters
3. Protect stored data
4. Encrypt transmission of cardholder data and sensitive information across public networks
5. Use and regularly update anti-virus software
6. Develop and maintain secure systems and applications
7. Restrict access to data by business need-to-know
8. Assign a unique ID to each person with computer access
9. Restrict physical access to cardholder data
10. Track and monitor all access to network resources and cardholder data
11. Regularly test security systems and processes
12. Maintain a policy that addresses information security

## **Handling Credit Cards**

- 1) If possible delegate handling to third party
- 2) Process transactions immediately and discard information as soon as possible
- 3) Do not store any information other than the authorization number returned upon processing. The latter must be logged.
- 4) Never display any portion of a Credit Card information, although PCI/CISP allow the display of the last 4 digits.

XXXX XXXY YYYZ YYYYC (X=Bank ID, Y=Account ID, Z=checksum)

(One could figure an account number in less than 100000 trials).

- 5) Never store the PVV (Pin Verification Value)

## **Handling Recurrent Credit Card Payments**

Example: subscription service.

This is the only case one may have to store Credit Card information.

- Get signed authorization from client
- Make sure you delete Credit Card information as soon as agreement expires
- Encrypt the Credit Card information

## **Handling Reversals**

Reversals: when funds are moved into a Credit Card Account (for example a refund or partial refund).

- This operation should always be authorized manually
- This operation should always be authorized by two different groups (to avoid internal fraud).

## **Handling chargebacks**

A special type of reversals, when the total charge amount is reversed.

- Same as reversal.

General rules:

- Validate Money (must be positive number)
- Never allow any money-handling function to accept negative values
- All chargebacks, reversals must be logged and require manual authorization
- There should be no online code/forms for chargebacks or reversals
- Consider placing restrictions based on CC Bank location
- For expensive goods, require phone authorization.

Credit Card Payment Glossary:

[http://www.echo-inc.com/credit\\_card\\_glossary.html](http://www.echo-inc.com/credit_card_glossary.html)

## Topic: Logging and Auditing

**ECT582**

**Secure  
Electronic  
Commerce**



## Logging requirements

Many industries are required by law to be:

**Auditable:**

All activities that affect user state and balances are tracked

- Application must record all accesses (success and failure), all transactions, all errors (application must fail safe, log the error but not expose error info to the user other than a ticket number).

**Traceable:**

It must be able to trace the execution of any activities at all tiers

- Example: Log system logins, application logins, database logins, etc.
- Other example: Log web server requests, application requests, database requests, etc.

**Highly Integrity:**

Logs must be protected against tampering and accidents.

- Burn a CD-R with all logs in a daily basis. The CDs must be kept for 6 months.

## **Logging tips**

### **Important:**

- Logfiles may have a maximum size. Make sure the size is large enough to protect against DoS attacks.
- Only Audit important events.
- Use a centralized logging facility and encrypt data during communication
- Hash the logs to prevent tampering with them.
- Never review the actual logs, always act on a copy of the logs.

## **Topic: Handling Payments**

**ECT582**

**Secure  
Electronic  
Commerce**



## VISA PCI/CISP

### PCI Guidelines:

1. Install and maintain a firewall configuration to protect data
2. Do not use vendor-supplied defaults for system passwords and other security parameters
3. Protect stored data
4. Encrypt transmission of cardholder data and sensitive information across public networks
5. Use and regularly update anti-virus software
6. Develop and maintain secure systems and applications
7. Restrict access to data by business need-to-know
8. Assign a unique ID to each person with computer access
9. Restrict physical access to cardholder data
10. Track and monitor all access to network resources and cardholder data
11. Regularly test security systems and processes
12. Maintain a policy that addresses information security

## Handling Credit Cards

- 1) If possible delegate handling to third party
- 2) Process transactions immediately and discard information as soon as possible
- 3) Do not store any information other than the authorization number returned upon processing. The latter must be logged.
- 4) Never display any portion of a Credit Card information, although PCI/CISP allow the display of the last 4 digits.

XXXX XXXY YYY YYYC (X=Bank ID, Y=Account ID, C=checksum)

(One could figure an account number in less than 100000 trials).

- 5) Never store the PVV (Pin Verification Value)

## **Handling Recurrent Credit Card Payments**

Example: subscription service.

This is the only case one may have to store Credit Card information.

- Get signed authorization from client
- Make sure you delete Credit Card information as soon as agreement expires
- Encrypt the Credit Card information

## **Handling Reversals**

Reversals: when funds are moved into a Credit Card Account (for example a refund or partial refund).

- This operation should always be authorized manually
- This operation should always be authorized by two different groups (to avoid internal fraud).

## **Handling chargebacks**

A special type of reversals, when the total charge amount is reversed.

- Same as reversal.

General rules:

- Validate Money (must be positive number)
- Never allow any money-handling function to accept negative values
- All chargebacks, reversals must be logged and require manual authorization
- There should be no online code/forms for chargebacks or reversals
- Consider placing restrictions based on CC Bank location
- For expensive goods, require phone authorization.

Credit Card Payment Glossary:

[http://www.echo-inc.com/credit\\_card\\_glossary.html](http://www.echo-inc.com/credit_card_glossary.html)

## **Topic: Definitions**

**ECT582**

**Secure  
Electronic  
Commerce**



## Data Validation

### Identity Management Systems

[http://en.wikipedia.org/wiki/Category:Identity\\_management\\_systems](http://en.wikipedia.org/wiki/Category:Identity_management_systems)

### Single Sign ON

[http://en.wikipedia.org/wiki/Single\\_sign-on](http://en.wikipedia.org/wiki/Single_sign-on)

### Lightweight Directory Access Protocol (LDAP)

[http://en.wikipedia.org/wiki/Lightweight\\_Directory\\_Access\\_Protocol](http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol)

### Kerberos

[http://en.wikipedia.org/wiki/Kerberos\\_%28protocol%29](http://en.wikipedia.org/wiki/Kerberos_%28protocol%29)

### Applications to web services:

<http://www-128.ibm.com/developerworks/webservices/library/ws-secure/>

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss#overview](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss#overview)

<http://en.wikipedia.org/wiki/SAML>

ECT582 - M. Di Pierro @ DePaul CTI

## Cryptography / Definitions

**Encryption:** the process of encoding data so that its meaning is not obvious

**Decryption:** the reverse process

**Cryptosystem:** a system (hardware or software) for encryption/decryption

**Plaintext:** data before encryption

**Ciphertext:** data after encryption

**Cryptology:** the study of encryption/decryption as a science

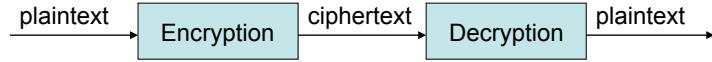
**Cryptographer:** tries to break encryption (legitimate)

**Cryptanalyst:** tries to break encryption (illegitimate)

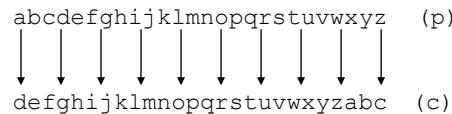


## Cryptography / Keyless Ciphers

## Keyless Ciphers



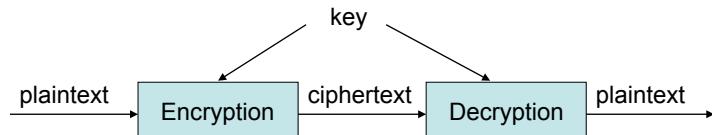
Example: Caesar Cipher (simplest substitution cipher)



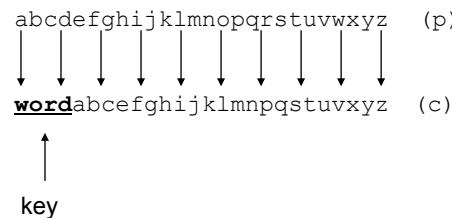
if p in {0,1,2,..., 25}, c = (p+3) % 26

## Cryptography / Symmetric-key Ciphers

## Symmetric-key Cryptosystem

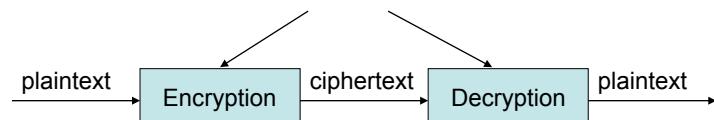


Example: Permutation Cipher (other simple substitution cipher)



## Cryptography / Random Sequence Ciphers

Random Sequence Ciphers      Random Sequence,  $r[i]$



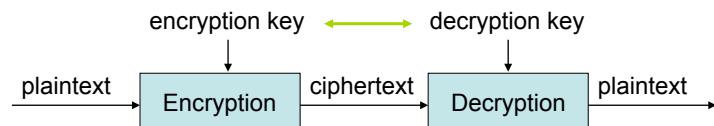
Example: Vernam Cipher

$$\text{if } p[i] \in \{0, 1, 2, \dots, 25\}, \quad c[i] = (p[i] + r[i]) \% 26 \\ p[i] = (c[i] - r[i] + 26 * n) \% 26$$

This cipher is perfect because for any ciphertext  $c$  and for each possible string  $p$  there is a key  $r$  that decode  $c$  into  $p$ .

## Cryptography / Public-key Ciphers

Public-key Ciphers



Definitions: encryption key is different from decryption key and one cannot be easily derived from the other, although they are related.

Example: RSA and Elliptic Curves Cryptosystems.  
Typically used for public key encryption.

## Other Symmetric-key ciphers

Hill Cipher:

```
for i in range(0,len(p),3):
    c[i+0]=(k[0][0] p[i+0] + k[0][1] p[i+1] + k[0][2] p[i+2]) % 26
    c[i+1]=(k[1][0] p[i+0] + k[1][1] p[i+1] + k[1][2] p[i+2]) % 26
    c[i+2]=(k[2][0] p[i+0] + k[2][1] p[i+1] + k[2][2] p[i+2]) % 26
encryption key is a matrix k, 3x3 of integer numbers. For decryption to exist one
must find a matrix k' such that  $k \times k' = 1$ .
```

Rotor machines

## Encryption Steps: Diffusion and Confusion

Diffusion: statistical properties of plaintext dissipated into long term statistics of ciphertext

$y[n]=(m[n]+m[n+1]+m[n+2]+\dots) \text{ mod } \dots$

Confusion: make the relationship between key value and statistics of ciphertext as complex as possible.

Typical operations

Binary shift:      01001101 -> 10011010                  out = (in << 1) + (in >> 7)

Binary xor:      01001101 + 11010100 = 10011001    out = in1 + in2

Permutation:      k0,k1,k2,k3,k4,k5,k6,k7=k4,k3,k2,k1,k7,k6,k5,k0

Encode large blocks:      ‘a’, 8bits; ‘ab’, 16 bits; ‘abcd’, 32bits; etc.

## Shannon Characteristics for Good Ciphers

- The amount of secrecy should determine the amount of work appropriate for encryption
- The set of keys and the algorithms should be free from complexity
- The implementation should be simple
- Errors in encryption/decryption should not propagate (?)
- The size of the encrypted text should not be larger than size of plaintext

## Encryption Algorithms: DES

### DES/3DES

The Data Encryption Standard (DES) was developed and endorsed by the U.S. government in 1977 as an official standard and forms the basis not only for the Automatic Teller Machines (ATM) PIN authentication but a variant is also utilized in UNIX password encryption. DES is a block cipher with 64-bit block size that uses 56-bit keys. Due to recent advances in computer technology, some experts no longer consider DES secure against all attacks; since then Triple-DES (3DES) has emerged as a stronger method. Using standard DES encryption, Triple-DES encrypts data three times and uses a different key for at least one of the three passes giving it a cumulative key size of 112-168 bits.

<http://www.aci.net/kalliste/des.htm>

## Encryption Algorithms: Blowfish, IDEA

### BLOWFISH

Blowfish is a symmetric block cipher just like DES or IDEA. It takes a variable-length key, from 32 to 448 bits, making it ideal for both domestic and exportable use. Bruce Schneier designed Blowfish in 1993 as a fast, free alternative to the then existing encryption algorithms. Since then Blowfish has been analyzed considerably, and is gaining acceptance as a strong encryption algorithm.

<http://www.schneier.com/blowfish.html>

### IDEA

International Data Encryption Algorithm (IDEA) is an algorithm that was developed by Dr. X. Lai and Prof. J. Massey in Switzerland in the early 1990s to replace the DES standard. It uses the same key for encryption and decryption, like DES operating on 8 bytes at a time. Unlike DES though it uses a 128 bit key. This key length makes it impossible to break by simply trying every key, and no other means of attack is known. It is a fast algorithm, and has also been implemented in hardware chipsets, making it even faster.

<http://vmsbox.cjb.net/idea.html>

## Encryption Algorithms: SEAL, RC4

### SEAL

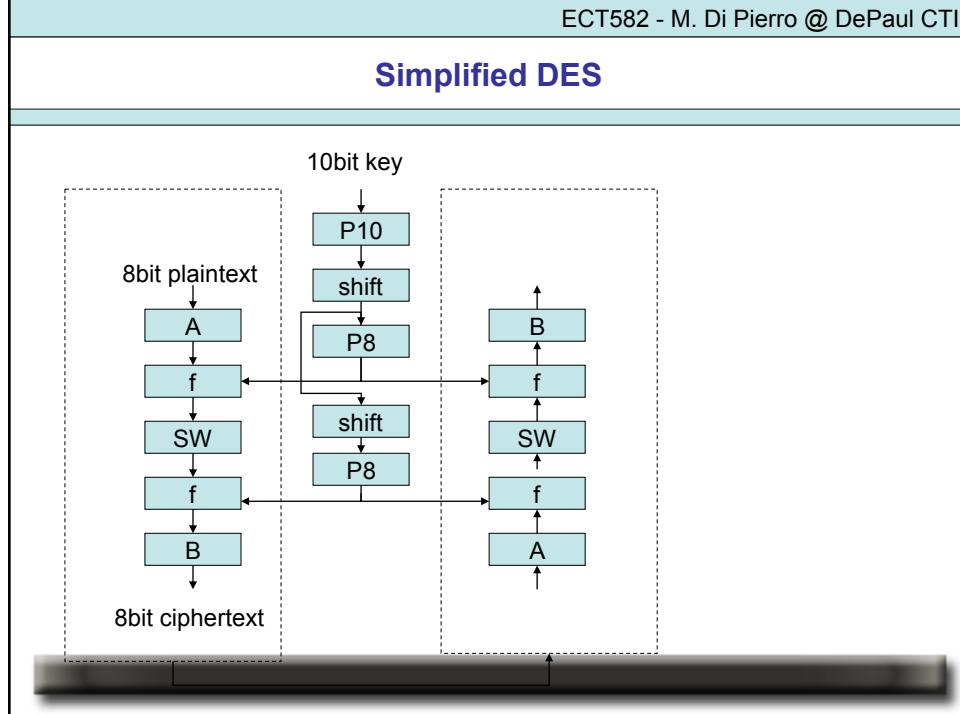
Rogaway and Coppersmith designed the Software-optimized Encryption Algorithm (SEAL) in 1993. It is a Stream-Cipher, i.e., data to be encrypted is continuously encrypted. Stream Ciphers are much faster than block ciphers (Blowfish, IDEA, DES) but have a longer initialization phase during which a large set of tables is done using the Secure Hash Algorithm. SEAL uses a 160 bit key for encryption and is considered very safe.

### RC4

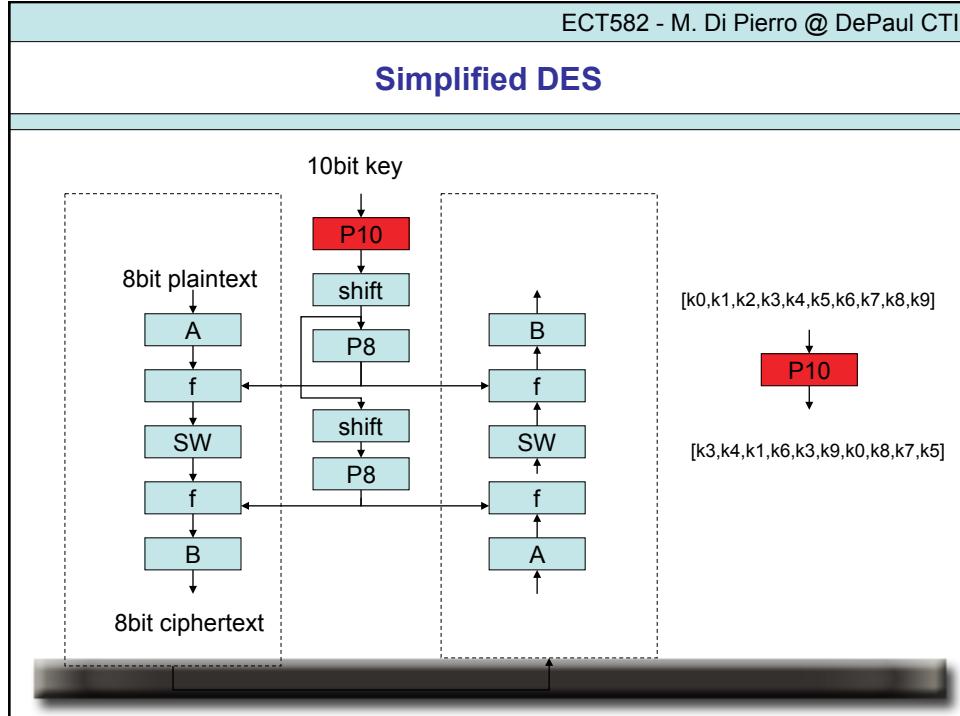
RC4 is a cipher invented by Ron Rivest, co-inventor of the RSA Scheme. It is used in a number of commercial systems like Lotus Notes and Netscape. It is a cipher with a key size of up to 2048 bits (256 bytes), which on the brief examination given it over the past year or so seems to be a relatively fast and strong cipher. It creates a stream of random bytes and XORing those bytes with the text. It is useful in situations in which a new key can be chosen for each message.

[http://www.fact-index.com/r/rc4\\_cipher.html](http://www.fact-index.com/r/rc4_cipher.html)

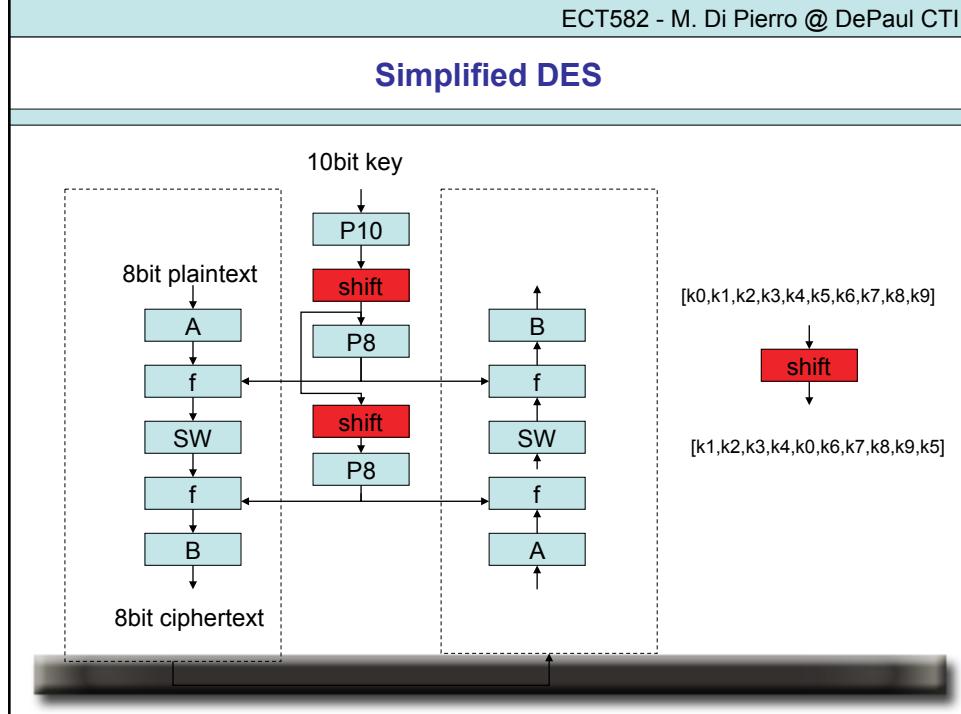
## Simplified DES



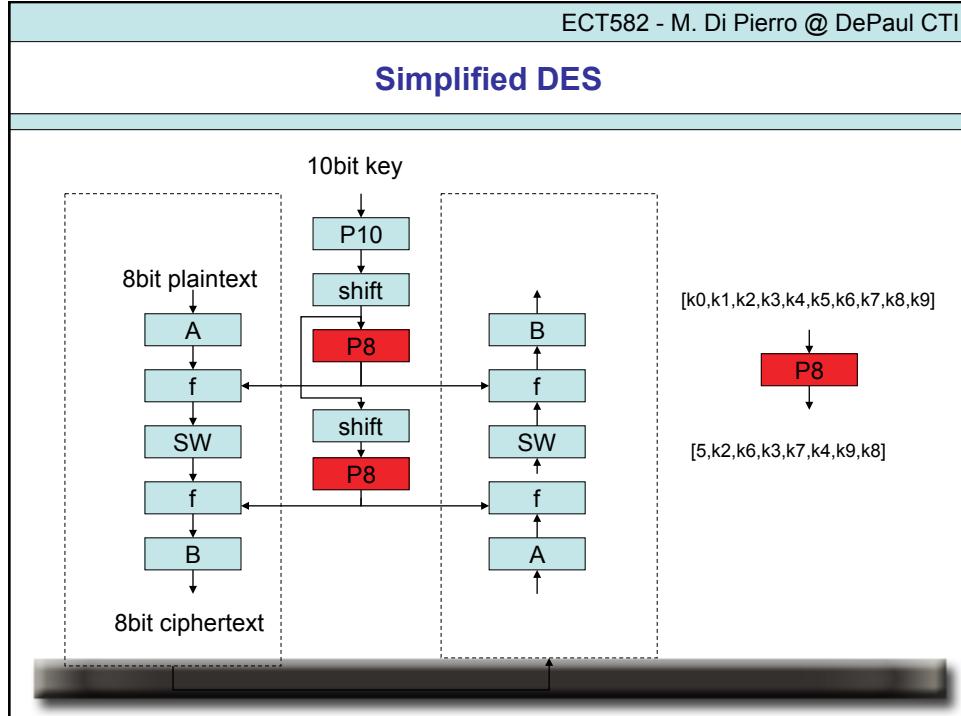
## Simplified DES



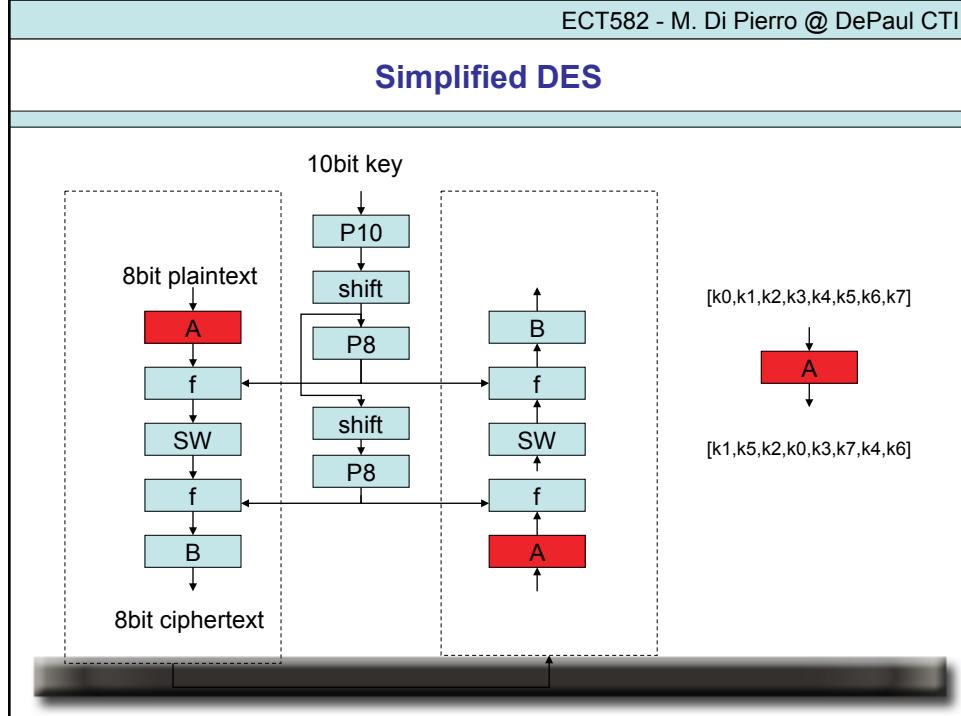
## Simplified DES



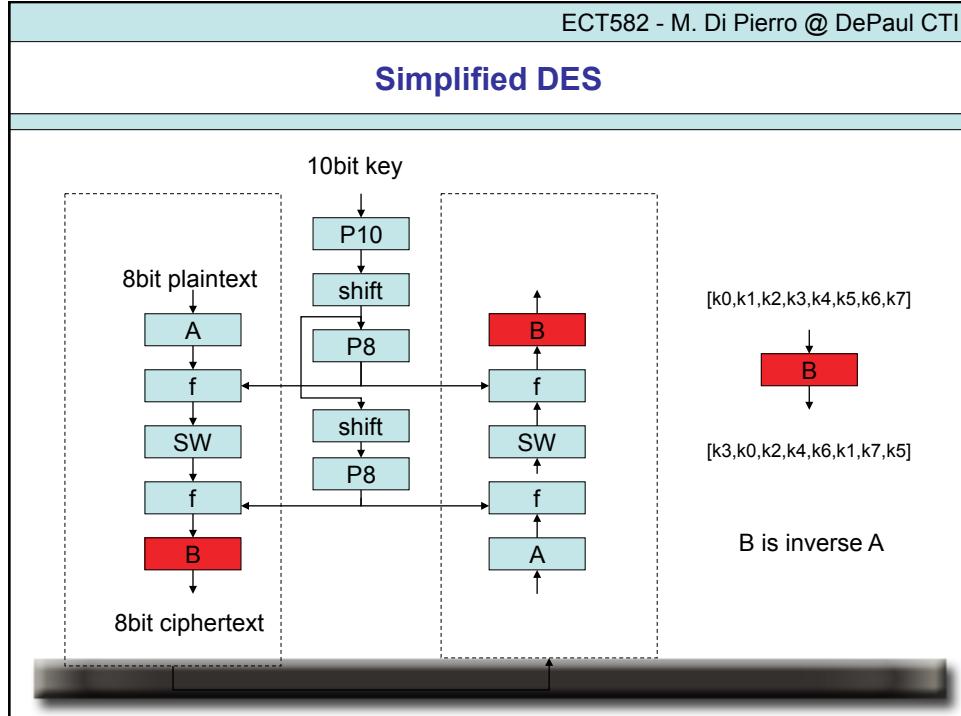
## Simplified DES



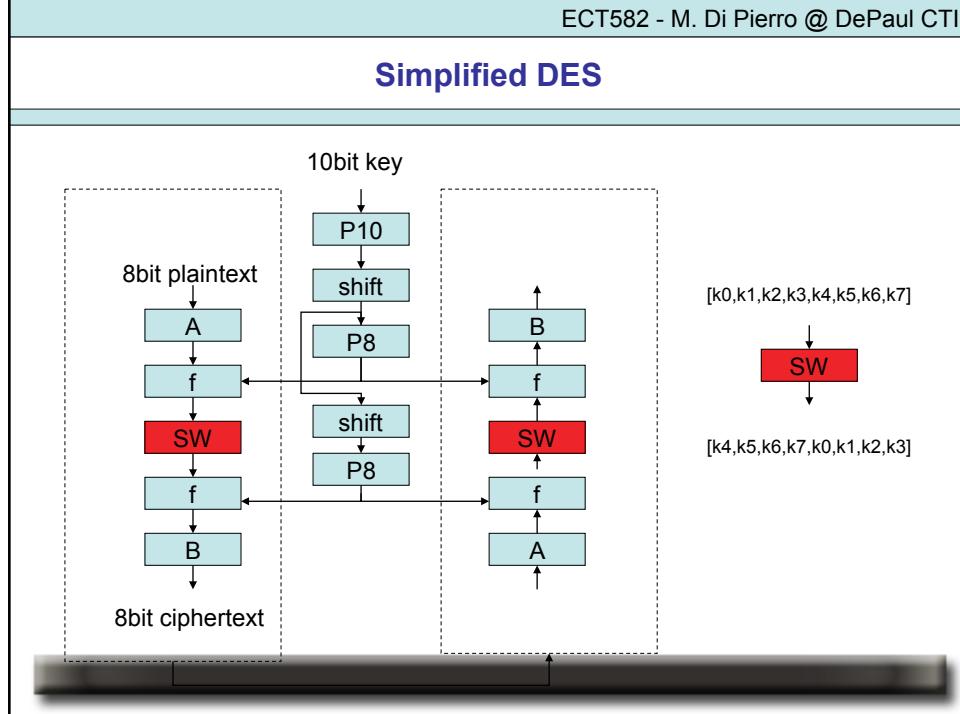
## Simplified DES



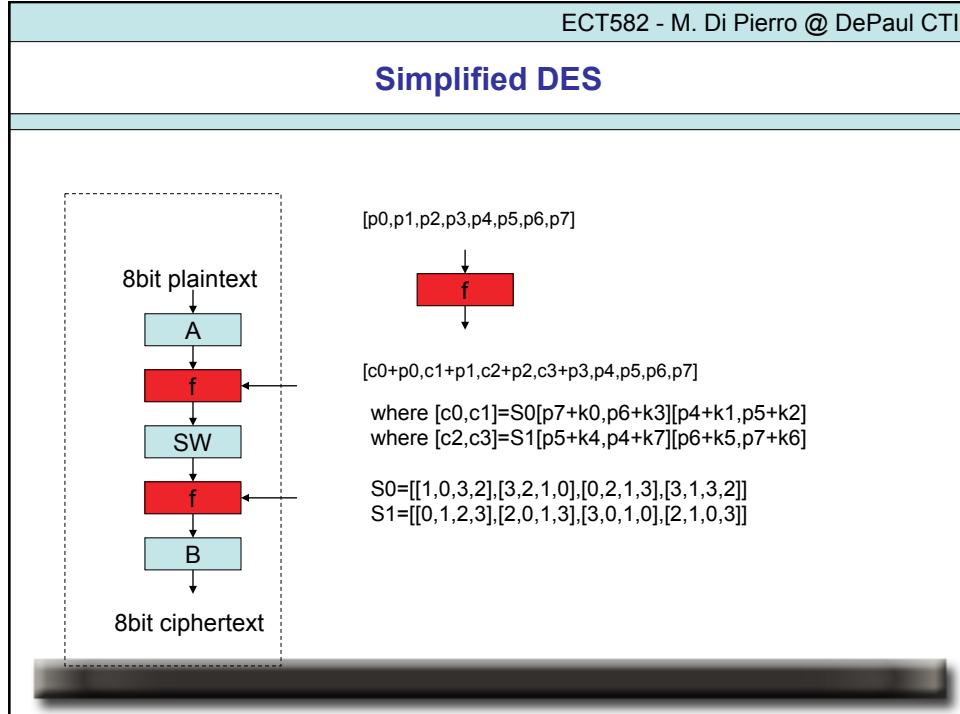
## Simplified DES



## Simplified DES



## Simplified DES



## Electronic Code Book Mode vs Cipher Block Chaining Mode

ECB:

$$c[i] = \text{encrypt}(p[i])$$

$$p[i] = \text{decrypt}(c[i])$$

CBC:

$$c[i] = \text{encrypt}(p[i] + c[i-1])$$

$$p[i] = \text{decrypt}(c[i]) + c[i-1]$$

## Encryption Attacks

Typical attacks

- Key search using a database
- Statistical analysis (count frequency of encrypted letters)
- Brute force attacks

For more secure encryption

- Use well tested algorithms
- compress data before encryption
- Use long key
- Use key that contains both uppercase, lowercase and numbers

## Legal Issues 1

The implications on society are staggering

Governments do not like citizens to keep secrets from them

In France, all nongovernmental cryptography is forbidden, and if used, a set of the secret keys must be handed up to the government

The US government has proposed an encryption scheme for all future US-installed digital phones that will include a special feature to allow the police to tap and decrypt telephone calls made within their shores

The US also has a law that prohibits exporting munitions, and encryption technology is classed as an “ammunition”

What's worse is that any encryption technology that can be exported must not have a key greater than 64 bits. <http://www.bxa.doc.gov/Encryption/>

## Legal Issues 2

Phil Zimmermann, who wrote an e-mail protection package called PGP (pretty good privacy) was accused of violating the US law

His software which uses 128 bit keys, was uploaded to the Internet by a friend, and made available globally

He was then accused of exporting ammunition

An Israeli researcher filed a US patent for a new digital signature technology - he then spent 6 months talking about it all over the world at conferences ...

The US patent office then instructed the Israeli to contact all Americans that knew of the research and tell them that disclosure of the research by them could result in a \$10,000 fine and/or possibly years in prison. I think it turned out well.

## Uses of encryption

### Link encryption:

Given a network, for each link of the network communication is encrypted.

Advantages:

- Source and destination of packets is encrypted

Disadvantages:

- Information can be stolen if routers are hacked
- Each couple of links have to share a secret key

### End-to-End Encryption

Given a network, messages are encrypted at the source and decrypted at the destination.

Advantages:

- Data is encrypted while in transit, routers do not have access to plaintext

Disadvantages:

- Source and destination of transmission is public
- Each couple of nodes in the network has to share a secret key

## Problem: Key distribution in End-to-End Encryption

Given a large network one needs a lot of keys shared among each couple of users and keys have to be occasionally replaced.

### Problem: how do we distribute the keys?

Without public key encryption:

- 1) Distribute all keys at once to each couple of users.
- 2) Use a single server and distribute one key to each user so that the user can securely communicate with server and get other keys.
- 3) Use Diffie-Hellman

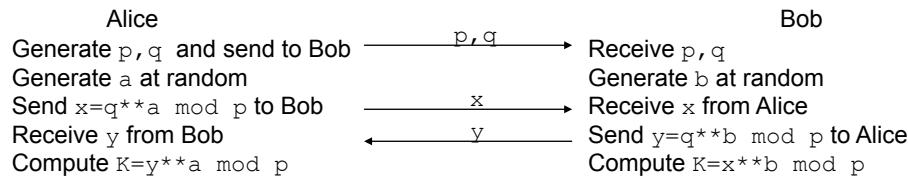
Better solutions is public key encryption. Later...

## Public Key Encryption: Diffie-Hellman

**Observation:**

- 1)  $q^{**}a \bmod p = (q \bmod p)^{**}a \bmod p$
- 2) If  $p$  is prime and  $q < p$  then, for any  $a$  and  $b$ ,  $(q^{**}a)^{**}b = (q^{**}b)^{**}a \bmod p$

**Key generation:**



Alice's  $K$  and Bob's  $K$  are the same.  $K$  is the Key.  
They have exchanged a key without ever transmitting it.

## Public Key Encryption: RSA

**Key Generation:**

Generate two large primes  $p, q$  (the difficult part!)  
 $n=p*q$   
 $\phi=(p-1)*(q-1)$   
choose  $e$  prime such that  $\gcd(e, \phi)=1$  usually  $e$  in  $[3, 17, 65537]$   
choose  $d$  such that  $e*d=1 \bmod \phi$  (use Extended Euclidean Algorithm)  
Public key =  $(n, e)$   
Private key =  $(n, d)$

**Encrypt:**

$$C=P^{**}e \bmod n$$

**Verify Signature on Data:**

$$(Data == Signature^{**}e \bmod n)$$

**Decrypt:**

$$P=C^{**}d \bmod n$$

**Sign Data:**

$$\text{Signature} = \text{Data}^{**}d \bmod n$$

## Public Key Encryption: RSA Example

### Key Generation:

Generate two large primes  $p=11$ ,  $q=3$  (the difficult part!)

$$n=p \cdot q = 33$$

$$\phi = (p-1) \cdot (q-1) = 20$$

choose  $e=3$  prime such that  $\gcd(e, \phi)=1$  usually  $e$  in  $[3, 17, 65537]$

choose  $d=7$  such that  $e \cdot d \equiv 1 \pmod{\phi}$  (use Extended Euclidean Algorithm)

$$\text{Public key} = (n, e) = (33, 3)$$

$$\text{Private key} = (n, d) = (33, 7)$$

### Encrypt:

$$P=7$$

$$C=7^{e=3} \pmod{33} = 13$$

### Decrypt:

$$C=13$$

$$P=C^{d=7} \pmod{33} = 7$$

### Verify Signature on Data:

$$\text{Data}=7; \text{Signature}=28$$

$$(7 \equiv 28^{e=3} \pmod{33})$$

### Sign Data:

$$\text{Data}=7$$

$$\text{Signature}=7^{d=7} \pmod{33} = 28$$

## Symmetric vs Public

Assurance	Prevents	Secret Key	Public Key
Confidentiality	Snooping	X	X
Authentication	Masquerading	X	X
Integrity	Message alteration w/o detection	X	X
Nonrepudiation	Sender's false denial		X

## Symmetric vs Public

Attribute	Secret Key	Public Key
Years in use	Thousands	Less than 50
Current standard	DES, 3DES, IDEA	RSA, Diffie-Hellman, DSA, elliptic curve?
Speed	Fast	Slow
Keys	Shared secret between at least two people	Private: kept concealed by one person Public: widely distributed
Non-repudiation	No Need trusted third party to act as witness	Yes Digital signatures: don't need trusted third party
Key length	56-bit obsolete 128-bit considered safe	1,024 suggested (RSA) ~172 (elliptic curve)

## Public-key Encryption Attacks

### Typical attacks

- Given the public key find the prime factors p and q such that  $n=p*q$ .  
This is technically possible but very slow.  
No polynomial algorithm for this factorization is known.

2) Reply Attack

3) Man-in-the-middle attack

### For more secure encryption

- Use a large public key (large primes p,q > 1024 bits)
- Timestamp data
- Use digital certificates and/or third party certification
- Use public key to encrypt short messages only such as another key.

## NOTATION

ID(a) = Identity of a  
 Ks = Session key  
 KUa = a public key  
 KRa = a private key  
 $E_{KUa}[P]$  = Plaintext encrypted using RSA and a public key  
 $D_{KRa}[C]$  = Ciphertext decrypted using RSA and a private key  
 [X,Y,Z] = a new message build concatenating the strings X, Y and Z

### Examples

$E_{KUb}[ID(a),KUa]$  reads:  
combine the identity of a with its public key and encrypt it with b public key.



## Send public keys

### Method 1 – Public Announcement of Public Keys

Attach the public key to your emails or post it on some web site so that it is available to everybody.

### Method 2 – Use a Public Directory

The directory maintains a database of ID,KU  
Registration is performed via encrypted communication  
Keys can be updated, works like telephone book

### Method 3 – Public Key Authority (\*)

Keys are not published but are stored by the Public Key Authority. Authorized users can retrieve them from the public key authority using the public key encryption to communicate with authority.

### Method 4 – Public Key Certificates (\*\*)

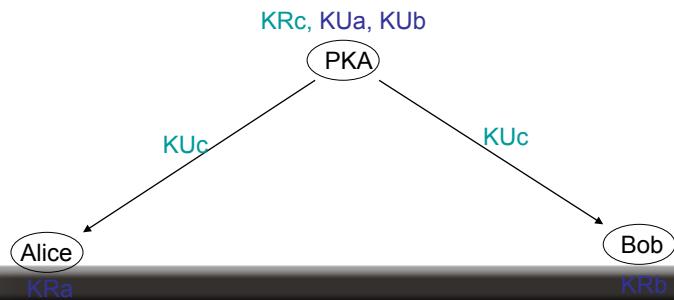
Parties authenticate each other by exchanging digital certificates provided by a trusted authority.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

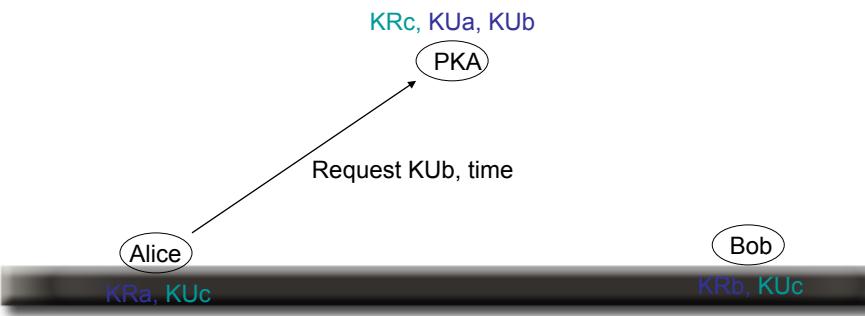
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

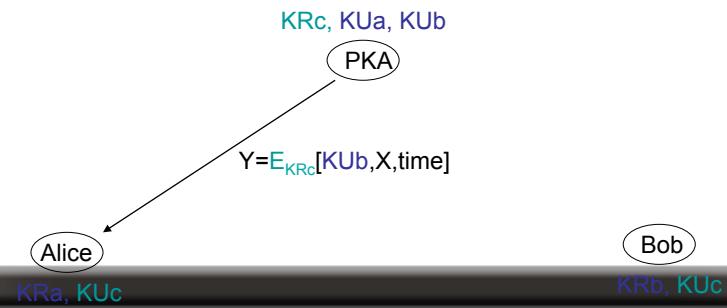
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

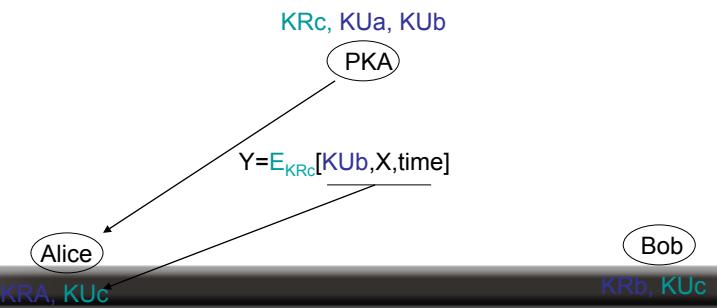
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

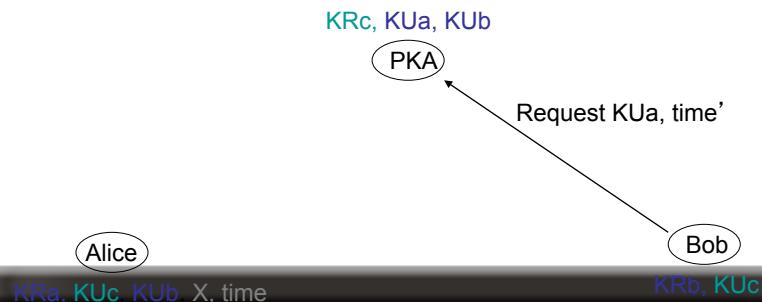
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

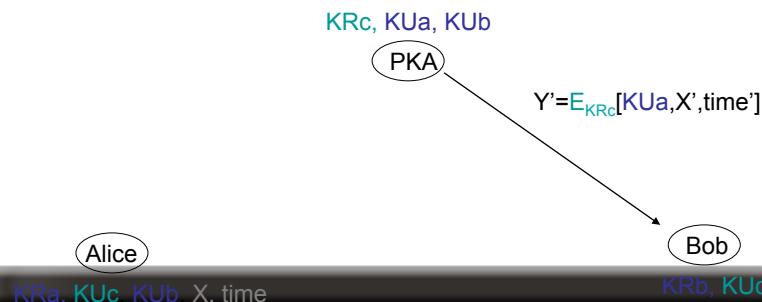
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

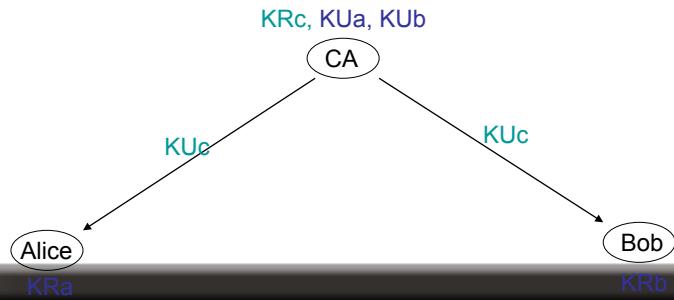
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

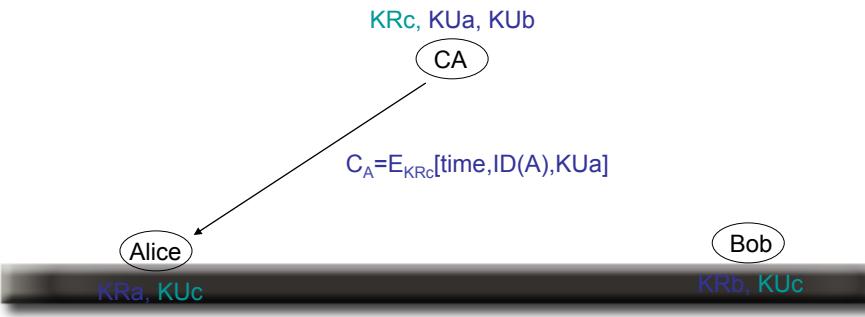
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

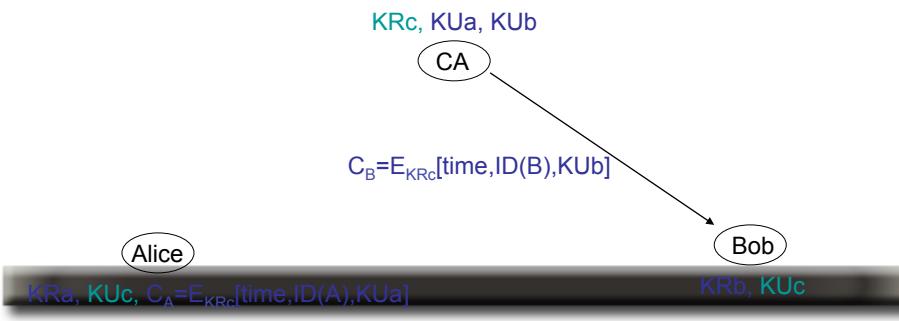
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.

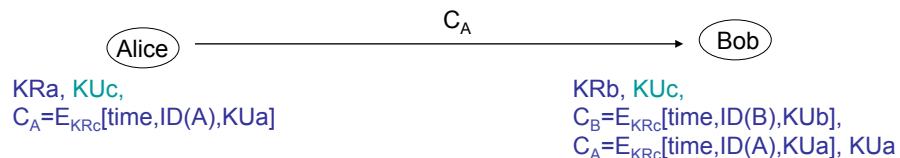


$KRa, KUc, C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$        $KRb, KUc, C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$

### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.



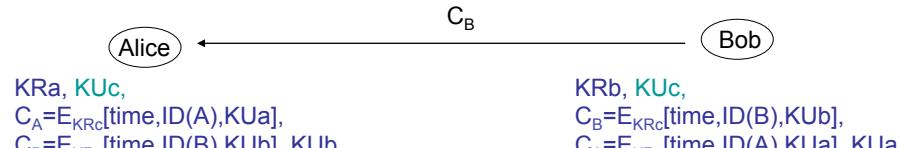
$KRa, KUc,$   
 $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$

$KRb, KUc,$   
 $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb],$   
 $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa], KUa$

### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

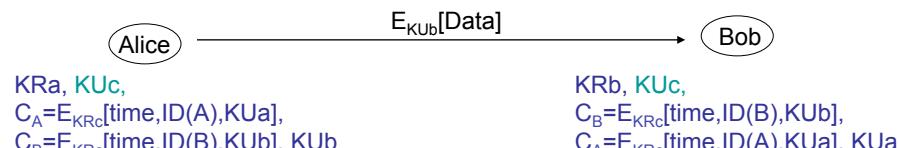
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

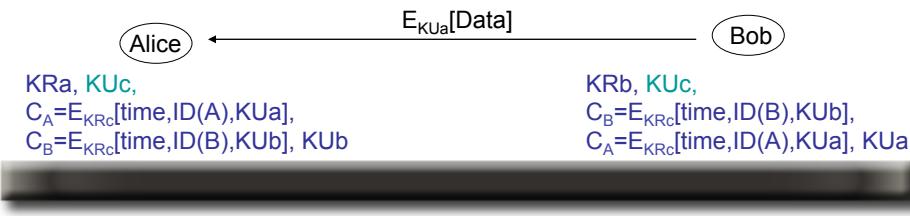
now A and B can communicate.



## Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.



## What is certificate?

A **certificate** is a string  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$  containing:

- the time the certificate was issued
- the ID of the certificate recipient
- and the public key of the certificate recipient

A certificate must be signed with the private key of the Certificate Authority

### Advantages:

Anybody can verify the certificate because the public key of the CA is public

Only CA can have written the certificate because nobody else has the CA private key

The Certificate certifies that the public key of the recipient is associate with its ID

As long one can trust the CA, the public key of the recipient must be authentic.

## Authentication and Authorization

**Authentication** is the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

**Authorization** is the process by which an entity attempts to confirm that another entity is allowed to access a resource.

The problem of authorization is often considered to be identical to that of authentication: however, there are many cases in which these two problems are different. For example, it is often desirable to grant access without requiring a unique identity. Familiar examples of authorization tokens include keys and tickets: they grant access without proving identity.

## Authentication / Hash Functions

A **hash function** is a function that converts an input from a (typically) large domain into an output in a (typically) smaller range (the *hash value*, often a subset of the integers).

The desired characteristics of a hash function in general are that  $H(x) \neq H(y)$  implies  $x \neq y$  and that  $H(x) = H(y)$  probably implies that  $x = y$ . i.e. it would be possible to uniquely identify most of these strings using this hash.

The output of a hash function is called the “digest” of the function.

A hash function is **cryptographically secure** if the digest is "random" to prospective **attackers**. i.e. it does not leak any information about the message itself, and it is computationally impractical to produce another other messages that produces the same digest. Any change to the message, even a single bit, should result in a dramatically different message digest when re-generated from the received message. A cryptographic hash function is considered secure if it is not computationally feasible to determine the content of a message from its message digest, nor to find "collisions", wherein two different messages have the same message digest.

## Authentication / Hash Functions

Checksums and Cyclic redundancy checks (CRCs) are quite distinct from cryptographic hash functions, and are used for different applications.

**MD5** is one of a series of message digest algorithms designed by Professor Ronald Rivest of MIT (Rivest, 1994). It has a 128-bit hash value. When analytic work indicated that MD5's predecessor — MD4 — was likely to be insecure, MD5 was designed in 1991 to be a secure replacement; weaknesses were indeed subsequently found in MD4 by Hans Dobbertin. Significant security flaws in MD5 were reported in 2004, and many systems that use it will probably switch to more secure alternatives.

The **SHA (Secure Hash Algorithm)** family is a set of related cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST). The first member of the family, published in 1993, is officially called SHA. However, today, it is often unofficially called **SHA-0** to avoid confusion with its successors. Two years later, **SHA-1**, the first successor to SHA, was published. Four more variants, have since been published with increased output ranges and a slightly different design: **SHA-224**, **SHA-256**, **SHA-384**, and **SHA-512** (all are sometimes referred to as **SHA-2**).

## Authentication / Hash Functions

Example, **logical xor** (cryptographically insecure):

- Break input in blocks of fixed size
- Perform padding of last block
- Compute logical xor of all blocks
- The result is the digest

- Example, **DES based hashing** (cryptographically secure):
- Break input in blocks of fixed size  $B[i]$
- Store a fixed random sequence in  $S[0]$
- Encrypt  $S[0]$  with the DES using  $B[0]$  as key, store ciphertext in  $S[1]$
- Encrypt  $S[1]$  with the DES using  $B[1]$  as key, store ciphertext in  $S[2]$
- ...
- Encrypt  $S[n-1]$  with the DES using  $B[n-1]$  as key, store ciphertext in  $S[n]$
- $S[n]$  is the digest

## Authentication. How?

Other common examples of access control involving authentication include:

- withdrawing cash from an [ATM](#).
- controlling a remote computer over the [Internet](#).
- using an Internet banking system.

The methods by which a human can authenticate themselves are generally classified into three cases:

- Something the user **is** (fingerprint, retinal pattern, DNA sequence, voice)
- Something the user **has** (ID card)
- Something the user **knows** (password, a pass phrase, a PIN)

**digital signatures** are a sequence of bits that permit authentication of digital information. They are often treated, sometimes too closely, as analogous to a physical signature on paper.

A **challenge-response test** is a test involving a set of questions (or "challenges"), that a system has to answer in order to identify itself. A very simple example is asking for a password.



## Digital Signature

User A wants to sign a sequence of bits X. He performs the following operations:

1. He computes the digest  $d = \text{hash}(X)$  using a cryptographically secure hash func.
2. He generates a public/private key pair  $K_{Ua}/K_{Ra}$
3. He encrypts the digest using the private key,  $c = E_{K_{Ra}}[d]$
4. The string c is the digital signature for X by A.

Anybody who has  $K_{Ua}$  can verify the signature by computing

$$D_{K_{Ua}}[c] == \text{hash}(X)$$

Only A can have created the digital signature because only A has  $K_{Ra}$ .



## Secure EMail

Given some secure Email text X:

1. Create a session key K
2. Encrypt X with session key K (for example using DES),  $Y=E_K[X]$
3. Encrypt K using public key of email recipient,  $c=E_{Kub}[K]$
4. Combine c with Y, encrypted message=[c, Y]
5. Recipient has KR<sub>a</sub> and decrypt K and use K to decrypt text X.

For additional protection the text itself should contain a hash value.

## Topic: Definitions

# ECT582

Secure  
Electronic  
Commerce



## Authentication and Authorization

**Authentication** is the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

**Authorization** is the process by which an entity attempts to confirm that another entity is allowed to access a resource.

The problem of authorization is often considered to be identical to that of authentication: however, there are many cases in which these two problems are different. For example, it is often desirable to grant access without requiring a unique identity. Familiar examples of authorization tokens include keys and tickets: they grant access without proving identity.

## Hash Functions

A **hash function** is a function that converts an input from a (typically) large domain into an output in a (typically) smaller range (the *hash value*, often a subset of the integers).

The desired characteristics of a hash function in general are that  $H(x) \neq H(y)$  implies  $x \neq y$  and that  $H(x) = H(y)$  probably implies that  $x = y$ . i.e. it would be possible to uniquely identify most of these strings using this hash.

The output of a hash function is called the “digest” of the function.

A hash function is **cryptographically secure** if the digest is "random" to prospective **attackers**. i.e. it does not leak any information about the message itself, and it is computationally impractical to produce another other messages that produces the same digest. Any change to the message, even a single bit, should result in a dramatically different message digest when re-generated from the received message. A cryptographic hash function is considered secure if it is not computationally feasible to determine the content of a message from its message digest, nor to find "collisions", wherein two different messages have the same message digest.

## Hash Functions

Checksums and Cyclic redundancy checks (CRCs) are quite distinct from cryptographic hash functions, and are used for different applications.

**MD5** is one of a series of message digest algorithms designed by Professor Ronald Rivest of MIT (Rivest, 1994). It has a 128-bit hash value. When analytic work indicated that MD5's predecessor — MD4 — was likely to be insecure, MD5 was designed in 1991 to be a secure replacement; weaknesses were indeed subsequently found in MD4 by Hans Dobbertin. Significant security flaws in MD5 were reported in 2004, and many systems that use it will probably switch to more secure alternatives.

The **SHA (Secure Hash Algorithm)** family is a set of related cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST). The first member of the family, published in 1993, is officially called SHA. However, today, it is often unofficially called **SHA-0** to avoid confusion with its successors. Two years later, **SHA-1**, the first successor to SHA, was published. Four more variants, have since been published with increased output ranges and a slightly different design: **SHA-224**, **SHA-256**, **SHA-384**, and **SHA-512** (all are sometimes referred to as **SHA-2**).

## Hash Functions

Example, **logical xor** (cryptographically insecure):

- Break input in blocks of fixed size
- Perform padding of last block
- Compute logical xor of all blocks
- The result is the digest

- Example, **DES based hashing** (cryptographically secure):
- Break input in blocks of fixed size  $B[i]$
- Store a fixed random sequence in  $S[0]$
- Encrypt  $S[0]$  with the DES using  $B[0]$  as key, store ciphertext in  $S[1]$
- Encrypt  $S[1]$  with the DES using  $B[1]$  as key, store ciphertext in  $S[2]$
- ...
- Encrypt  $S[n-1]$  with the DES using  $B[n-1]$  as key, store ciphertext in  $S[n]$
- $S[n]$  is the digest

## Data Validation

### Identity Management Systems

[http://en.wikipedia.org/wiki/Category:Identity\\_management\\_systems](http://en.wikipedia.org/wiki/Category:Identity_management_systems)

### Single Sign ON

[http://en.wikipedia.org/wiki/Single\\_sign-on](http://en.wikipedia.org/wiki/Single_sign-on)

### Lightweight Directory Access Protocol (LDAP)

[http://en.wikipedia.org/wiki/Lightweight\\_Directory\\_Access\\_Protocol](http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol)

### Kerberos

[http://en.wikipedia.org/wiki/Kerberos\\_%28protocol%29](http://en.wikipedia.org/wiki/Kerberos_%28protocol%29)

### Applications to web services:

<http://www-128.ibm.com/developerworks/webservices/library/ws-secure/>

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss#overview](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss#overview)

<http://en.wikipedia.org/wiki/SAML>

ECT582 - M. Di Pierro @ DePaul CTI

## Uses of encryption

### Link encryption:

Given a network, for each link of the network communication is encrypted.

Advantages:

- Source and destination of packets is encrypted

Disadvantages:

- Information can be stolen if routers are hacked
- Each couple of links have to share a secret key

### End-to-End Encryption

Given a network, messages are encrypted at the source and decrypted at the destination.

Advantages:

- Data is encrypted while in transit, routers do not have access to plaintext

Disadvantages:

- Source and destination of transmission is public
- Each couple of nodes in the network has to share a secret key

## Cryptography / Definitions

**Encryption:** the process of encoding data so that its meaning is not obvious

**Decryption:** the reverse process

**Cryptosystem:** a system (hardware or software) for encryption/decryption

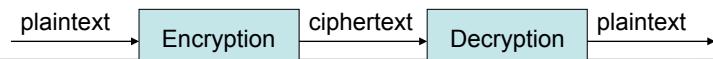
**Plaintext:** data before encryption

**Ciphertext:** data after encryption

**Cryptology:** the study of encryption/decryption as a science

**Cryptographer:** tries to break encryption (legitimate)

**Cryptanalyst:** tries to break encryption (illegitimate)

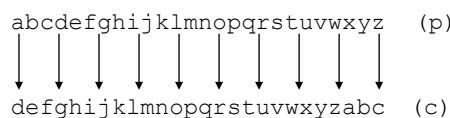


## Cryptography / Keyless Ciphers

**Keyless Ciphers**



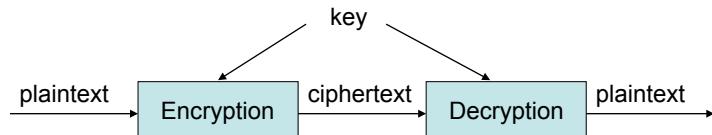
Example: Caesar Cipher (simplest substitution cipher)



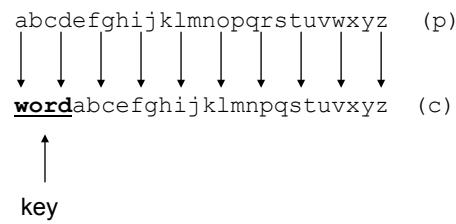
if  $p \in \{0, 1, 2, \dots, 25\}$ ,  $c = (p+3) \% 26$

## Cryptography / Symmetric-key Ciphers

Symmetric-key Cryptosystem



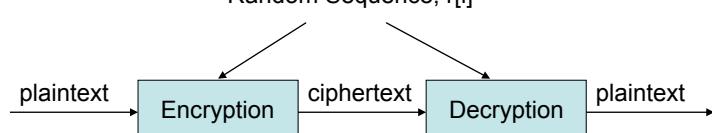
Example: Permutation Cipher (other simple substitution cipher)



## Cryptography / Random Sequence Ciphers

Random Sequence Ciphers

Random Sequence,  $r[i]$



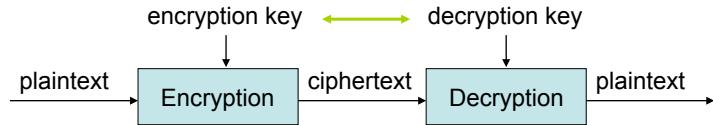
Example: Vernam Cipher

$$\begin{aligned} \text{if } p[i] \in \{0, 1, 2, \dots, 25\}, \quad & c[i] = (p[i] + r[i]) \% 26 \\ & p[i] = (c[i] - r[i] + 26 * n) \% 26 \end{aligned}$$

This cipher is perfect because for any ciphertext  $c$  and for each possible string  $p$  there is a key  $r$  that decode  $c$  into  $p$ .

## Cryptography / Public-key Ciphers

### Public-key Ciphers



Definitions: encryption key is different from decryption key and one cannot be easily derived from the other, although they are related.

Example: RSA and Elliptic Curves Cryptosystems.  
Typically used for public key encryption.

## Other Symmetric-key ciphers

### Hill Cipher:

```

for i in range(0,len(p),3):
    c[i+0]=(k[0][0] p[i+0] + k[0][1] p[i+1] + k[0][2] p[i+2]) % 26
    c[i+1]=(k[1][0] p[i+0] + k[1][1] p[i+1] + k[1][2] p[i+2]) % 26
    c[i+2]=(k[2][0] p[i+0] + k[2][1] p[i+1] + k[2][2] p[i+2]) % 26
  
```

encryption key is a matrix k, 3x3 of integer numbers. For decryption to exist one must find a matrix k' such that  $k \times k' = 1$ .

Rotor machines

## Encryption Steps: Diffusion and Confusion

**Diffusion:** statistical properties of plaintext dissipated into long term statistics of ciphertext

$$y[n] = (m[n] + m[n+1] + m[n+2] + \dots) \bmod \dots$$

**Confusion:** make the relationship between key value and statistics of ciphertext as complex as possible.

Typical operations

Binary shift:      01001101 -> 10011010      out = (in << 1) + (in >> 7)

Binary xor:      01001101 + 11010100 = 10011001      out = in1 + in2

Permutation:      k0,k1,k2,k3,k4,k5,k6,k7=k4,k3,k2,k1,k7,k6,k5,k0

Encode large blocks:      ‘a’, 8bits; ‘ab’, 16 bits; ‘abcd’, 32bits; etc.

## Shannon Characteristics for Good Ciphers

The amount of secrecy should determine the amount of work appropriate for encryption

The set of keys and the algorithms should be free from complexity

The implementation should be simple

Errors in encryption/decryption should not propagate (?)

The size of the encrypted text should not be larger than size of plaintext

## Encryption Algorithms: DES

### DES/3DES

The Data Encryption Standard (DES) was developed and endorsed by the U.S. government in 1977 as an official standard and forms the basis not only for the Automatic Teller Machines (ATM) PIN authentication but a variant is also utilized in UNIX password encryption. DES is a block cipher with 64-bit block size that uses 56-bit keys. Due to recent advances in computer technology, some experts no longer consider DES secure against all attacks; since then Triple-DES (3DES) has emerged as a stronger method. Using standard DES encryption, Triple-DES encrypts data three times and uses a different key for at least one of the three passes giving it a cumulative key size of 112-168 bits.

<http://www.aci.net/kalliste/des.htm>

## Encryption Algorithms: Blowfish, IDEA

### BLOWFISH

Blowfish is a symmetric block cipher just like DES or IDEA. It takes a variable-length key, from 32 to 448 bits, making it ideal for both domestic and exportable use. Bruce Schneier designed Blowfish in 1993 as a fast, free alternative to the then existing encryption algorithms. Since then Blowfish has been analyzed considerably, and is gaining acceptance as a strong encryption algorithm.

<http://www.schneier.com/blowfish.html>

### IDEA

International Data Encryption Algorithm (IDEA) is an algorithm that was developed by Dr. X. Lai and Prof. J. Massey in Switzerland in the early 1990s to replace the DES standard. It uses the same key for encryption and decryption, like DES operating on 8 bytes at a time. Unlike DES though it uses a 128 bit key. This key length makes it impossible to break by simply trying every key, and no other means of attack is known. It is a fast algorithm, and has also been implemented in hardware chipsets, making it even faster.

<http://vmsbox.cjb.net/idea.html>

## Encryption Algorithms: SEAL, RC4

### **SEAL**

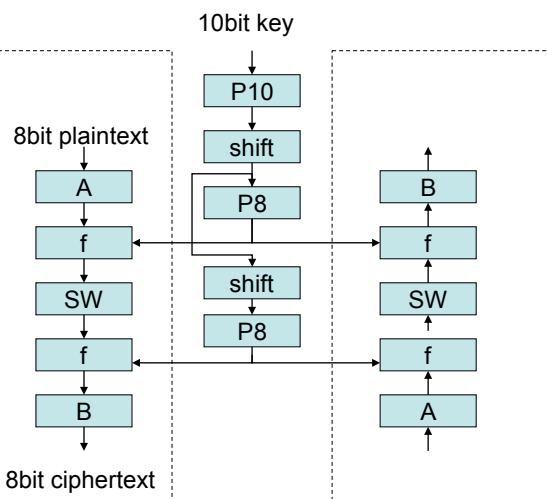
Rogaway and Coppersmith designed the Software-optimized Encryption Algorithm (SEAL) in 1993. It is a Stream-Cipher, i.e., data to be encrypted is continuously encrypted. Stream Ciphers are much faster than block ciphers (Blowfish, IDEA, DES) but have a longer initialization phase during which a large set of tables is done using the Secure Hash Algorithm. SEAL uses a 160 bit key for encryption and is considered very safe.

### **RC4**

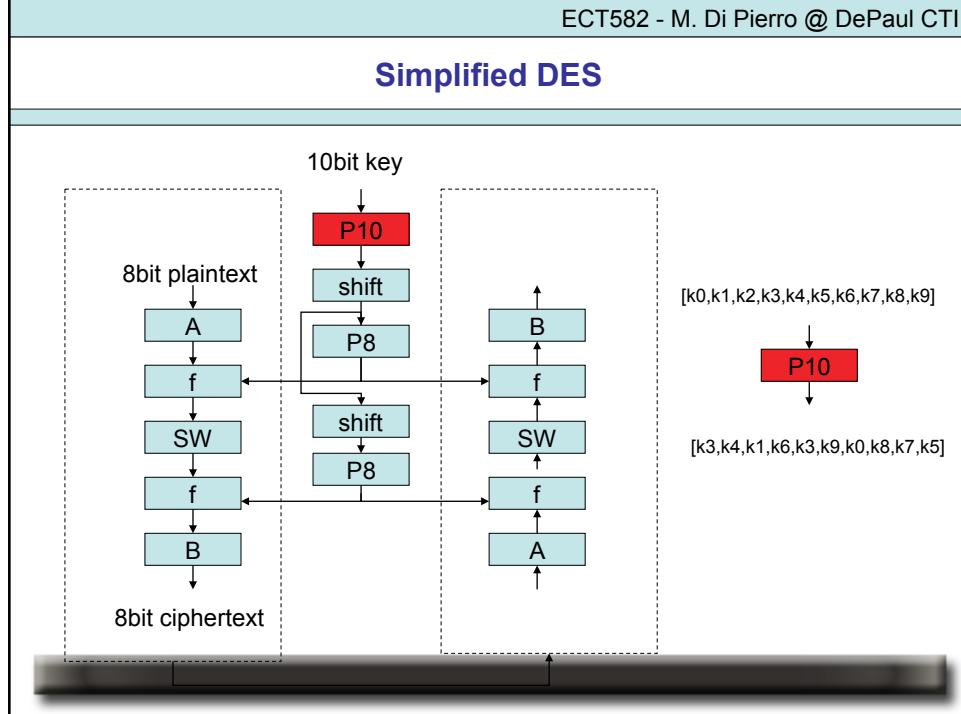
RC4 is a cipher invented by Ron Rivest, co-inventor of the RSA Scheme. It is used in a number of commercial systems like Lotus Notes and Netscape. It is a cipher with a key size of up to 2048 bits (256 bytes), which on the brief examination given it over the past year or so seems to be a relatively fast and strong cipher. It creates a stream of random bytes and XORing those bytes with the text. It is useful in situations in which a new key can be chosen for each message.

[http://www.fact-index.com/r/rc/rc4\\_cipher.html](http://www.fact-index.com/r/rc/rc4_cipher.html)

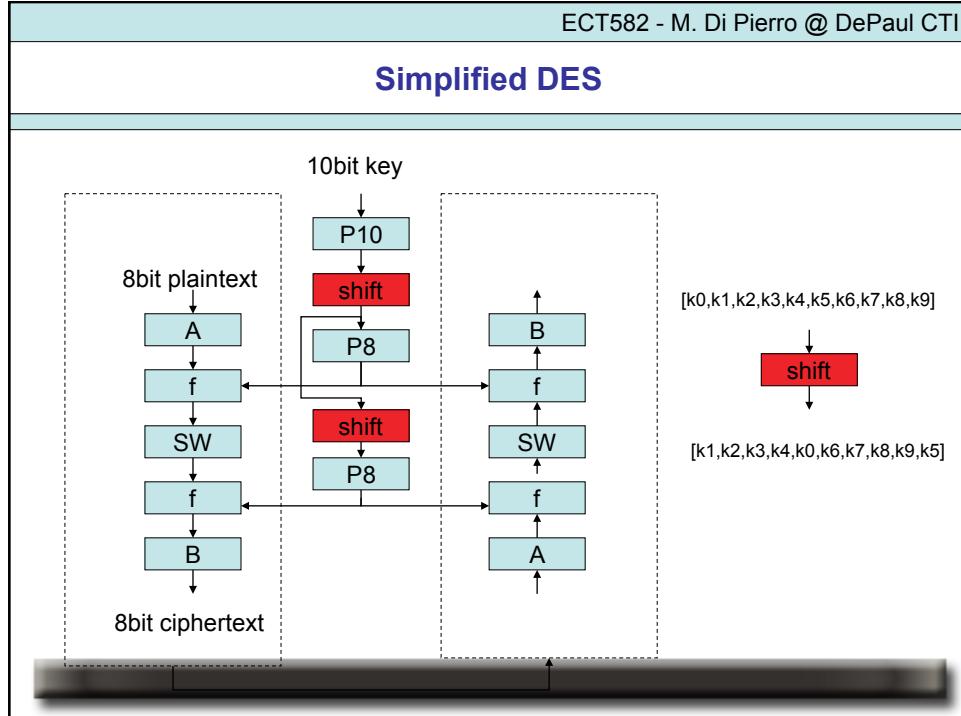
## Simplified DES

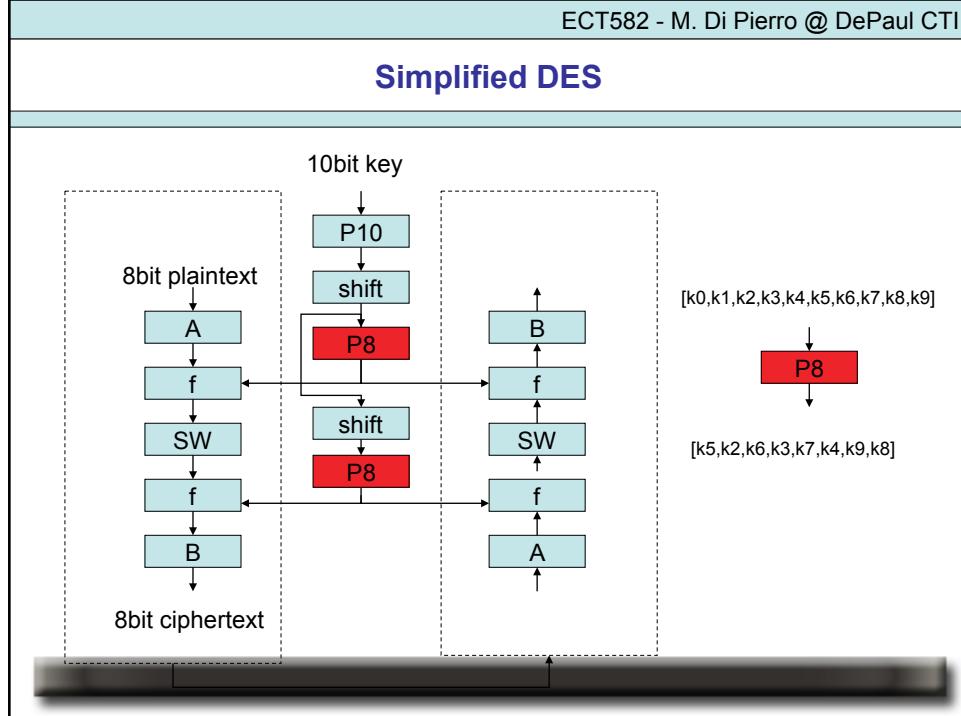
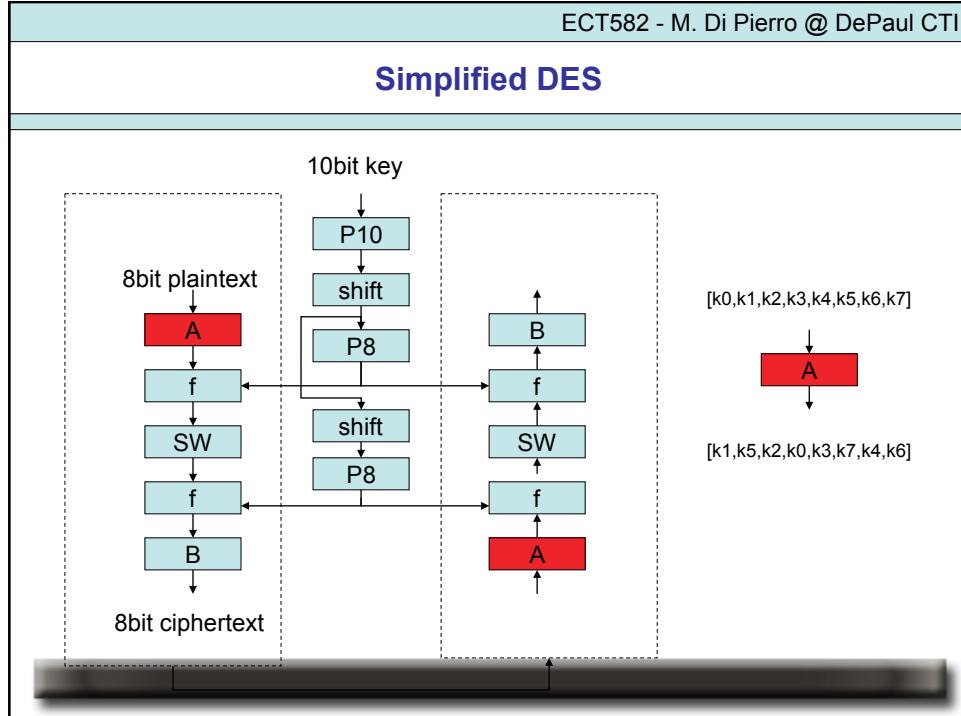


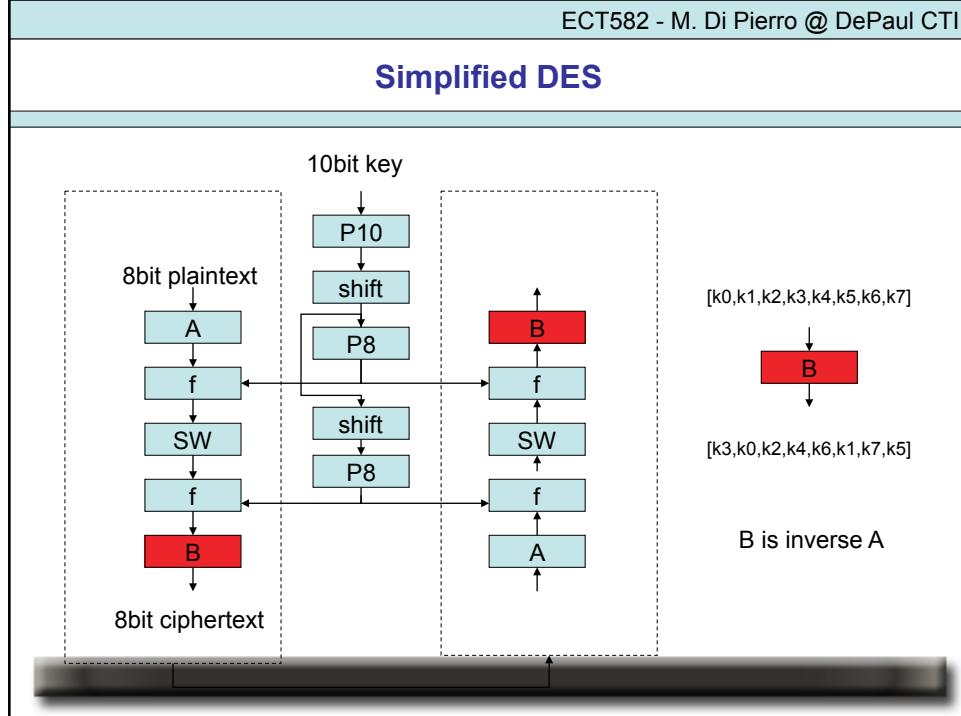
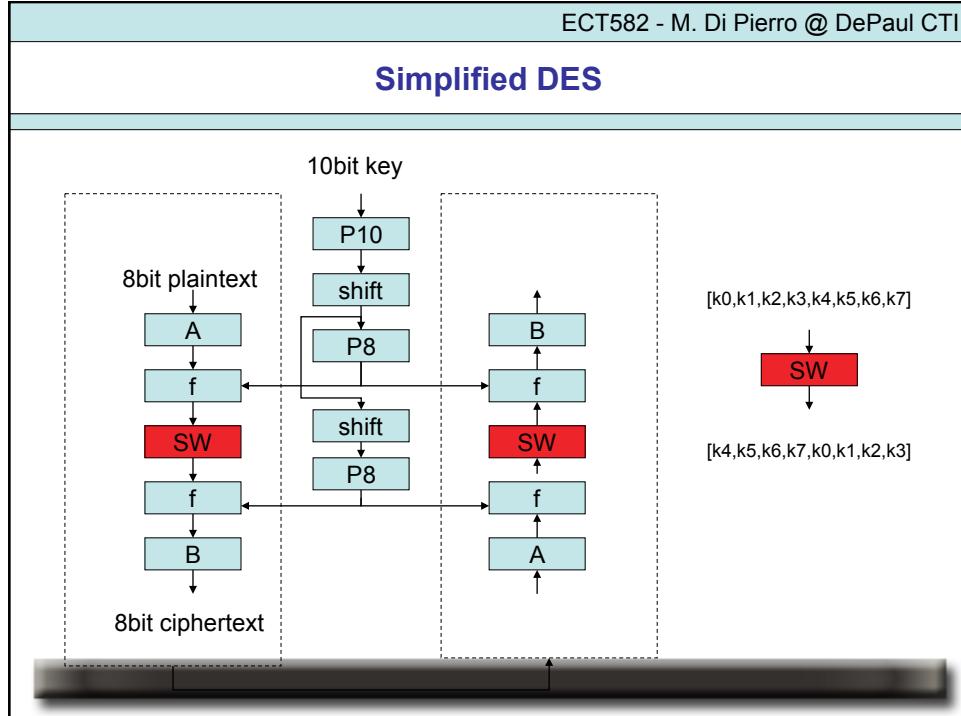
## Simplified DES



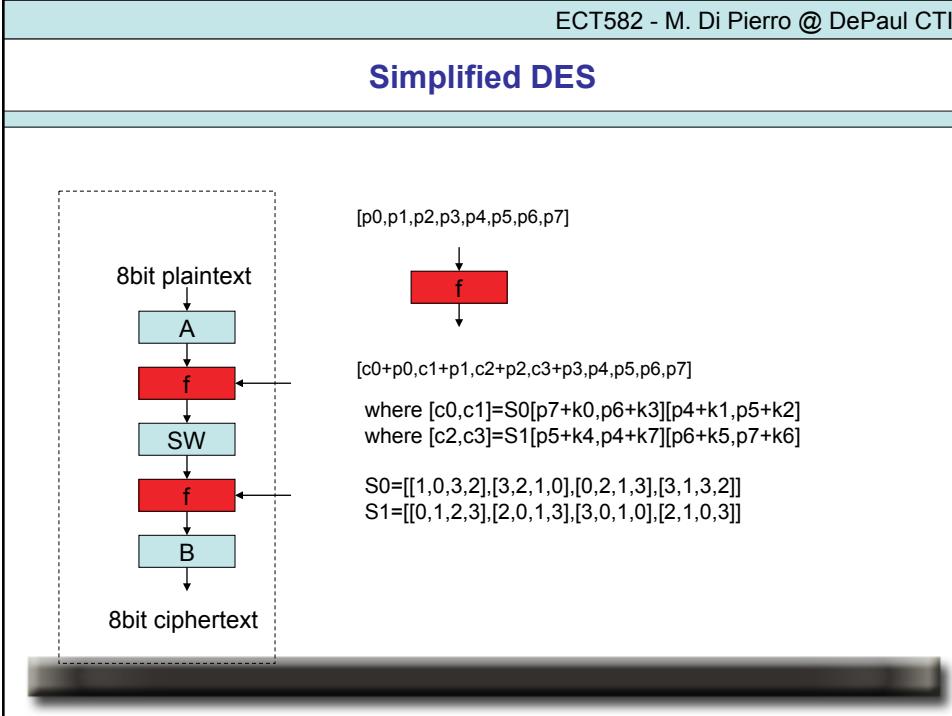
## Simplified DES



**Simplified DES****Simplified DES**

**Simplified DES****Simplified DES**

## Simplified DES



## Electronic Code Book Mode vs Cipher Block Chaining Mode

ECB:

$$c[i] = \text{encrypt}(p[i])$$

$$p[i] = \text{decrypt}(c[i])$$

CBC:

$$c[i] = \text{encrypt}(p[i] + c[i-1])$$

$$p[i] = \text{decrypt}(c[i]) + c[i-1]$$

## Encryption Attacks

### Typical attacks

- Key search using a database
- Statistical analysis (count frequency of encrypted letters)
- Brute force attacks

### For more secure encryption

- Use well tested algorithms
- compress data before encryption
- Use long key
- Use key that contains both uppercase, lowercase and numbers

## Problem: Key distribution in End-to-End Encryption

Given a large network one needs a lot of keys shared among each couple of users and keys have to be occasionally replaced.

Problem: how do we distribute the keys?

Without public key encryption:

- 1) Distribute all keys at once to each couple of users.
- 2) Use a single server and distribute one key to each user so that the user can securely communicate with server and get other keys.
- 3) Use Diffie-Hellman

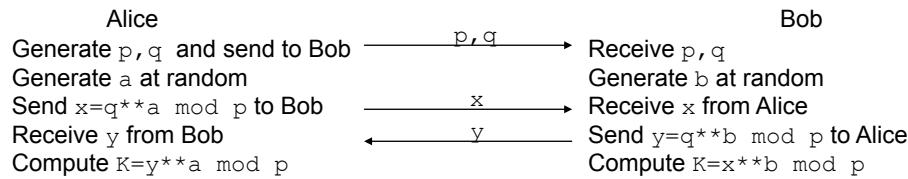
Better solutions is public key encryption. Later...

## Public Key Encryption: Diffie-Hellman

Observation:

- 1)  $q^{**}a \bmod p = (q \bmod p)^{**}a \bmod p$
- 2) If  $p$  is prime and  $q < p$  then, for any  $a$  and  $b$ ,  $(q^{**}a)^{**}b = (q^{**}b)^{**}a \bmod p$

Key generation:



Alice's  $K$  and Bob's  $K$  are the same.  $K$  is the Key.  
They have exchanged a key without ever transmitting it.

## Public Key Encryption: RSA

Key Generation:

Generate two large primes  $p, q$  (the difficult part!)  
 $n=p*q$   
 $\phi=(p-1)*(q-1)$   
choose  $e$  prime such that  $\gcd(e, \phi)=1$  usually  $e$  in  $[3, 17, 65537]$   
choose  $d$  such that  $e*d=1 \bmod \phi$  (use Extended Euclidean Algorithm)  
Public key =  $(n, e)$   
Private key =  $(n, d)$

Encrypt:

$$C=P^{**}e \bmod n$$

Verify Signature on Data:

$$(Data == Signature^{**}e \bmod n)$$

Decrypt:

$$P=C^{**}d \bmod n$$

Sign Data:

$$\text{Signature} = \text{Data}^{**}d \bmod n$$

## Public Key Encryption: RSA Example

### Key Generation:

Generate two large primes  $p=11$ ,  $q=3$  (the difficult part!)

$$n=p \cdot q = 33$$

$$\phi = (p-1) \cdot (q-1) = 20$$

choose  $e=3$  prime such that  $\gcd(e, \phi)=1$  usually  $e$  in  $[3, 17, 65537]$

choose  $d=7$  such that  $e \cdot d \equiv 1 \pmod{\phi}$  (use Extended Euclidean Algorithm)

$$\text{Public key} = (n, e) = (33, 3)$$

$$\text{Private key} = (n, d) = (33, 7)$$

### Encrypt:

$$P=7$$

$$C=7^{e=3} \pmod{33} = 13$$

### Decrypt:

$$C=13$$

$$P=C^{d=7} \pmod{33} = 7$$

### Verify Signature on Data:

$$\text{Data}=7; \text{Signature}=28$$

$$(7 \equiv 28^{e=3} \pmod{33})$$

### Sign Data:

$$\text{Data}=7$$

$$\text{Signature}=7^{d=7} \pmod{33} = 28$$

## Symmetric vs Public

Assurance	Prevents	Secret Key	Public Key
Confidentiality	Snooping	X	X
Authentication	Masquerading	X	X
Integrity	Message alteration w/o detection	X	X
Nonrepudiation	Sender's false denial		X

## Symmetric vs Public

Attribute	Secret Key	Public Key
Years in use	Thousands	Less than 50
Current standard	DES, 3DES, IDEA	RSA, Diffie-Hellman, DSA, elliptic curve?
Speed	Fast	Slow
Keys	Shared secret between at least two people	Private: kept concealed by one person Public: widely distributed
Non-repudiation	No Need trusted third party to act as witness	Yes Digital signatures: don't need trusted third party
Key length	56-bit obsolete 128-bit considered safe	1,024 suggested (RSA) ~172 (elliptic curve)

## Digital Signature

User A wants to sign a sequence of bits X. He performs the following operations:

1. He computes the digest  $d = \text{hash}(X)$  using a cryptographically secure hash func.
2. He generates a public/private key pair  $K_{Ua}/K_{Ra}$
3. He encrypts the digest using the private key,  $c = E_{K_{Ra}}[d]$
4. The string c is the digital signature for X by A.

Anybody who has  $K_{Ua}$  can verify the signature by computing

$$D_{K_{Ua}}[c] == \text{hash}(X)$$

Only A can have created the digital signature because only A has  $K_{Ra}$ .

## HMAC (cryptographic hash code)

HMAC( $M, K$ ): hashing  $M$  with a key  $K$

1.  $K+ = K, 44*0x00$
2. Return  $H[(K+ + 0x5C), H[(K+ + 0x36), M]]$

Advantages:

- Works with any hash function  $H$
- Same performance as  $H$
- Handles key  $K$  in a simple way
- Well-understood cryptographic analysis
- Faster in software than DES
- No export restrictions
- Library code widely available

## Public-key Encryption Attacks

Typical attacks

- Given the public key find the prime factors  $p$  and  $q$  such that  $n=p*q$ .  
This is technically possible but very slow.  
No polynomial algorithm for this factorization is known.

2) Reply Attack

3) Man-in-the-middle attack

For more secure encryption

- Use a large public key (large primes  $p,q > 1024$  bits)
- Timestamp data
- Use digital certificates and/or third party certification
- Use public key to encrypt short messages only such as another key.

## Authentication. How?

Other common examples of access control involving authentication include:

- withdrawing cash from an [ATM](#).
- controlling a remote computer over the [Internet](#).
- using an Internet banking system.

The methods by which a human can authenticate themselves are generally classified into three cases:

- Something the user **is** (fingerprint, retinal pattern, DNA sequence, voice)
- Something the user **has** (ID card)
- Something the user **knows** (password, a pass phrase, a PIN)

**digital signatures** are a sequence of bits that permit authentication of digital information. They are often treated, sometimes too closely, as analogous to a physical signature on paper.

A **challenge-response test** is a test involving a set of questions (or "challenges"), that a system has to answer in order to identify itself. A very simple example is asking for a password.

## Send public keys

### Method 1 – Public Announcement of Public Keys

Attach the public key to your emails or post it on some web site so that it is available to everybody.

### Method 2 – Use a Public Directory

The directory maintains a database of ID,KU

Registration is performed via encrypted communication

Keys can be updated, works like telephone book

### Method 3 – Public Key Authority (\*)

Keys are not published but are stored by the Public Key Authority. Authorized users can retrieve them from the public key authority using the public key encryption to communicate with authority.

### Method 4 – Public Key Certificates (\*\*)

Parties authenticate each other by exchanging digital certificates provided by a trusted authority

## NOTATION

- ID(a) = Identity of a  
 Ks = Session key (symmetric)  
 KUa = a public key (RSA or elliptic curves)  
 KRa = a private key (RSA or elliptic curves)  
 $E_{KUa}[P]$  = Plaintext encrypted using RSA and a public key  
 $D_{KRa}[C]$  = Ciphertext decrypted using RSA and a private key  
 [X,Y,Z] = a new message build concatenating the strings X, Y and Z

Examples

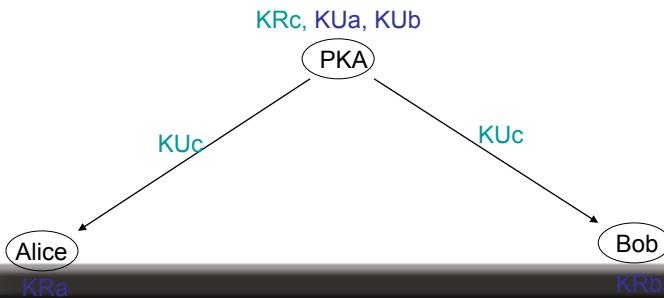
$E_{KUb}[ID(a),KUa]$  reads:  
combine the identity of a with its public key and encrypt it with b public key.



## Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

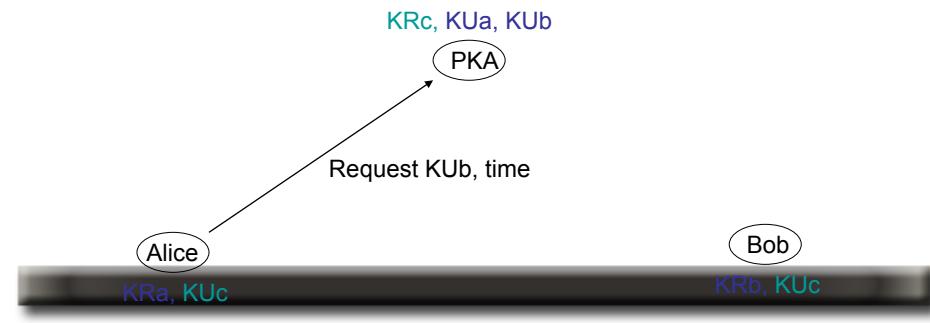
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

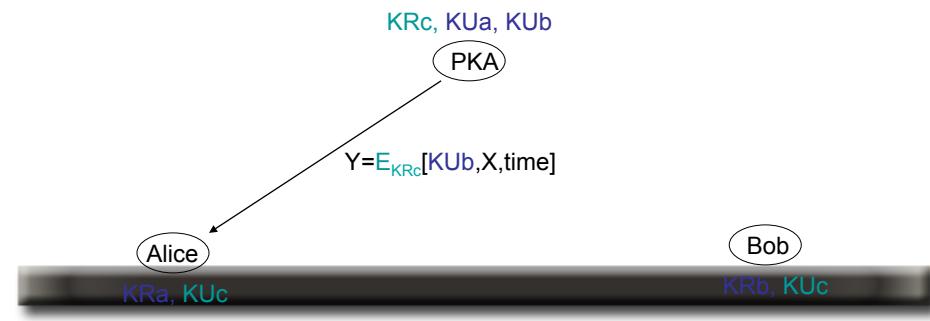
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) **PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$**
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

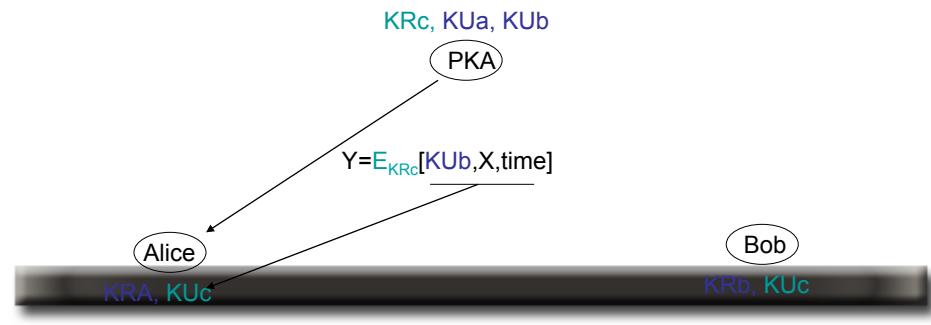
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

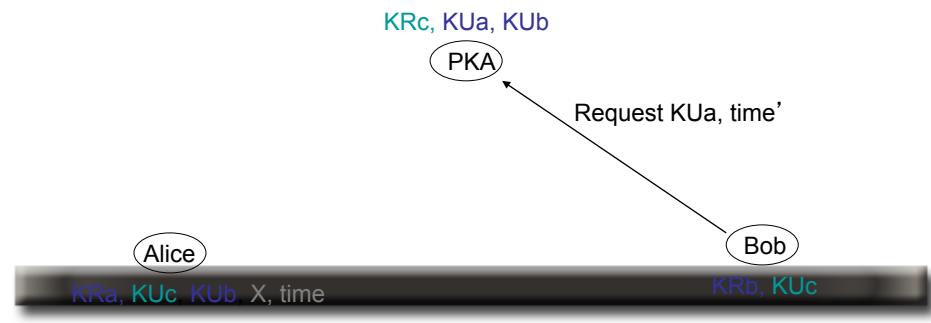
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

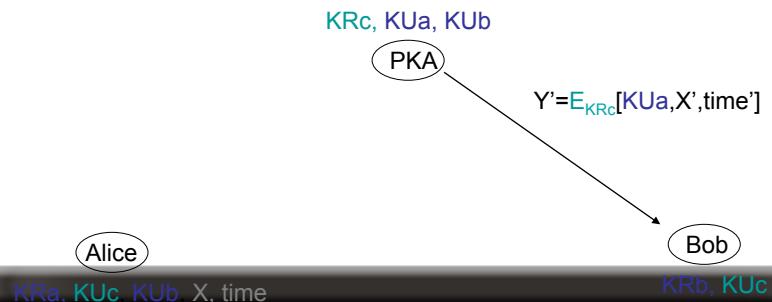
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

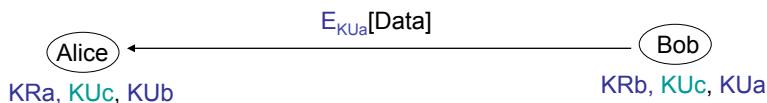
now A and B can communicate.



### Method 3 – Public Key Authority (\*)

- 1) All parties are issued KUc (the PKA public key)
- 2) A sends time-stamped request X to PKA asking for KUb
- 3) PKA responds with  $Y = E_{KRc}[KUb, X, \text{time}]$
- 4) A uses KUc to decrypt  $[KUb, X, \text{time}] = D_{KUc}[Y]$  and stores KUb
- 5) B retrieves KUa in the same manner

now A and B can communicate.



### Method 4 – What is certificate?

A **certificate** is a string  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$  containing:

- the time the certificate was issued
- the ID of the certificate recipient
- and the public key of the certificate recipient

A certificate must be signed with the private key of the Certificate Authority

**Advantages:**

Anybody can verify the certificate because the public key of the CA is public

Only CA can have written the certificate because nobody else has the CA private key

The Certificate certifies that the public key of the recipient is associate with its ID

As long one can trust the CA, the public key of the recipient must be authentic.

## Method 4 – What is a Certificate Authority?

A certificate authority or certification authority (CA) is an entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CA's are characteristic of many public key infrastructure (PKI) schemes.

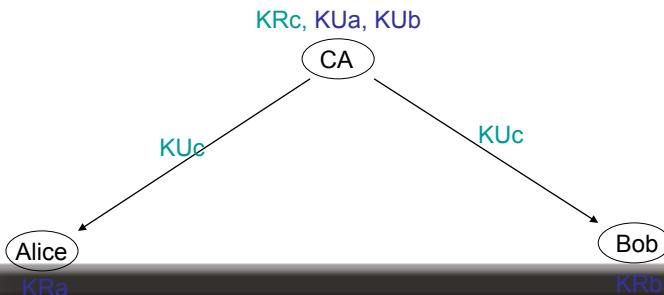
Examples:

- <http://www.verisign.com/> (\$599/year)
- <http://www.thawte.com/> (\$199/year)
- <http://www.cacert.org/> (free)
- <http://www.digitalcertificate.us/>
- <http://www.ancert.com/>

## Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its  $KUc$  to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via  $KUc$

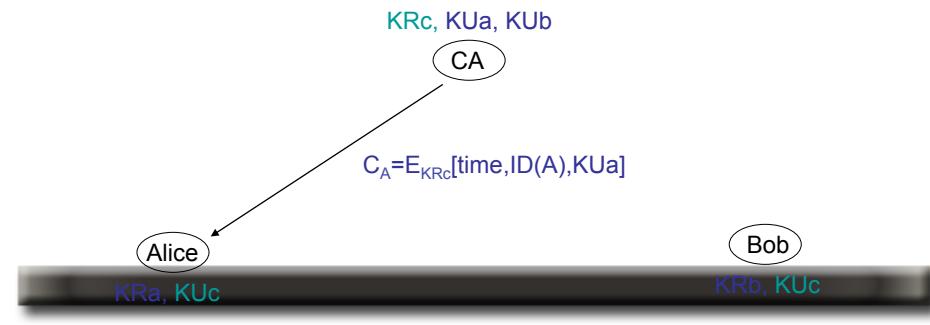
now A and B can communicate.



## Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUC to A and B
  - 2) The CA issues a certificate to A,  $C_A = E_{KRC}[\text{time}, \text{ID}(A), KUa]$
  - 3) The CA issues a certificate to B,  $C_B = E_{KRC}[\text{time}', \text{ID}(B), KUb]$
  - 4) A and B exchange their certificate which they can decrypt via KUC

now A and B can communicate.



## Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUC to A and B
  - 2) The CA issues a certificate to A,  $C_A = E_{KRC}[\text{time}, \text{ID}(A), KUA]$
  - 3) The CA issues a certificate to B,  $C_B = E_{KRC}[\text{time}', \text{ID}(B), KUB]$
  - 4) A and B exchange their certificate which they can decrypt via KUC

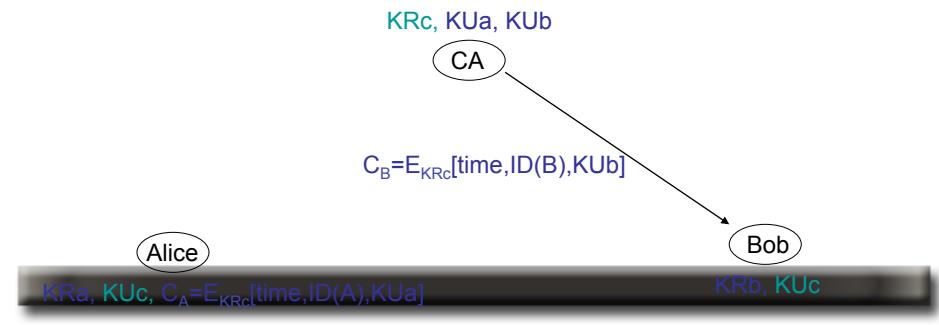
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

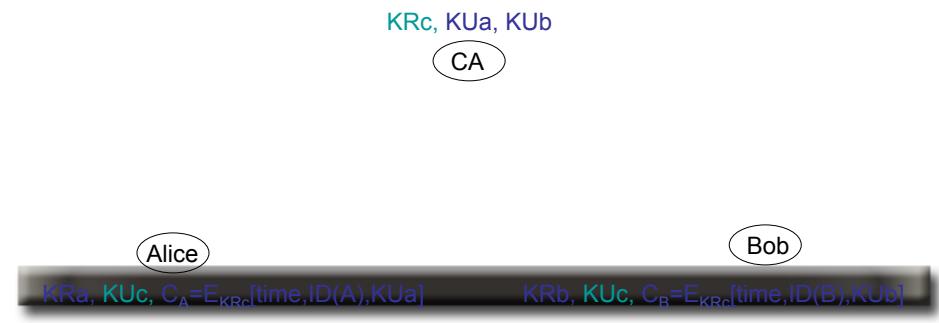
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

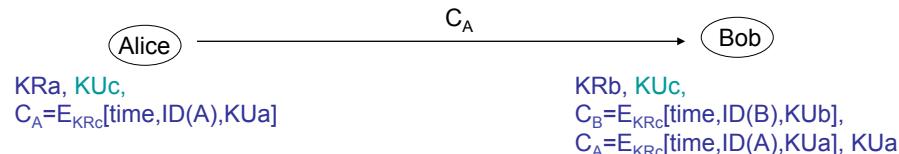
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

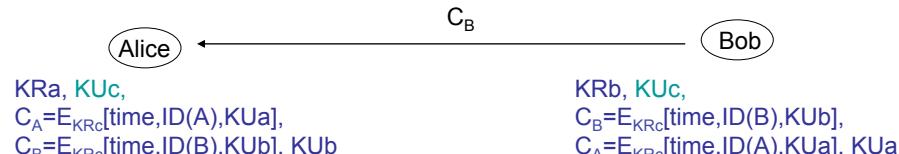
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

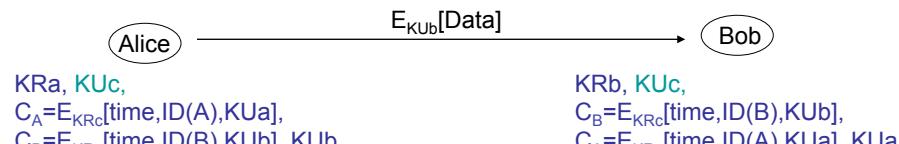
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

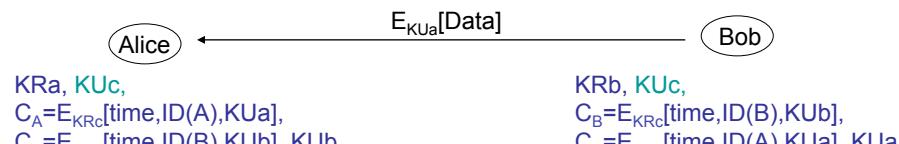
now A and B can communicate.



### Method 4 – Public Key Certificates (\*\*)

- 1) CA sends its KUc to A and B
- 2) The CA issues a certificate to A,  $C_A = E_{KRc}[\text{time}, \text{ID}(A), KUa]$
- 3) The CA issues a certificate to B,  $C_B = E_{KRc}[\text{time}', \text{ID}(B), KUb]$
- 4) A and B exchange their certificate which they can decrypt via KUc

now A and B can communicate.



## ITU-T, X.509, and S/MIME

**ITU-T:** An international organization that defines standards for telegraphic and telephone equipment. For example, the Bell 212A standard for 1200-bps communication in North America is observed internationally as ITU-T V.22. For 2400-bps communication, most U.S. manufacturers observe V.22 bis.

**X.509:** It is a specification for digital **certificates** published by the ITU-T (International Telecommunications Union - Telecommunication). It specifies information and attributes required for the identification of a person or a computer system. Version 3 (X.509v3) defines the format for certificate extensions, used to store additional information about the certificate holder (e.g. academic degrees, position within a company, authorities to sign) or to define (resp. limit) the certificate usage (e.g. private use, official use, for signing, for encryption, etc.).

**S/MIME:** Secure Multipurpose Internet Mail Extension: A de-facto standard that was developed by RSA Data Security Inc. for encrypting and signing e-mails using digital certificates based on **X.509v3**.

## X.509 specs

A X.509 certificate (v3) contains:

- Version
- **Certificate unique ID** (generated by the Certificate authority)
- Algorithm identifier
- Issuer Name (verisign? SnakeOil?)
- Life span (start and end dates)
- Subject Name (A)
- Public Key of Subject (KUa)
- Issuer Unique ID
- Subject Unique ID
- Optional Parameters
- Digital Signature (hash of 1...10 encrypted with issuer's private key)

## X.509 specs

A X/509 certificate revocation (v3) contains:

- Algorithm identifier
- Update date
- Next update date
- **Certificate unique ID**
- Revocation date
- ...
- **Certificate unique ID**
- Revocation date
- Digital Signature (hash of 1...8 encrypted with issuer's public key)

Certificate revocation is issued when:

- User is no longer certified
- User public key is compromised
- Certificate Authority's key is compromised

## X.509 specs

Any application that accepts X.509 certificates should contact the CA and make sure the certificate is not revoked.

## Certificates catch!

A certificate does not prove you are. It proves that something digitally signed with your private key originated from you.

## Using openssl (self signed certificate)

Self-signed certificates (useful only for testing):

```
openssl genrsa -out PRIVATE_KEY 1024
openssl rsa -in PRIVATE_KEY -pubout -out PUBLIC_KEY
openssl req -new -days VALID_DAYS -key PRIVATE_KEY -x509 -out CERTIFICATE_FILE
```

Where:

- PRIVATE\_KEY is the name of the file where the private key will be written
- PUBLIC\_KEY is the name of the file where the public key will be written
- CERTIFICATE\_FILE is the name of the file that will contain the certificate

For apache2 certificates are located in /etc/apache2/ssl/

Attention:

- Must activate the certificates

## Using openssl (request for certificates)

Request for certificates:

```
openssl genrsa -out PRIVATE_KEY 1024
openssl rsa -in PRIVATE_KEY -pubout -out PUBLIC_KEY
openssl req -new -days VALID_DAYS -key PRIVATE_KEY -out SIGNING_REQUEST
```

Where:

- PRIVATE\_KEY is the name of the file where the private key will be written
- PUBLIC\_KEY is the name of the file where the public key will be written
- CERTIFICATE\_FILE is the name of the file that will contain the certificate

Send the SIGNING\_REQUEST to the CA and they will send back a certificate file.

## Kerberos

**Kerberos** is a computer network authentication protocol developed at MIT (Athena project) which allows individuals communicating over an insecure network to prove their identity to one another in a secure manner (single sign-on).

Kerberos prevents eavesdropping or replay attacks, and ensures the integrity of the data.

Its designers aimed primarily at a client-server model, and it provides mutual authentication — both the user and the service verify each other's identity.

Kerberos builds on symmetric key cryptography and requires a trusted third party.

## Kerberos - Threats

A user may gain access to a workstation and pretend to be another user from that workstation

A user may later the network address of a workstation and pretend to be connecting from a different workstation

A user may eavesdrop and use reply attack.

## Kerberos - Requirements

Secure: eavesdropper should not be able to obtain all necessary information (obtained via a secret key)

Reliable: Distributed server architecture

Transparent: easy to use, only requires password

Scalable: Support a large number of users and workstations

## Kerberos - Components

User and Client PC: Seeks access to remote workstation

Workstation: Resource the User from Client PC seeks access to.

Authentication Server: system in charge of verifying user access right and creating one time session keys and tickets.

A set of Users, Workstations and one Authentication service define a realm.

## Kerberos

Each party (**Alice**, **Bob**) shares a unique symmetric key with the Authentication Service (**S**)

**A** is the identity of Alice, **B** is the identity of Bob

**KBS** is the symmetric key shared by Bob and S

**KAS** is the symmetric key shared by Alice and S

If Alice wants to talk to Bob:

- 1) **Alice** sends to **S** her ID and Bob's ID  
A, B
- 2) **S** generates a random key KAB and sends back to **Alice**  
 $E_{KAS}[TS, L, KAB, A, E_{KBS}[TS, L, KAB, A]]$
- 3) **Alice** verifies the message and send to **Bob**  
 $E_{KBS}[TS, L, KAB, A], E_{KAB}[TA, A]$
- 4) **Bob** sends back to **Alice** the acknowledgement  
 $E_{KAB}[TA+1]$

## Kerberos. V4 vs V5

Kerberos V5:

- 1) Supports multiple encryption algorithms (V4 only DES)
- 2) Supports multiple network protocols (V4 only IPv4)
- 3) Specifies a standard message byte ordering (V4 didn't)
- 4) Ticket lifetime can be longer than 1280 minutes (limit for V4)
- 5) Forbids authentication forward (security hole in V4)
- 6) Allows Inter-realm authentication.

## Secure EMail

Given some secure Email text X:

1. Create a session key K
2. Encrypt X with session key K (for example using DES),  $Y=E_K[X]$
3. Encrypt K using public key of email recipient,  $c=E_{Kub}[K]$
4. Combine c with Y, encrypted message=[c,Y]
5. Recipient has KR<sub>a</sub> and decrypt K and use K to decrypt text X.

For additional protection the text itself should contain a hash value.

## **Topic: IP Security**

**ECT582**

**Secure  
Electronic  
Commerce**



ECT582 - M. Di Pierro @ DePaul CTI

### **IPsec**

Example of applications:

- Secure branch offices connectivity over the internet
- Secure remote access over the internet
- Establishing extranet and intranet connectivity with partners
- Enhancing electronic commerce security

IPsec provides point-to-point encryption where each point is network device such as router. IPsec is also implemented at the OS level windows and Linux.

Include two different protocols: AH and ESP

## IPsec benefits

- If implemented in router/firewall it can be applied to all traffic
- IPsec cannot be bypassed if IP traffic only
- IPsec just below transport layer, no need to change software or protocols
- Transparent to users
- Can provide security for individual users if needed.

## IPsec Definitions

AH = Authentication Header (an header is added to packets)

ESP = Encapsulating Security Payload (all packets are encrypted + header)

Both protocols provide the following services:

- Access control (AH,ESP)
- Connectionless Integrity (AH, ESP optional)
- Data Origin Authentication (AH, ESP optional)
- Rejection of replayed packets (ESP)
- Confidentiality (ESP)

## IPsec Security Associations

SA = Security Association is a one-way relationship between sender and receiver.

An SA is identified by the following info carried in the AH or ESP:

- Security Protocol Identifier (AH or ESP?)
- Security Parameter Index: A string of bits having only local significance that allows the receiver to decide how to process the information
- IP Destination address (behind router/firewall address)
- Sequence number counter

## IPsec Transport or Tunnel

Both AH and ESP support two security modes:

Transport: Each packet payload is encrypted (used in host-to-host)

Tunnel: Protection for entire IP packet: header + payload. (used in net-to-net)

## IPsec AH

AH contains:

Next header  
Payload length  
Security parameter index  
Sequence number  
Authentication Data (for integrity check of the payload, usually HMAC)  
Payload (not encrypted)

## IPsec ESP

ESP contains:

Security parameter index  
Sequence number  
Payload + padding + padding length (encrypted)  
Next header (encrypted)  
Authentication Data (for integrity check of payload, usually HMAC)

IPsec ESP encryption supports:

3DES, RC5, IDEA, 3IDEA, CAST, Blowfish

## IPsec Key Management

Manual: Configured by sysadmin (most secure way as long as trust in sysadmin)

Automated: Automated using Oakley Key Distribution Protocol

Oakley Key Distribution uses Diffie-Hellman for key exchange but:

- 1) Allows parties to negotiate parameters
- 2) Uses cookies (random numbers in cookies) to prevent key exchange from forged addresses and therefore prevent clogging attacks
- 3) Authenticates the Diffie-Hellman to prevent man-in-the-middle attack (support for Digital signatures, Public key encryption, Symmetry key encryption)

# Game Physics in a Nutshell

Massimo Di Pierro

## Contents

<b>1 Definitions</b>	<b>2</b>
<b>2 Notation</b>	<b>3</b>
<b>3 Newton First and Second Laws</b>	<b>4</b>
<b>4 Types of Forces and Newton Third Law</b>	<b>6</b>
4.1 Gravity . . . . .	6
4.2 Spring . . . . .	7
4.3 Friction . . . . .	7
<b>5 Rotations</b>	<b>7</b>
<b>6 Spinning Objects</b>	<b>9</b>
<b>7 Newton Second Law for Rotation</b>	<b>10</b>
<b>8 Assembling Rigid Bodies</b>	<b>13</b>
<b>9 Quaternions</b>	<b>13</b>
<b>10 Conservation Laws</b>	<b>14</b>
<b>11 Collisions</b>	<b>15</b>
<b>12 Geometry of collision</b>	<b>18</b>
<b>13 Implementation</b>	<b>19</b>

Source: <https://github.com/mdipierro/cylon>

## 1 Definitions

Disclaimer :*This document is a work in progress. Here only talk about classical physics (200 years old stuff) and all definitions and formulas are consistent within that framework. Classical physics works well for objects that have relative velocity much smaller than the speed of light and are macroscopic thus not affected by the measurement process.*

The word **Physics** comes the greek *physiké* which means *science of nature*. It is the study of natural phenomena by means of quantitative observations and formal languages such as mathematics. The ultimate goal of physics is that describing the entire universe using few fundamental laws expressed in the mathematical languages.

A **Cartesian coordinate system** is a framework that allows to specify each point uniquely using numerical coordinates, which are the distances from the point to D fixed perpendicular directed lines, measured in the same unit of length. Each reference line is called a coordinate axis or just axis of the system, and the point where they meet is its origin. The coordinates can also be defined as the positions of the perpendicular projections of the point onto the two axes, expressed as signed distances from the origin. Given two points identified by coordinates  $\vec{p} = (p_x, p_y, p_z)$  and  $\vec{q} = (q_x, q_y, q_z)$ , the distance between the two points is given by

$$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \quad (1)$$

and indicated with  $d_{pq} = |\vec{p} - \vec{q}|$ .

A **reference frame** or **observational frame of reference** is a set of axes and their orientation, tied together with information about the motion of the observer. For example you can have two observers at the same location at the same moment in time but in motion one respect to the other. They may use the same set of axes and the same orientation yet, when measuring speed of objects respect to their own speed they will find different values. In particular if we get in the shoes of observer  $O$  and we measure the speed of point  $P$  we find veloticy  $\vec{v}$ . If we instead get of the shoes of observer  $O'$  and we measure the speed of point  $P$  we find a velocity  $\vec{v}'$ . This means that the velocity of observer  $O'$  as measured by  $O$  is  $\vec{w} = \vec{v} - \vec{v}'$  and the veloticy of  $O$

as measured by  $O'$  is  $\vec{w}' = -\vec{w} = \vec{v}' - \vec{v}$ . This law of composition of velocities is called **Galilean Invariance**.

A reference frame is called an **intertial reference frame** if it describes time and space homogeneously, isotropically, and in a time-independent manner. An observer is in an inertial reference frame if the observer feels no force acting upon him/her. All inertial frames are in a state of constant, rectilinear motion with respect to one another. If an observer  $A$  in an inertial reference frame and he/she measures the position and velocity of a point  $P$  as  $\vec{p}, \vec{v}_p$ , then a different observer  $B$  from a reference frame moving at velocity  $\vec{w}$  respect to  $A$  will measure different position and velocity for the same point  $P$ ,  $\vec{p}' = \vec{p} - \vec{w}t, \vec{v}'_p = \vec{v}_p - \vec{w}$ . Laws of physics like  $F = ma$  are normally formulated in an inertial reference frame and require “corrections” when used in a non-inertial reference frame.

**Degrees of freedom** are the number of parameters required to specify the state of a system. A particle is only identified by its coordinates (3 numbers in 3D). A rigid body is identified by its position and its orientation ( $3+3=6$  numbers in 3D). A system comprised of  $n$  point particles and  $m$  constraints has  $3n - m$  degrees of freedom. A system comprised of  $n$  rigid bodies and  $m$  constraints has  $6n - m$  degrees of freedom.

## 2 Notation

3D Vectors are indicated with a *vector* on top, as in  $\vec{p}$ . The norm of a vector is indicated with  $|\vec{p}|$  or with the same letter as the vector without decoration,  $p$ . The direction of a vector  $\vec{p}/p$  is indicated with a *hat*,  $\hat{p}$  and it is called a *versor*.

We use the letter  $t$  to label time, the label  $i$  to label parts of a system (for example components of a rigid body), the labels  $j = 0, 1, 2 = X, Y, Z$  and  $k = 0, 1, 2 = X, Y, Z$  to indicate components of a vector. Quaternions also have an extra index  $j, k = W = 3$ . 4D Vectors (used here to represent quaternions) are marked with a *tilde* superscript, as in  $\tilde{\theta}$ .

Here is a legend for the notation used in these notes:

Name	Symbol
point	$A, B, P, \dots$
position	$\vec{p}, \vec{p}_A, \vec{q}, \dots$
component of a vector	$p_k$ for $k = 1, 2, 3$ or $k = X, Y, Z$
velocity	$\vec{v}$
acceleration	$\vec{a}$
momentum	$\vec{K} = m\vec{v}$
mass	$m$
force	$\vec{F}$
rotation	$R$
angular velocity	$\vec{\omega}$
angular acceleration	$\vec{\alpha}$
angular momentum	$\vec{L} = I\vec{\omega}$
inertia tensor	$I$

We use the symbol  $=$  to indicate an equivalence (result of an observation, consequence of a formula, or an assumption) but we use the symbol  $\equiv$  to indicate a definition (the left hand term defined in terms of the right hand term).

### 3 Newton First and Second Laws

Classical Mechanics starts with Newton's Laws. The first two of them can be summarized as follows:

When observed from an inertial reference frame, objects like to move at constant velocity. If an object changes its velocity we say it is subject to a force. A force is something external that acts on the object and is proportional to its acceleration. The proportionality factor is a property of the object which we call mass.

Let's measure the position of an object at different times...

time	$t$	$t + dt$	$t + 2dt$	$t + 3dt$	$t + 4dt$	$t + 5dt$	$\dots$
position	$\vec{p}_t$	$\vec{p}_{t+dt}$	$\vec{p}_{t+2dt}$	$\vec{p}_{t+3dt}$	$\vec{p}_{t+4dt}$	$\vec{p}_{t+5dt}$	$\dots$

We define the velocity as the change in position per unit of time:

$$\vec{v}_t \equiv \frac{\vec{p}_{t+dt} - \vec{p}_t}{dt} \quad (2)$$

and we define the acceleration as the change in velocity per unit of time

$$\vec{a}_t \equiv \frac{\vec{v}_{t+dt} - \vec{v}_t}{dt} \quad (3)$$

time	$t$	$t + dt$	$t + 2dt$	...
position	$\vec{p}_t$	$\vec{p}_{t+dt}$	$\vec{p}_{t+2dt}$	...
velocity	$\vec{v}_t = \frac{\vec{p}_{t+dt} - \vec{p}_t}{dt}$	$\vec{v}_{t+dt} = \frac{\vec{p}_{t+2dt} - \vec{p}_{t+dt}}{dt}$	...	...
acceleration	$\vec{a}_t = \frac{\vec{v}_{t+dt} - \vec{v}_t}{dt}$	...	...	...

If we can determine  $\vec{p}$  at different times, we can compute  $v$  and  $a$  at different times. Notice that if the velocity is constant the acceleration is zero:

$$\vec{a}_t \equiv (\vec{v}_{t+dt} - \vec{v}_t)/dt = 0 \quad (4)$$

We now solve eq. 2 and eq. 3 in  $\vec{p}_{t+dt}$  and  $\vec{v}_{t+dt}$  respectively. We find:

$$\vec{p}_{t+dt} = \vec{p}_t + \vec{v}_t dt \quad (5)$$

$$\vec{v}_{t+dt} = \vec{v}_t + \vec{a}_t dt \quad (6)$$

The Newton's laws (as stated above) say that if the velocity changes. I.e. there is an acceleration, then the object is subject to a force. The law does not say that if two objects are subject to the same force, they will have the same acceleration. Therefore, it is reasonable to assume that

$$\vec{F}_t \propto \vec{a}_t \quad (7)$$

and the proportionality factor is different for different objects. We call this proportionality factor  $m$ :

$$\vec{F}_t = m\vec{a}_t \quad (8)$$

We replace the solution of eq. 8 for  $a_t$  in eq. 6 and we obtain:

$$\vec{p}_{t+dt} = \vec{p}_t + \vec{v}_t dt \quad (9)$$

$$\vec{v}_{t+dt} = \vec{v}_t + (1/m)\vec{F}_t dt \quad (10)$$

In other words: *if we know the position and the velocity of the object, and the force acting on the object at time  $t$  we can compute the position and the velocity at time  $t + dt$ .*

The set of eqs. 9 and 10 are called an *Euler integrator*. Technically they are only correct in the limit  $dt \rightarrow 0$  but they provide a decent approximation for small  $dt$  if the force does not change significantly over the time interval  $dt$ .

So far we assumed  $m$  is constant. What if  $m$  changes with time? A more accurate way to write the Newton equation is in terms of a quantity we call *momentum*, defined as:

$$\vec{K}_t \equiv m_t \vec{v}_t \quad (11)$$

In terms of  $\vec{K}_t$ , eq. 8 can be written as

$$\vec{F}_t = \frac{\vec{K}_{t+dt} - \vec{K}_t}{dt} \quad (12)$$

Now we can perform a change of variables from  $v$  to  $K$  and rewrite the Euler integrator as follows:

$$\vec{F}_t = \sum_i \vec{F}_i \quad (\text{compute force}) \quad (13)$$

$$\vec{v}_t = m_t^{-1} \vec{K}_t \quad (\text{compute velocity}) \quad (14)$$

$$\vec{p}_{t+dt} = \vec{p}_t + \vec{v}_t dt \quad (\text{update position}) \quad (15)$$

$$\vec{K}_{t+dt} = \vec{K}_t + \vec{F}_t dt \quad (\text{update momentum}) \quad (16)$$

In other words: *if we know the position and the momentum of the object, and the force acting on the object at time  $t$ , we can compute the position and the momentum at time  $t + dt$ . If we know the mass of the object we can compute the velocity from the momentum.*

If multiple forces act on the same object,  $\vec{F}$  is the sum of those forces:

$$\vec{F} = \sum_i \vec{F}_i \quad (17)$$

where  $\vec{F}_i$  is the force caused by iteration  $i$  (could be gravity, could be a spring, etc.). This called *d'Alambert's principle*.

## 4 Types of Forces and Newton Third Law

Here we are exclusively interested in the following types of forces:

### 4.1 Gravity

$$\vec{F}_t^{gravity} \equiv (0, -m_t g, 0) \quad (18)$$

where  $g = 9.8$  meters/second.

## 4.2 Spring

$$\vec{F}_t^{spring} \equiv \kappa(|\vec{q}_t - \vec{p}_t| - L) \frac{\vec{q}_t - \vec{p}_q}{|\vec{q}_t - \vec{p}_t|} \quad (19)$$

where  $\vec{p}_t$  is the position of the mass at time  $t$ ,  $q_t$  is the position of the other end of the spring at the same time,  $\kappa$  is a constant that describes the rigidity of the spring, and  $L$  is the rest length of the spring. Notice that when  $|\vec{p}_t - \vec{q}_t| = L$  there is no force, when  $|\vec{p}_t - \vec{q}_t| > L$  the force is attractive and when  $|\vec{p}_t - \vec{q}_t| < L$  the force is repulsive.

## 4.3 Friction

$$\vec{F}_t^{friction} \equiv -\gamma \vec{v}_t \quad (20)$$

$\gamma$  is called the *friction coefficient*.

Newton's third law states that if a body A exercises a force  $F$  on a body B, then body B exerces a force  $-F$  on body A. In any simulation we have to account for this using force generators that link objects to each other.

## 5 Rotations

A position vector  $\vec{p}$  can be multiplied by a matrix

$$\vec{p}' = R\vec{p} \quad (21)$$

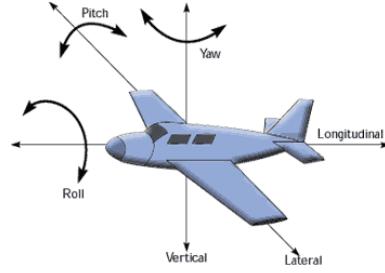
The matrix multiplication maps one vector  $p$  onto another vector  $p'$ . If we now require that the matrix  $R$  have determinant equal to 1 it follows that for every vector  $p$

$$|\vec{p}'| = |R\vec{p}| = |R||\vec{p}| = |\vec{p}| \quad (22)$$

i.e. the  $R$  transformation preserves the vector length. If we also requires that the inverse of  $R$  be the same as its transposed ( $R^{-1} = R^t$ ) then, for every two vectors  $p$  and  $q$  we obtain:

$$\vec{p}' \cdot \vec{q}' = (R\vec{p}) \cdot (R\vec{q}) = \vec{p} \cdot \vec{q} \quad (23)$$

i.e. the  $R$  transformation preserves scalar products (and therefore angles). A matrix  $R$  meeting the two conditions above is called an *orthogonal matrix* or a *rotation*.



A rotation of the angle  $\theta$  around the  $X$ -axes can be written as

$$R_X(\theta) \equiv \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \quad (24)$$

A rotation of the angle  $\theta$  around the  $Y$ -axes can be written as

$$R_Y(\theta) \equiv \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \quad (25)$$

A rotation of the angle  $\theta$  around the  $Z$ -axes can be written as

$$R_Z(\theta) \equiv \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (26)$$

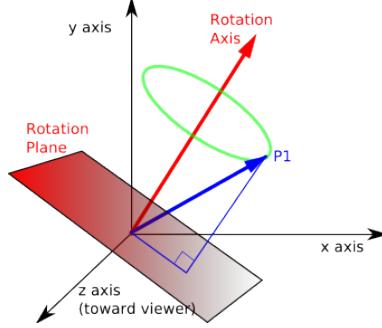
A rotation of the angle  $\theta$  around an arbitrary direction  $\hat{n} = (n_x, n_y, n_z)$  is given by:

$$R_{\hat{n}}(\theta) = \begin{pmatrix} tn_x n_x + c & tn_x n_y - sn_z & tn_x n_z + sn_y \\ tn_x n_y + sn_z & tn_y n_y + c & tn_y n_z - sn_x \\ tn_x n_z - sn_y & tn_y n_z + sn_x & tn_z n_z + c \end{pmatrix} \quad (27)$$

where  $c = \cos(\theta)$  and  $s = \sin(\theta)$  and  $t = (1 - \cos(\theta))$ .

It is also convenient to merge the directional information  $\hat{n}$  with the angular information  $\theta$  into a single vector  $\vec{\theta} \equiv \theta \hat{n}$  and use the more compact notation:

$$R(\vec{\theta}) \equiv R_{\hat{n}}(\theta) \quad (28)$$



A rotation can also be used to represent the orientation of a body.

If a body is oriented according to  $R(\vec{\theta})$  and it gets rotated by another matrix  $R(\vec{\beta})$  the body will end up being oriented in a different direction:

$$R(\vec{\theta}') = R(\vec{\beta})R(\vec{\theta}) \quad (29)$$

notice that

$$\vec{\beta} = (\beta m_x, \beta m_y, \beta m_z) \quad (30)$$

$$\vec{\theta} = (\theta n_x, \theta n_y, \theta n_z) \quad (31)$$

$$\vec{\theta}' = (\theta' n'_x, \theta' n'_y, \theta' n'_z) \quad (32)$$

For an infinitesimal rotation  $\beta = \omega dt$  we can write

$$R' = R(\vec{\omega} dt)R \quad (33)$$

where

$$\vec{\omega} \equiv (\omega n_x, \omega n_y, \omega n_z) \quad (34)$$

is called angular velocity. What is the explicit expression for  $R(\vec{\omega} dt)$  and first order in  $dt$ ?

## 6 Spinning Objects

When a body is rotating with constant angular velocity  $\vec{\omega}$ , its rotation is proportional to time  $t$  only. We say the body is spinning. Here we consider body spinning around the direction  $\hat{n} = (n_x, n_y, n_z)$  of an angular velocity  $\omega$ . The angular position of the object at time  $t$  is therefore  $\vec{\theta}_t = \omega t \hat{n}$ . Its orientation at time  $t$  is therefore given by:

$$\vec{\theta}_t \equiv (\omega t n_x, \omega t n_y, \omega t n_z) \quad (35)$$

So if we now consider the position of a vector  $\vec{p}$  subject to rotation we obtain:

$$\vec{p}_t = R(\vec{\theta}_t) \vec{p} \quad (36)$$

and we can compute its velocity using the definition:

$$\vec{v}_t = (\vec{p}_{t+dt} - \vec{p}_t)/dt \quad (37)$$

$$= (R(\vec{\theta}_{t+dt})\vec{p} - R(\vec{\theta}_t)\vec{p})/dt \quad (38)$$

$$= \vec{\omega} \times \vec{p} \quad (39)$$

where  $\vec{\omega}$  is the angular velocity, same as eq. 34.

## 7 Newton Second Law for Rotation

Before we have formulated the second law of Newton as

$$\vec{F}_t = \frac{\vec{K}_{t+dt} - \vec{K}_t}{dt} \quad (40)$$

where  $\vec{K}_t = m_t \vec{v}_t$ . We now consider a rigid body comprised of one mass  $m$  at position  $\vec{p}$ , connected by a solid rod at the origin of the axes. The mass is subject to a force  $\vec{F}$ .

If there were no rod, the mass would move according to the Newton equation above. Because of the rod, the mass is not free and it feels only the component of the force orthogonal to the direction of the rod  $\vec{r}$  which is the same as the position of the mass  $\vec{p}$  because the rod is pinned at the origin of the axes. In order to apply Newton law only to the components orthogonal to  $r$  we perform a cross product of both terms

$$\vec{r}_t \times \vec{F}_t = \vec{r}_t \times \frac{\vec{K}_{t+dt} - \vec{K}_t}{dt} \quad (41)$$

$$= \frac{\vec{r}_t \times \vec{K}_{t+dt} - \vec{r}_t \times \vec{K}_t}{dt} \quad (42)$$

and if we define the *torque* as

$$\vec{\tau} \equiv \vec{r} \times \vec{F} \quad (43)$$

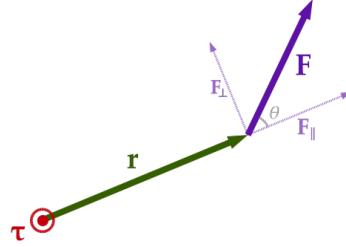
and the *angular momentum* as

$$\vec{L} \equiv \vec{r} \times \vec{K} = m\vec{r} \times \vec{v} \quad (44)$$

we can rewrite eq. 42 as

$$\vec{\tau}_t = \frac{L_{t+dt} - L_t}{dt} \quad (45)$$

which is the analogous of *Newton equation for rotations*.



Notice that if we substitute eq. 39 which gives the velocity in terms of the angular velocity into eq. 44 we obtain:

$$\vec{L} = \vec{r} \times (m\vec{\omega} \times \vec{r}) = (m|\vec{r}|^2)\vec{\omega} \quad (46)$$

The coefficient  $(m|\vec{r}|^2)$  is called *moment of inertia*.

Let us now consider a rigid body comprised of many masses subject to constraints. For each mass  $m_i$  at point  $r_i$  we can write:

$$\vec{\tau}_0 = \vec{L}_0 \quad (47)$$

$$\vec{\tau}_1 = \vec{L}_1 \quad (48)$$

$$\vec{\tau}_2 = \vec{L}_2 \quad (49)$$

$$\dots \quad \dots \quad (50)$$

$$\sum_i \vec{\tau}_i = \sum_i \vec{L}_i \quad (51)$$

$$\vec{\tau} = \vec{L} \quad (52)$$

In the last step we have used the following definitions:

$$\vec{\tau} \equiv \sum_i \vec{\tau}_i = \sum_i \vec{r}_i \times \vec{F}_i \quad (53)$$

and

$$\vec{L} \equiv \sum_i \vec{L}_i = \sum_i \vec{r}_i \times \vec{K}_i = m_i \vec{r}_i \times \vec{v}_i \quad (54)$$

Eq. 45 is still valid, but eq. 54 becomes:

$$\vec{L} = \sum_i \vec{r}_i \times (m_i \vec{\omega} \times \vec{r}_i) = I \vec{\omega} \quad (55)$$

where  $I$  is a 3x3 matrix, called *Inertia Tensor* with components:

$$I_{jk} \equiv \sum_i m_i (\delta_{jk} |\vec{r}_i|^2 - r_{i,j} r_{i,k}) \quad (56)$$

Here  $r_{i,j}$  is the  $j$ -th component (X=0, Y=1, or Z=2) of vector  $\vec{r}_i$ .

Finally we can augment the Euler integrator with equivalent formulas for rotations:

$$\vec{F}_t = \sum_i \vec{F}_i \quad (\text{compute force}) \quad (57)$$

$$\vec{\tau}_t = \sum_i \vec{r}_i \times \vec{F}_i \quad (\text{compute torque}) \quad (58)$$

$$\vec{v}_t = m_t^{-1} \vec{K}_t \quad (\text{compute velocity}) \quad (59)$$

$$\vec{\omega}_t = I_t^{-1} \vec{L}_t \quad (\text{compute angular velocity}) \quad (60)$$

$$\vec{p}_{t+dt} = \vec{p}_t + \vec{v}_t dt \quad (\text{update position}) \quad (61)$$

$$\vec{K}_{t+dt} = \vec{K}_t + \vec{F}_t dt \quad (\text{update momentum}) \quad (62)$$

$$R_t = R(\omega_t dt) R_t \quad (\text{update orientation}) \quad (63)$$

$$\vec{L}_{t+dt} = \vec{L}_t + \vec{\tau}_t dt \quad (\text{update angular momentum}) \quad (64)$$

In other words: *If we know the state of the rigid body characterized by the position  $p_t$  and momentum  $K_t$  of its center of mass, if we know its orientation  $T_t$  and its angular momentum  $L_t$ , and if we know the total force  $F_t$  and the total torque  $\tau_t$  acting on the body, we can compute new state of the rigid body  $(p, K, R, L)$  at time  $t + dt$ .*

Given the position  $p_t$  of the center of mass of the body and its orientation  $\tilde{\theta}_t$ , a generic point  $\vec{r}_i$  of the body, can be rotated and translated into the global reference frame according to:

$$R_t \vec{r}_i + p_t \quad (65)$$

and the velocity of the point is

$$\vec{\omega}_t \times \vec{r}_i + \vec{v}_t \quad (66)$$

The position of a point  $\vec{p}$  rotated by  $R$  around point  $\vec{q}$  is given by:

$$\vec{p}' = R(\vec{p} - \vec{q}) + \vec{q} = R\vec{p} + (1 - R)\vec{q} \quad (67)$$

## 8 Assembling Rigid Bodies

Consider a composite rigid body whose parts are smaller rigid bodies characterized by

Component	Position	Mass	Moment of Inertial
0	$\vec{p}_0$	$m_0$	$I_0$
1	$\vec{p}_1$	$m_1$	$I_1$
2	$\vec{p}_2$	$m_2$	$I_2$
...	...	...	...

The composite object will have:

$$m_{total} = \sum_i m_i \quad (68)$$

$$\vec{p}_{cm} = \sum \vec{p}_i m_i / m_{total} \quad (69)$$

$$I_{total} = \sum_i I_i + \Delta I(m_i, \vec{p}_i - \vec{p}_{total}) \quad (70)$$

where

$$\Delta I(m, \vec{r})_{jk} \equiv m(\delta_{jk}|\vec{r}|^2 - r_j r_k) \quad (71)$$

The object  $I$  is a  $3 \times 3$  matrix and it is called *Inertia Tensor*. The inertia tensor depends on the orientation of the object. If the object is rotated by  $R_t$ , the inertia tensor changes:

$$I_0 \rightarrow I_t = R_t I_0 R_t^{-1} \quad (72)$$

## 9 Quaternions

Quaternions provide a way to represent an orientation in 3D without using a rotation but using a 4-vector instead. They are not used in the code because they are not necessary.

It can be convenient to define a four dimension vector (marked by a tilde):

$$\tilde{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3) = (n_x \sin(\theta/2), n_y \sin(\theta/2), n_z \sin(\theta/2), \cos(\theta/2)) \quad (73)$$

and with a change of variable the matrix of rotation  $R_{\hat{n}}(\theta)$  can be re-written:

$$R(\tilde{\theta}) \equiv \begin{pmatrix} 2(\theta_0\theta_0 + \theta_1\theta_1) - 1 & 2(\theta_0\theta_1 - \theta_2\theta_3) & 2(\theta_0\theta_2 - \theta_1\theta_3) \\ 2(\theta_0\theta_1 + \theta_2\theta_3) & 2(\theta_1\theta_1 + \theta_3\theta_3) - 1 & 2(\theta_1\theta_2 - \theta_0\theta_3) \\ 2(\theta_0\theta_3 - \theta_1\theta_2) & 2(\theta_0\theta_3 + \theta_1\theta_2) & 2(\theta_2\theta_2 + \theta_3\theta_3) - 1 \end{pmatrix} \quad (74)$$

The 4-vector  $\tilde{\theta}$  defines an *orientation* and it is called a *quaternion* although its mathematical properties are not relevant here and are not discussed. The associated matrix  $R(\tilde{\theta})$  can be used to rotate the points of an object into an orientation.

If a body is oriented according to  $R(\tilde{\theta})$  and it gets rotated by another matrix  $R_{\hat{n}}(\beta)$  the body will end up being oriented in a different direction:

$$R(\tilde{\theta}') = R_{\hat{n}}(\beta)R(\tilde{\theta}) \quad (75)$$

What's  $\tilde{\theta}'$  as function of  $\hat{n}$ ,  $\beta$  and  $\tilde{\theta}$ ? An explicit calculation (omitted) reveals:

$$\tilde{\theta}' = \tilde{\theta} + \frac{1}{2}\vec{\beta}\tilde{\theta} \quad (76)$$

where  $\vec{\beta}\tilde{\theta}$  is defined as

$$\vec{\beta}\tilde{\theta} = \begin{pmatrix} -(\beta_0\theta_0 + \beta_1\theta_1 + \beta_2\theta_2) \\ (\beta_0\theta_3 + \beta_1\theta_2 - \beta_2\theta_1) \\ (\beta_1\theta_3 + \beta_2\theta_0 - \beta_0\theta_2) \\ (\beta_2\theta_3 + \beta_0\theta_1 - \beta_1\theta_0) \end{pmatrix} \quad (77)$$

where  $(\beta_0, \beta_1, \beta_2) = (\beta n_x, \beta n_y, \beta n_z)$

Eq. 76 will come up handy later when talking about spinning objects. In this case for an infinitesimal rotation  $\beta = \omega dt$ :

$$\tilde{\theta}' = \tilde{\theta} + \frac{1}{2}\vec{\omega}\tilde{\theta}dt \quad (78)$$

## 10 Conservation Laws

The third Newton law says that for each force  $\vec{F}_i$  there is an opposite force  $\vec{F}_j = -\vec{F}_i$  acting on a different body. This means that we add up all forces acting on all bodies:

$$\sum_i \vec{F}_i = \sum_i d\vec{K}_i/dt = 0 \quad (79)$$

(where we used the dot notation to indicate a derivative). This is equivalent to the statement that

$$\sum_i \vec{K}_i = \text{constant} \quad (80)$$

This is called *conservation of momentum*. It states that the sum of momenta  $K$  of all bodies has to be constant.

It can be proven (but we do not because it requires Lagrangian Mechanics) that for each invariance of the system there is an associated conservation law.

Translation invariance implies *Conservation of momentum*:

$$\sum_i \vec{K}_i = \text{constant} \quad (81)$$

Rotation invariance implies *Conservation of angular momentum*:

$$\sum_i \vec{L}_i = \text{constant} \quad (82)$$

Time translation invariance implies *Conservation of energy*:

$$\sum_i E_i = \text{constant} \quad (83)$$

where  $E_i$ , the energy of body  $i$  and it equal to:

$$E_i = \frac{1}{2}mv_i^2 + \frac{1}{2}I\omega_i^2 - \int_{t_0}^t \vec{F}_i(t) \cdot \vec{v}_i(t) dt - \int_{t_0}^t \vec{\tau}_i(t) \cdot \vec{\omega}_i(t) dt \quad (84)$$

The difference pieces are called respectively: kinetic energy, rotational kinetic energy, potential energy, and rotational potential energy. They measure respectively: how fast the body moves, how fast it spins, how much work was done by the forces acting upon it, and how much work was done by the torques acting upon it.  $t_0$  is a time that one can start arbitrarily and by convention.

## 11 Collisions

Let's first consider particles instead of rigid bodies.

Given the velocities of two bodies  $\vec{v}_A$  and  $\vec{v}_B$  before a collision we want to determine the velocities of the bodies after the collision,  $\vec{v}'_A$  and  $\vec{v}'_B$ . The closing velocity is defined as:

$$\vec{v}_{closing} \equiv \vec{v}_B - \vec{v}_A \quad (85)$$

The separating velocity (relative velocity after the collision) is defined as:

$$\vec{v}_{separating} \equiv \vec{v}'_B - \vec{v}'_A \quad (86)$$

The separating velocity can be written in terms of the closing velocity:

$$\vec{v}_{separating} = -c\vec{v}_{closing} \quad (87)$$

where  $c$  is a restitution coefficient. For an elastic collision  $c = 1$  and for an inelastic collision  $c = 0$ . Conservation of momentum and the above relation between closing and separating velocity yields:

$$\vec{K}'_A = \vec{K}_A + \vec{J} \quad (88)$$

$$\vec{K}'_B = \vec{K}_B - \vec{J} \quad (89)$$

where

$$\vec{J} = \frac{(c+1)\vec{v}_{closing} \cdot \hat{n}}{1/m_A + 1/m_B} \hat{n} \quad (90)$$

In the case of collision of the object A with a plane B or other object of infinite mass

$$\vec{K}'_A = \vec{K}_A + \lim_{m_A \rightarrow \infty} + \vec{J} = \vec{K}_A + m_A(c+1)(\vec{v}_{closing} \cdot \hat{n})\hat{n} \quad (91)$$

When the  $v_{closing}$  is in the same direction as  $n$  then:

$$\vec{K}'_A = \vec{K}_A + m_A(c+1)\vec{v}_{closing} \quad (92)$$

In the case of rigid bodies eq. 87 applies to the points of contact between two rigid bodies. The collision is completed determined by the collision point  $\vec{q}$  and the normal to the collision plane  $\hat{n}$ .

To properly consider its effect on the entire body we have to rewrite the velocity in terms of the velocities of the center of mass plus a component due to the angular velocity.

$$\vec{J}_\perp = \frac{(c+1)\vec{v}_{closing} \cdot \hat{n}}{1/m_A + 1/m_B + [(1/I_A)(\vec{r}_{cA} \times \hat{n}) \times r_{cA} + (1/I_B)(\vec{r}_{cB} \times \hat{n}) \times r_{rB}] \cdot \hat{n}} \hat{n} \quad (93)$$

Here  $\vec{r}_{cA} = \vec{q} - \vec{p}_A$  is the point of collision respect to the center of mass  $\vec{p}_A$ ,  $\vec{v}_{cA} = \vec{\omega}_A \times \vec{r}_A + \vec{v}_A$  is the velocity of the collision point before the collision,  $m_A$

is the mass of body A,  $I_A$  is its moment of inertia.  $\hat{n}$  is a versor orthogonal to the contact point. Similarly,  $\vec{r}_{cB} = \vec{q} - \vec{p}_B$  and  $\vec{v}_{cB} = \vec{\omega}_B \times \vec{r}_B + \vec{v}_B$ .

Once this impulse is computed we can deal with the it for each body by adding the contribution of the impulse to force and torque.

In presence of friction at the contact point, there is also a tangential impulse to take into account:

$$\vec{J}_{\parallel} = \frac{(c_f + 1)\vec{v}_{closing} \cdot \hat{t}}{1/m_A + 1/m_B + [(1/I_A)(\vec{r}_{cA} \times \hat{t}) \times r_{cA} + (1/I_B)(\vec{r}_{cB} \times \hat{t}) \times r_{rB}]} \cdot \hat{t} \quad (94)$$

where  $\hat{t}$  is a versor orthogonal to  $n$  and to the relative velocity beween the colliding points:

$$\hat{t} = \frac{\hat{n} \times \vec{v}_{closing}}{|\hat{n} \times \vec{v}_{closing}|} \quad (95)$$

In general  $c_f$  is different from  $c$ .

Given the total impulse  $\vec{J} = \vec{J}_{\perp} + \vec{J}_{\parallel}$ , we can resolve the collisiong by correcting the momenta and angular momenta:

$$\vec{K}'_A = K_A + \vec{J} \quad (96)$$

$$\vec{\tau}'_A = \tau_A + \vec{r}_{cA} \times \vec{J} \quad (97)$$

$$\vec{K}'_B = K_B - \vec{J} \quad (98)$$

$$\vec{\tau}'_B = \tau_B - \vec{r}_{cB} \times \vec{J} \quad (99)$$

For if two objects we do the following:

- find the point of contact and compute  $r_{cA}, r_{cB}$
- find the compenetration and shift the objects back to cancel compenetration
- compute the impulse (eq. 94)
- update the force and torque to include one time contribution of the impulse (eq. 99).

## 12 Geometry of collision

Here we are concerned with the problem of terminating if two objects have collided. If the objects are made of vertices and faces the possible collisions can be vertex-vertex, vertex-edge, vertex-face, edge-face, face-face. Most of them can be ignored but vertex-face and edge-edge.

Let's focus on vertex-face collision. In order to resolve the collisions we need to:

- detect a collision has occurred;
- find the collision point,  $\vec{q}$ ;
- find the normal to the collision plane,  $\hat{n}$ ;
- resolve as in previous section.

We will assume the body is convex (the center of mass is inside the body).

We will label  $\vec{p}_A$  the center of mass of body  $A$ ,  $\vec{p}_B$  the center of mass of body  $B$ ,  $\vec{q}$  the collision point,  $\hat{n}$  the normal to the collision point,  $\vec{f}_1, \vec{f}_2, \vec{f}_3$  the vertices of the face of  $A$  and  $\vec{u}$  the vertex of body  $B$  which may have hit the face of object  $A$ .

First we compute  $\vec{n}$ :

$$\vec{n} = (\vec{f}_2 - \vec{f}_1) \times (\vec{f}_3 - \vec{f}_1) \quad (100)$$

and normalized it  $\hat{n} = \vec{n}/n$ . Than we compute  $q$ :

$$\vec{q} = \vec{u} - (\vec{u} \cdot \hat{n} - \vec{f}_1 \cdot \hat{n})\hat{n} \quad (101)$$

And we check that  $\vec{q}$  lays inside the face:

$$|(\vec{q} - \vec{f}_1) \times (\vec{f}_2 - \vec{f}_1)| + \quad (102)$$

$$|(\vec{q} - \vec{f}_2) \times (\vec{f}_3 - \vec{f}_2)| + \quad (103)$$

$$|(\vec{q} - \vec{f}_3) \times (\vec{f}_1 - \vec{f}_3)| \leq |(\vec{f}_1 - \vec{f}_2) \times (\vec{f}_3 - \vec{f}_1)| \quad (104)$$

If this is the case, we check the the projection along  $\hat{n}$  of the points  $\hat{p}_A$  (the center of mass of  $A$ ), the vertex  $\hat{v}$ , the collision point  $\vec{q}$  and  $\vec{p}_B$  (the center of mass of  $B$ ) and in proper spatial order:

$$((\vec{p}_A - \vec{p}_B) \cdot \hat{n})((\vec{u} - \vec{q}) \cdot \hat{n}) \geq 0 \quad (105)$$

If the collision occurs the penetration is  $\vec{p} - \vec{q}$ .

Finally we loop over all faces of  $A$ , and vertices of  $B$ , we run the above algorithm and if a collision occurs we resolve the penetration, we compute the velocity of the collision point and we apply the impulse at collision point as discussed in the previous section.

## 13 Implementation

```

1 // Program name: cylon.cpp
2 // Author: Massimo Di Pierro
3 // License: BSD

```

Import the necessary libraries:

```

1 #include <iostream>
2 #include <sstream>
3 #include <string>
4 #include <iostream>
5 #include "math.h"
6 #include "stdlib.h"
7 #include "vector"
8 #include "set"
9 #if defined(_MSC_VER)
10 #include <gl/glut.h>
11 #else
12 #include <GLUT/glut.h>
13 #endif
14 using namespace std;

```

Define useful macros and constants:

```

1 #define self (*this)
2 #define array vector // to avoid name collisions
3 #define forXYZ(i) for(int i=0; i<3; i++)
4 #define forEach(i,s) for(i=(s).begin();i!=(s).end();i++)
5 #define OBJ(iterator) (*(*iterator))
6 const float Pi = 3.1415926535897931;
7 const float PRECISION = 0.00001;
8 const int X = 0;
9 const int Y = 1;
10 const int Z = 2;
11 const float DT = 0.017f; // Hard-coded dt for Universe::evolve()
12 const int MSPF = 17; // msec per frame: glutTimerFunc() only takes integers
                      // for params
13                         // lower value = higher frame rate
14
15
16 float uniform(float a=0, float b=1) {
17     int n = 10000;
18     return a+(b-a)*(float)(rand() % n)/n;
19 }

```

Define class vector:

```

1 class Vector {
2 public:
3     float v[3];
4     Vector(float x=0, float y=0, float z=0) {
5         v[X] = x; v[Y] = y; v[Z] = z;
6     }
7     float operator()(int i) const { return v[i]; }
8     float &operator()(int i) { return v[i]; }
9 };

```

Define operations between vectors:

```

1 Vector operator*(float c, const Vector &v) {
2     return Vector(c*v(X),c*v(Y),c*v(Z));
3 }
4 Vector operator*(const Vector &v, float c) {
5     return c*v;
6 }
7 Vector operator/(const Vector &v, float c) {
8     return (1.0/c)*v;
9 }
10 Vector operator+(const Vector &v, const Vector &w) {
11     return Vector(v(X)+w(X),v(Y)+w(Y),v(Z)+w(Z));
12 }
13 Vector operator-(const Vector &v, const Vector &w) {
14     return Vector(v(X)-w(X),v(Y)-w(Y),v(Z)-w(Z));
15 }
16 float operator*(const Vector &v, const Vector &w) {
17     return v(X)*w(X) + v(Y)*w(Y) + v(Z)*w(Z);
18 }
19 float norm(const Vector &v) {
20     return sqrt(v*v);
21 }
22 Vector versor(const Vector &v) {
23     float d = norm(v);
24     return (d>0)?(v/d):v;
25 }
26 Vector cross(const Vector &v, const Vector &w) {
27     return Vector(v(Y)*w(Z)-v(Z)*w(Y),
28                   v(Z)*w(X)-v(X)*w(Z),
29                   v(X)*w(Y)-v(Y)*w(X));
30 }

```

Class matrix is a base for rotation and inertiatensor:

```

1 class Matrix {
2 public:
3     float m[3][3];
4     Matrix() { forXYZ(i) forXYZ(j) m[i][j] = 0; }
5     const float operator()(int i, int j) const { return m[i][j]; }
6     float &operator()(int i, int j) { return m[i][j]; }
7     Matrix t();
8 };

```

Operations between matrices:

```

1 Vector operator*(const Matrix &R, const Vector &v) {
2     return Vector(R(X,X)*v(X)+R(X,Y)*v(Y)+R(X,Z)*v(Z),
3                   R(Y,X)*v(X)+R(Y,Y)*v(Y)+R(Y,Z)*v(Z),
4                   R(Z,X)*v(X)+R(Z,Y)*v(Y)+R(Z,Z)*v(Z));
5 }
6
7 Matrix operator*(const Matrix &R, const Matrix &S) {
8     Matrix T;
9     forXYZ(i) forXYZ(j) forXYZ(k) T(i,j) += R(i,k)*S(k,j);
10    return T;
11 }
12
13 float det(const Matrix &R) {
14     return R(X,X)*(R(Z,Z)*R(Y,Y)-R(Z,Y)*R(Y,Z));

```

```

15     -R(Y,X)*(R(Z,Z)*R(X,Y)-R(Z,Y)*R(X,Z))
16     +R(Z,X)*(R(Y,Z)*R(X,Y)-R(Y,Y)*R(X,Z));
17 }
18
19 Matrix operator/(float c, const Matrix &R) {
20     Matrix T;
21     float d = c/det(R);
22     T(X,X) = (R(Z,Z)*R(Y,Y)-R(Z,Y)*R(Y,Z))*d;
23     T(X,Y) = (R(Z,Y)*R(X,Z)-R(Z,Z)*R(X,Y))*d;
24     T(X,Z) = (R(Y,Z)*R(X,Y)-R(Y,Y)*R(X,Z))*d;
25     T(Y,X) = (R(Z,X)*R(Y,Z)-R(Z,Z)*R(Y,X))*d;
26     T(Y,Y) = (R(Z,Z)*R(X,X)-R(Z,X)*R(X,Z))*d;
27     T(Y,Z) = (R(Y,X)*R(X,Z)-R(Y,Z)*R(X,X))*d;
28     T(Z,X) = (R(Z,Y)*R(Y,X)-R(Z,X)*R(Y,Y))*d;
29     T(Z,Y) = (R(Z,Y)*R(X,Y)-R(Z,Y)*R(X,X))*d;
30     T(Z,Z) = (R(Y,Y)*R(X,X)-R(Y,X)*R(X,Y))*d;
31     return T;
32 }
33
34 Matrix Matrix::t() {
35     Matrix Mt;
36     forXYZ(j) forXYZ(k) Mt(j,k)=self(k,j);
37     return Mt;
38 }

```

Class rotation is a matrix:

```

1 class Rotation : public Matrix {
2 public:
3     Rotation() { forXYZ(i) forXYZ(j) m[i][j] = (i==j)?1:0; }
4     Rotation(const Vector& v) {
5         float theta = norm(v);
6         if(theta<PRECISION) {
7             forXYZ(i) forXYZ(j) m[i][j] = (i==j)?1:0;
8         } else {
9             float s = sin(theta), c = cos(theta);
10            float t = 1-c;
11            float x = v(X)/theta, y = v(Y)/theta, z = v(Z)/theta;
12            m[X][X] = t*x*x+c;   m[X][Y] = t*x*y-s*z;   m[X][Z] = t*x*z+s*y;
13            m[Y][X] = t*x*y+s*z; m[Y][Y] = t*y*y+c;   m[Y][Z] = t*y*z-s*x;
14            m[Z][X] = t*x*z-s*y; m[Z][Y] = t*y*z+s*x; m[Z][Z] = t*z*z+c;
15        }
16    }
17 }

```

Class inertiatensor is also a matrix:

```

1 class InertiaTensor : public Matrix {};
2 InertiaTensor operator+(const InertiaTensor &a, const InertiaTensor &b) {
3     InertiaTensor c = a;
4     forXYZ(i) forXYZ(j) c(i,j) += b(i,j);
5     return c;
6 }

```

Class body (describes a rigid body object):

```

1 class Body {
2 public:

```

```

3 // object shape
4 float radius;
5 array<Vector> r; // vertices in local coordinates
6 array<array<int>> faces;
7 Vector color; // color of the object;
8 bool visible;
9 // properties of the body ///////////////////////////////
10 bool locked; // if set true, don't integrate
11 float m; // mass
12 InertiaTensor I; // moments of inertia (3x3 matrix)
13 // state of the body ///////////////////////////////
14 Vector p; // position of the center of mass
15 Vector K; // momentum
16 Matrix R; // orientation
17 Vector L; // angular momentum
18 // auxiliary variables ///////////////////////////////
19 Matrix inv_I; // 1/I
20 Vector F;
21 Vector tau;
22 Vector v; // velocity
23 Vector omega; // angular velocity
24 array<Vector> Rr; // rotated r's.
25 array<Vector> vertices; // rotated and shifted r's
26 // forces and constraints
27 // ...
28 Body(float m=1.0, float radius=0.2, bool locked=false) {
29     this->m = m;
30     this->radius = radius;
31     this->locked = locked;
32     this->R = Rotation();
33     I(X,X)=I(Y,Y)=I(Z,Z)=m;
34     this->inv_I = 1.0/I;
35     this->r.push_back(Vector(0,0,0)); // object contains one point
36     this->color = Vector(1,0,0);
37     this->visible = true;
38     this->update_vertices();
39 }
40 void clear() { F(X)=F(Y)=F(Z)=tau(X)=tau(Y)=tau(Z)=0; }
41 void update_vertices();
42 void integrator(float dt);
43 void loadObj(const string & file, float scale);
44 void draw();
45 };

```

Rotate and shift all vertices from local to universe:

```

1 void Body::update_vertices() {
2     if(Rr.size()!=r.size()) Rr.resize(r.size());
3     if(vertices.size()!=r.size()) vertices.resize(r.size());
4     for(int i=0; i<r.size(); i++) {
5         Rr[i] = R*r[i];
6         vertices[i]=Rr[i]+p;
7     }
8 }

```

Euler integrator:

```

1 void Body::integrator(float dt) {

```

```

2     v      = (1.0/m)*K;
3     omega = R*(inv_I*(R.t()*L));      // R*inv_I*R.t() in rotated frame
4     p      = p + v*dt;                // shift
5     K      = K + F*dt;                // push
6     R      = Rotation(omega*dt)*R;    // rotate
7     L      = L + tau*dt;              // spin
8     update_vertices();
9 }

```

Interface for all forces. the constructor can be specific of the force. the apply methods adds the contribution to f and tau:

```

1 class Force {
2 public:
3     virtual void apply(float dt)=0;
4     virtual void draw() {};
5 };

```

Gravity is a force:

```

1 class GravityForce : public Force {
2 public:
3     Body *body;
4     float g;
5     GravityForce(Body *body, float g = 0.01) {
6         this->body = body; this->g = g;
7     }
8     void apply(float dt) {
9         body->F(Y) -= (body->m)*g;
10    }
11 };

```

Spring is a force:

```

1 class SpringForce : public Force {
2 public:
3     Body *bodyA;
4     Body *bodyB;
5     int iA, iB;
6     float kappa, L;
7     SpringForce(Body *bodyA, int iA, Body *bodyB, int iB,
8                  float kappa, float L) {
9         this->bodyA = bodyA; this->iA = iA;
10        this->bodyB = bodyB; this->iB = iB;
11        this->kappa = kappa; this->L = L;
12    }
13    void apply(float dt) {
14        Vector d = ((iB<0)?(bodyB->p):(bodyB->vertices[iB]))-
15            ((iA<0)?(bodyA->p):(bodyA->vertices[iA]));
16        float n = norm(d);
17        if(n>PRECISION) {
18            Vector F = kappa*(n-L)*(d/n);
19            bodyA->F = bodyA->F+F;
20            bodyB->F = bodyB->F-F;
21            if(iA>=0)
22                bodyA->tau = bodyA->tau + cross(bodyA->Rr[iA],F);
23            if(iB>=0)
24                bodyB->tau = bodyB->tau - cross(bodyB->Rr[iB],F);

```

```

25     }
26 }
27 void draw();
28 };

```

A spring can be anchored to a pin:

```

1 class AnchoredSpringForce : public Force {
2 public:
3     Body *body;
4     int i;
5     Vector pin;
6     float kappa, L;
7     AnchoredSpringForce(Body *body, int i, Vector pin,
8                           float kappa, float L) {
9         this->body = body; this->i = i;
10        this->pin = pin;
11        this->kappa = kappa; this->L = L;
12    }
13    void apply(float dt) {
14        Vector d = pin - ((i<0)?(body->p):(body->vertices[i]));
15        float n = norm(d);
16        Vector F = kappa*(n-L)*d/n;
17        body->F = body->F+F;
18        if(i>=0)
19            body->tau = body->tau + cross(body->Rr[i],F);
20    }
21 };

```

Friction is also a force (this ignores the shape of the body, assumes a sphere):

```

1 class FrictionForce: public Force {
2 public:
3     Body *body;
4     float gamma;
5     FrictionForce(Body *body, float gamma) {
6         this->body = body; this->gamma = gamma;
7     }
8     void apply(float dt) {
9         body->F = body->F-gamma*(body->v)*dt; // ignores shape
10    }
11 };

```

Class water, one instance floods the universe:

```

1 class Water: public Force {
2 public:
3     float level, wave, speed;
4     float m[41][41];
5     float t, x0,dx;
6     set<Body*> *bodies;
7     Water(set<Body*> *bodies, float level,
8           float wave=0.2, float speed=0.2) {
9         this->bodies = bodies;
10        this->level = level; this->wave = wave, this->speed = speed;
11        t = 0; x0 = 5.0; dx = 0.25;
12    }
13    void apply(float dt) {

```

```

14     set<Body*>::iterator ibody;
15     t = t+dt;
16     for(int i=0; i<41; i++)
17       for(int j=0; j<41; j++)
18         this->m[i][j] = level+wave/2*sin(speed*t+i)+wave/2*sin(0.5*i*j);
19     forEach(ibody,*bodies) {
20       Body &body = OBJ(ibody); // dereference
21       int i = (body.p(X)+x0)/dx;
22       int j = (body.p(Z)+x0)/dx;
23       if(body.p(Y)<m[i][j]) {
24         body.F = body.F + (Vector(0,1,0)-2.0*(body.v))*dt;
25         body.L = (1.0-dt)*body.L;
26       }
27     }
28   }
29   void draw();
30 };
31
32 Vector resolve_collision(Body &A, Body &B, Vector q, Vector n,
33                           float c, float cf) {
34   Vector r_cA = q-A.p;
35   Vector r_cB = q-B.p;
36   Vector v_cA = cross(A.omega,r_cA)+A.v;
37   Vector v_cB = cross(B.omega,r_cB)+B.v;
38   Vector v_closing = v_cB-v_cA;
39   Vector t = versor(cross(n,v_closing));
40   Vector Jo,Jp,J;
41   Jo = (c+1)*(v_closing*n) /
42     (1.0/A.m+1.0/B.m+(A.inv_I*cross(cross(r_cA,n),r_cA) +
43     B.inv_I*cross(cross(r_cB,n),r_cB))*n)*n;
44   Jp = (cf+1)*(v_closing*t) /
45     (1.0/A.m+1.0/B.m+(A.inv_I*cross(cross(r_cA,t),r_cA) +
46     B.inv_I*cross(cross(r_cB,t),r_cB))*t)*t;
47   J = Jo+Jp;
48   A.K = A.K + J;
49   B.K = B.K - J;
50   A.L = A.L + cross(r_cA,J);
51   B.L = B.L - cross(r_cB,J);
52 }

```

A constraint has detect and resolve methods:

```

1 class Constraint {
2 public:
3   virtual bool detect()=0;
4   virtual void resolve(float dt)=0;
5   virtual void draw() {}
6 };

```

Class to deal with collision with static plane (ignore rotation):

```

1 class PlaneConstraint: public Constraint {
2 public:
3   Body *body;
4   Vector n,d;
5   float penetration, restitution;
6   // d is the distance of the plane from origin
7   // n is a versor orthogonal to plane

```

```

8   // (in opposite direction from collision)
9   PlaneConstraint(Body *body, float restitution,
10    const Vector &d, const Vector &n) {
11     this->body = body;
12     this->n = n; this->d = d;
13     this->restitution = restitution;
14   }
15   bool detect() {
16     penetration = body->p*n-d*n + body->radius;
17     return penetration>=0;
18   }
19   void resolve(float dt) {
20     // move the object back if stuck on plane
21     body->p = body->p - penetration*n;
22     float K_ortho = n*body->K;
23     // optional, deal with friction
24     Vector L_ortho = -(body->radius)*cross(n, body->K-K_ortho);
25     body->L = (n*body->L)*n + L_ortho;
26     // reverse momentum
27     if(K_ortho>0)
28       body->K = body->K - (restitution+1)*(K_ortho)*n;
29   }
30 };

```

Default all-2-all collision handler:

```

1 class All2AllCollisions : public Constraint {
2 public:
3   float restitution;
4   set<Body*> *bodies;
5   All2AllCollisions(set<Body*> *bodies, float c=0.5) {
6     this->bodies=bodies;
7     restitution = c;
8   }
9   bool detect() { return true; }
10  void resolve(float dt) {
11    set<Body*>::iterator ibodyA, ibodyB;
12    forEach(ibodyA,*bodies) {
13      forEach(ibodyB,*bodies) {
14        if((*ibodyA)<(*ibodyB)) {
15          Body &A = OBJ(ibodyA); // dereference
16          Body &B = OBJ(ibodyB); // dereference
17          Vector d = B.p-A.p;
18          Vector n = d/norm(d);
19          Vector v_closing = B.v-A.v;
20          float penetration = A.radius+B.radius-norm(d);
21          if(penetration>0 && v_closing*n<0) {
22            // move bodies to eliminate penetration
23            Vector delta = (penetration/(A.m+B.m))*versor(d);
24            A.p = A.p-B.m*delta;
25            B.p = B.p+A.m*delta;
26            // compute contact point
27            Vector n = versor(B.p-A.p);
28            Vector q = A.p + A.radius*n;
29            resolve_collision(A,B,q,n,restitution,restitution);
30        }
31      }
32    }

```

```

33     }
34 }
35 };

```

A universe stores bodies, forces, constraints and evolves in time:

```

1 class Universe {
2 public:
3     float dt;
4     // universe state
5     set<Body*> bodies;
6     set<Force*> forces;
7     set<Constraint*> constraints;
8     // useful iterators
9     set<Body*>::iterator ibody;
10    set<Force*>::iterator iforce;
11    set<Constraint*>::iterator iconstraint;
12    int frame;
13    Universe() {
14        frame = 0;
15    }
16    ~Universe() {
17        forEach(ibody,bodies) delete (*ibody);
18        forEach(iforce,forces) delete (*iforce);
19        forEach(iconstraint,constraints) delete (*iconstraint);
20    }
21    // evolve universe
22    void evolve() {
23        // clear forces and torques
24        forEach(ibody,bodies)
25            OBJ(ibody).clear();
26        // compute forces and torques
27        forEach(iforce,forces)
28            OBJ(iforce).apply(dt);
29        callback();
30        // integrate
31        forEach(ibody,bodies)
32            if(!OBJ(ibody).locked)
33                OBJ(ibody).integrator(dt);
34        // handle collisions (not quite right yet)
35        forEach(iconstraint,constraints)
36            if(OBJ(iconstraint).detect())
37                OBJ(iconstraint).resolve(dt);
38        frame++;
39    }
40 public:
41     virtual void build_universe() { bodies.insert(new Body()); };
42     virtual void callback() {};
43     virtual void mouse(int button, int state, int x, int y) {};
44     virtual void keyboard(unsigned char key, int x, int y) {};
45 };

```

Auxiliary functions translate moments of inertia:

```

1 InertiaTensor dI(float m, const Vector &r) {
2     InertiaTensor I;
3     float r2 = r*r;
4     forXYZ(j) forXYZ(k) I(j,k) = m*((j==k)?r2:0)-r(j)*r(k));

```

```

5     return I;
6 }

```

Auxiliary function to include to merge two bodies needs some more work...:

```

1 Body operator+(const Body &a, const Body &b) {
2     Body c;
3
4     c.color = 0.5*(a.color+b.color);
5     c.radius = max(a.radius+norm(a.p-c.p),b.radius+norm(b.p-c.p));
6     c.R = Rotation();
7     c.m = (a.m+b.m);
8     c.p = (a.m*a.p + b.m*b.p)/c.m;
9     c.K = a.K+b.K;
10    c.L = a.L+cross(a.p-c.p,a.K)+b.L+cross(b.p-c.p,b.K);
11    Vector da = a.p-c.p;
12    Vector db = b.p-c.p;
13    int na = a.r.size();
14    int nb = b.r.size();
15    c.I(X,X) = c.I(Y,Y)= c.I(Z,Z) =0;
16    // copy all r
17    Vector v;
18    c.r.resize(na+nb);
19    for(int i=0; i<na; i++) {
20        v = a.R*a.r[i]+da;
21        c.r[i] = v;
22        c.I = c.I + dI(a.m/na,v);
23    }
24    for(int i=0; i<nb; i++) {
25        v = b.R*b.r[i]+db;
26        c.r[i+na] = v;
27        c.I = c.I + dI(b.m/nb,v);
28    }
29    // copy all faces and re-label r
30    int m = a.faces.size();
31    c.faces.resize(a.faces.size()+b.faces.size());
32    for(int j=0; j<a.faces.size(); j++)
33        c.faces[j]=a.faces[j];
34    for(int j=0; j<b.faces.size(); j++)
35        for(int k=0; k<b.faces[j].size(); k++)
36            c.faces[j+m].push_back(b.faces[j][k]+na);
37    c.inv_I = 1.0/c.I;
38    c.update_vertices();
39    return c;
40 }

```

Function that loads an wavefront obj file into a body:

```

1 void Body::loadObj(const string & file, float scale=0.5) {
2     ifstream input;
3     string line;
4     float x,y,z;
5     r.resize(0);
6     faces.resize(0);
7     input.open(file.c_str());
8     if(input.is_open()) {
9         while(input.good()) {
10             getline(input, line);

```

```

11     if(line.length()>0) {
12         string initialVal;
13         istringstream instream;
14         instream.str(line);
15         instream >> initialVal;
16         if(initialVal=="v") {
17             instream >> x >> y >> z;
18             Vector p = scale*Vector(x,y,z);
19             r.push_back(p);
20             radius = max(radius,norm(p));
21         } else if (initialVal=="f") {
22             array<int> path;
23             while(instream >> x) path.push_back(x-1);
24             faces.push_back(path);
25         }
26     }
27 }
28 update_vertices();
29 }
30 }
```

Make a universe with an airplane:

```

1 class MyUniverseAirplane : public Universe {
2     Body plane;
3 public:
4     void build_universe() {
5         plane.color=Vector(1,0,0); //red
6         plane.loadObj("assets/plane.obj",0.5);
7         plane.R = Rotation(Vector(0,Pi,0));
8         bodies.insert(&plane);
9         forces.insert(new GravityForce(&plane,0.5));
10        // constraints.insert(new PlaneConstraint(&plane,0.0,Vector(0,-0.2,0),
11                                Vector(0,-1,0)));
12    }
13    void callback() {
14    }
15    void keyboard(unsigned char key, int x, int y) {
16        if(key=='w') plane.L = plane.L + Vector(+0.1,0,0);
17        if(key=='a') plane.L = plane.L + Vector(0,0,+0.1);
18        if(key=='d') plane.L = plane.L + Vector(0,0,-0.1);
19        if(key=='z') plane.L = plane.L + Vector(-0.1,0,0);
20        if(key=='n') plane.K = plane.K + Vector(0,0,+1);
21        if(key=='m') plane.K = plane.K + Vector(0,0,-1);
22    };
23 }
24
25 array<float> random_vector(int n, float M) {
26     float one_norm = 0.0;
27     array<float> m(n);
28     for(int i=0; i<n; i++) { m[i] = uniform(); one_norm += m[i]; }
29     for(int i=0; i<n; i++) m[i]=M*m[i]/one_norm;
30 }
31 }
```

Make a universe with an exploding body:

```

1 class SimpleUniverse : public Universe {
2 private:
3     Body* xb;
4 public:
5     void build_universe() {
6         float m = 1.0;
7         float v = 5.0;
8         float theta = Pi/2.2;
9         Body &b = *new Body(m);
10        b.color = Vector(uniform(),uniform(),uniform()); //red
11        b.loadObj("assets/sphere.obj",0.5);
12        b.p = Vector(0,10,-10);
13        b.K = Vector(0,0,0);
14        b.L = Vector(0,0,0);
15        bodies.insert(&b);
16        forces.insert(new GravityForce(&b,1.0));
17        xb = &b;
18    }
19    void callback() {
20        if(frame==150) {
21            int n = 50;
22            float vx,vy,vz;
23            float E = 0.5;
24            float mysum_x=0, mysum_y=0, mysum_z=0;
25            array<float> m = random_vector(n,xb->m);
26            array<float> eps_x(n), eps_y(n), eps_z(n);
27            xb->visible = false;
28            for(int j=0; j<n; j++) {
29                cout << j << " " << uniform() << endl;
30                Body &a = *new Body(m[j]);
31                a.color = Vector(uniform(),uniform(),uniform());
32                a.p = Vector(xb->p(X), xb->p(Y), xb->p(Z));
33                if(j<n-1) {
34                    eps_x[j] = uniform(-1,1)*sqrt(E);
35                    eps_y[j] = uniform(-1,1)*sqrt(E);
36                    eps_z[j] = uniform(-1,1)*sqrt(E);
37                    mysum_x += m[j]*eps_x[j];
38                    mysum_y += m[j]*eps_y[j];
39                    mysum_z += m[j]*eps_z[j];
40                } else {
41                    eps_x[j] = -mysum_x/m[j];
42                    eps_y[j] = -mysum_y/m[j];
43                    eps_z[j] = -mysum_z/m[j];
44                }
45                vx = xb->v(X) + eps_x[j];
46                vy = xb->v(Y) + eps_y[j];
47                vz = xb->v(Z) + eps_z[j];
48                a.K = Vector(m[j]*vx, m[j]*vy, m[j]*vz);
49                bodies.insert(&a);
50                forces.insert(new GravityForce(&a,1.0));
51            }
52        }
53        float restitution = 0.5;
54        forEach(ibody,bodies)
55        if(OBJ(ibody).p(Y)<0 && OBJ(ibody).K(Y)<0) {

```

```

58     OBJ(ibody).p(Y)=-restitution*OBJ(ibody).p(Y);
59     OBJ(ibody).K(Y)=-restitution*OBJ(ibody).K(Y);
60 }
61 }
62 };

```

Make a universe with some bouncing balls:

```

1 class MyUniverse : public Universe {
2 public:
3     void build_universe() {
4         Body *b_old = 0;
5         for(int i=0; i<30; i++) {
6             Body &b = *new Body();
7             b.color=Vector(uniform(0,1),uniform(0,1),uniform(0,1));
8             b.loadObj("assets/sphere.obj");
9             b.p = Vector(uniform(-2,2),uniform(1,3),uniform(-2,2));
10            b.K = Vector(uniform(-2,2),uniform(-2,2),0);
11            b.L = Vector(uniform(0,1),uniform(0,1),uniform(0,1));
12            bodies.insert(&b);
13            forces.insert(new GravityForce(&b,1));
14            constraints.insert(new PlaneConstraint(&b,0.9,Vector(0,0,0),
15                                Vector(0,-1,0)));
16            constraints.insert(new PlaneConstraint(&b,0.9,Vector(0,5,0),
17                                Vector(0,+1,0)));
18            constraints.insert(new PlaneConstraint(&b,0.9,Vector(5,0,0),
19                                Vector(1,0,0)));
20            constraints.insert(new PlaneConstraint(&b,0.9,Vector(-5,0,0),
21                                Vector(-1,0,0)));
22            constraints.insert(new PlaneConstraint(&b,0.9,Vector(0,0,5),
23                                Vector(0,0,1)));
24            constraints.insert(new PlaneConstraint(&b,0.9,Vector(0,0,-5),
25                                Vector(0,0,-1)));
26        }
27        constraints.insert(new All2AllCollisions(&bodies,0.8));
28        // forces.insert(new Water(&bodies,3.0));
29    }
30 };

```

Make a universe with composite objects made of cubes:

```

1 class MyUniverseCubes : public Universe {
2 public:
3     Body cube(const Vector &p, const Vector& theta, const Vector &color) {
4         Body cube;
5         cube.color = color;
6         cube.p = p;
7         cube.R = Rotation(theta);
8         for(int x=-1; x<=+1; x+=2)
9             for(int y=-1; y<=+1; y+=2)
10                 for(int z=-1; z<=+1; z+=2)
11                     cube.r.push_back(Vector(x,y,z));
12         cube.faces.resize(6);
13         for(int i=0; i<2; i++) {
14             cube.faces[i].push_back(0+4*i);
15             cube.faces[i].push_back(1+4*i);
16             cube.faces[i].push_back(3+4*i);
17             cube.faces[i].push_back(2+4*i);

```

```

18     cube.faces[i+2].push_back(0+2*i);
19     cube.faces[i+2].push_back(1+2*i);
20     cube.faces[i+2].push_back(5+2*i);
21     cube.faces[i+2].push_back(4+2*i);
22     cube.faces[i+4].push_back(0+i);
23     cube.faces[i+4].push_back(2+i);
24     cube.faces[i+4].push_back(6+i);
25     cube.faces[i+4].push_back(4+i);
26 }
27 cube.update_vertices();
28 return cube;
29 }
30 void build_universe() {
31     Body *thing = new Body();
32     (*thing) =
33         cube(Vector(0,4,0),Vector(0,1,0),Vector(0,1,0)) +
34         cube(Vector(1,4,0),Vector(1,0,0),Vector(0,1,0.2)) +
35         cube(Vector(1,4.5,.5),Vector(1,0,1),Vector(0.2,1));
36     (*thing).L = Vector(0,0.1,0);
37     bodies.insert(thing);
38     Body *other_thing = new Body();
39     (*other_thing) = (*thing);
40     (*other_thing).p = Vector(-1,3,0);
41     (*other_thing).L = Vector(-0.1,0.0,0.1);
42     bodies.insert(other_thing);
43 }
44 };
45
46 class CompositionUniverse: public Universe {
47 private:
48     Body b;
49 public:
50     void build_universe() {
51         Body a;
52         float x,y,z;
53         for(int i=0; i<8; i++) {
54             x = 2*((i>>0)&1)-1;
55             y = 2*((i>>1)&1)-1;
56             z = 2*((i>>2)&1)-1;
57             a.p = Vector(x,y+1,z-5);
58             if(i==0) b = a; else b=b+a;
59         }
60         b.L = Vector(5,10,0);
61         bodies.insert(&b);
62     }
63 };
64
65 MyUniverse universe;
// MyUniverseAirplane universe;
// MyUniverseCubes universe;
// SimpleUniverse universe;
// CompositionUniverse universe;

```

Glut code below creates a window in which to display the scene:

```

1 void createWindow(const char* title) {
2     int width = 640, height = 480;
3     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

```

```

4   glutInitWindowSize(width,height);
5   glutInitWindowPosition(0,0);
6   glutCreateWindow(title);
7   glClearColor(0.9f, 0.95f, 1.0f, 1.0f);
8   glEnable(GL_DEPTH_TEST);
9   glShadeModel(GL_SMOOTH);
10  glMatrixMode(GL_PROJECTION);
11  glLoadIdentity();
12  gluPerspective(60.0, (double)width/(double)height, 1.0, 500.0);
13  glMatrixMode(GL_MODELVIEW);
14 }

```

Called each frame to update the 3d scene. delegates to the application:

```

1 void update(int value) {
2   // evolve world
3   universe.dt = DT;
4   universe.evolve();
5   glutTimerFunc(MSPF, update, 0);
6   glutPostRedisplay();
7 }

```

Function called each frame to display the 3d scene. it draws all bodies, forces and constraints:

```

1 void display() {
2   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
3   glLoadIdentity();
4   gluLookAt(0.0,3.5,10.0, 0.0,3.5,0.0, 0.0,1.0,0.0);
5   forEach(universe.ibody,universe.bodies)
6     if(OBJ(universe.ibody).visible)
7       OBJ(universe.ibody).draw();
8   forEach(universe.iforce,universe.forces)
9     OBJ(universe.iforce).draw();
10  forEach(universe.iconstraint,universe.constraints)
11    OBJ(universe.iconstraint).draw();
12  // update the displayed content
13  glFlush();
14  glutSwapBuffers();
15 }

```

Code that draws an body:

```

1 void Body::draw() {
2   glPolygonMode(GL_FRONT,GL_FILL);
3   if(faces.size()==0) {
4     glColor3f(color(0),color(1),color(2));
5     int n = vertices.size();
6     for(int i=0; i<n; i++) {
7       glPushMatrix();
8       glTranslatef(vertices[i].v[X],vertices[i].v[Y],vertices[i].v[Z]);
9       glutSolidSphere(0.2,10,10);
10      glPopMatrix();
11    }
12  } else
13    for(int i=0; i<faces.size(); i++) {
14      float k = 0.5*(1.0+(float)(i+1)/faces.size());
15      glColor3f(color(0)*k,color(1)*k,color(2)*k);

```

```

16     glBegin(GL_POLYGON);
17     for(int j=0; j<faces[i].size(); j++)
18         glVertex3fv(vertices[faces[i][j]].v);
19     glVertex3fv(vertices[faces[i][0]].v);
20     glEnd();
21 }
22 }
```

Code that draws a spring:

```

1 void SpringForce::draw() {
2     glColor3f(0,0,0);
3     glPolygonMode(GL_FRONT_AND_BACK ,GL_LINE);
4     glBegin(GL_LINES);
5     if(iA<0)    glVertex3fv(bodyA->p.v);
6     else    glVertex3fv(bodyA->vertices[iA].v);
7     if(iB<0)    glVertex3fv(bodyB->p.v);
8     else    glVertex3fv(bodyB->vertices[iB].v);
9     glEnd();
10 }
```

Draw the water:

```

1 void Water::draw() {
2     glPolygonMode(GL_FRONT,GL_FILL);
3     for(int i=0; i<40; i++) {
4         glBegin(GL_POLYGON);
5         for(int j=0; j<40; j++) {
6             glColor3f(0,0,0.5+0.25*(m[i][j]-level+wave)/wave);
7             glVertex3fv(Vector(-x0+dx*i,m[i][j],-x0+dx*j).v);
8             glVertex3fv(Vector(-x0+dx*i,m[i][j+1],-x0+dx*j+dx).v);
9             glVertex3fv(Vector(-x0+dx*i+dx,m[i+1][j+1],-x0+dx*j+dx).v);
10            glVertex3fv(Vector(-x0+dx*i+dx,m[i+1][j],-x0+dx*j).v);
11        }
12        glEnd();
13    }
14 }
```

Function called when the display window changes size:

```

1 void reshape(int width, int height) {
2     glViewport(0, 0, width, height);
3 }
```

Function called when a mouse button is pressed:

```

1 void mouse(int button, int state, int x, int y) {
2     universe.mouse(button,state,x,y);
3 }
```

Function called when a key is pressed:

```

1 void keyboard(unsigned char key, int x, int y) {
2     universe.keyboard(key,x,y);
3 }
```

Called when the mouse is dragged:

```

1 void motion(int x, int y) { }
```

The main function. everythign starts here:

```
1 int main(int argc, char** argv) {
2     // Create the application and its window
3     glutInit(&argc, argv);
4     createWindow("Cylon");
5     // fill universe with stuff
6     universe.build_universe();
7     // Set up the appropriate handler functions
8     glutReshapeFunc(reshape);
9     glutKeyboardFunc(keyboard);
10    glutDisplayFunc(display);
11    glutMouseFunc(mouse);
12    glutMotionFunc(motion);
13    glutTimerFunc(MSPF, update, 0); // Do not refresh faster than target
14                                framerate
14    // Enter into a loop
15    glutMainLoop();
16    return 0;
17 }
```

## LEGEND



- Red = Specific software or terms
- Blue = Important steps or instructions

- Body text = 18 pt Arial
- Hyperlinks = 18pt Courier new

- Code is white on black with a green border
- Console input is green on black

- Titles = 32 pt Arial cyan (shadowed) + 20pt sub title

IT378 - M. Di Pierro @ DePaul CTI

**Topic:**



# Host Security



IT378 - M. Di Pierro @ DePaul CTI

## Tentative" Program Overview



**Week 1 - Introduction. General OS Security Features. Security Threats**

**Week 2 - Securing Windows / Implementing Policies / Protecting Files (Windows)**

**Week 3 - System recovery, Scanners and Auditing (Windows)**

**Week 4 - Intro to Linux. Installation. Basic Commands (UNIX/Linux)**

**Week 5 - Securing Linux (UNIX/Linux)**

**Week 6 - Apache with mod\_ssl and Samba (UNIX/Linux/Windows)**

**Week 7 - Email server and/or DNS server (UNIX/Linux)**

**Week 8 - Configuring a firewall under Linux (UNIX/Linux)**

**Week 9 - Encryption / VPN and ssh tunnels (UNIX/Linux/Windows)**

**Week 10 - Other issues, final review.**

IT378 - M. Di Pierro @ DePaul CTI

## Basic Tools Covered



### Encryption Tools:

ssl/ssh/putTY - secure socket layer and ssh tunnels

### Password Cracking Tools:

crack, John the Ripper - break password encryption

### Network Tools and Applications:

netstat, ActivePorts	- get info about open tcp/udp connections
tcpdump, Ethereal	- monitor network traffic
Cerberus	- security scanner
nmap, SuperScan	- port scanner
snort	- intrusion detection
hackman,hiew	- binary editor and disassembler
Back Officer	- simulates services and monitors for port scans
ZoneAlarm	- Firewall
Ad-Aware, Spybot, MSFT	- anti spyware
iptables	- firewall
Apache	- web server
Bind	- DNS server
....	- POP3/SMTP serve

**Has anyone used these tools?**

IT378 - M. Di Pierro @ DePaul CTI

## Introduction



IT378 - M. Di Pierro @ DePaul CTI

## Introduction / Definitions



**Vulnerability:** weakness in the security of a system

**Threat:** a set of circumstances that can cause harm

**Attack:** an intentional exploitation of a vulnerability

**Interception:** unauthorized party has control of an asset (information or resource)

**Interruption:** an asset (information or resource) becomes lost or unavailable

**Modification:** an asset (information or resource) is changed

**Fabrication:** new fake information or resources are created and made available

Confidentiality + Integrity + Availability = Security

IT378 - M. Di Pierro @ DePaul CTI

## Introduction / Vulnerabilities



### Vulnerabilities: Hardware :

- Interception or Theft
- Interruption or Damage
- Modification

### Vulnerability: Software:

- Interception (software is stolen)
- Interruption (software is deleted)
- Modification (modified for by a Virus/Trojan/Trapdoor/Logic Bomb)
- Fabrication (malicious software is inserted in the system)

### Vulnerability: Data:

- Interception (data is stolen)
- Interruption (data is not flowing as it should)
- Modification (data is modified without permission, example Salami Attack)
- Fabrication (fake data is inserted in the system)

IT378 - M. Di Pierro @ DePaul CTI

## Why Security ?



**Protecting Information and Resources**

**Ensuring Privacy**

**Facilitating Workflow**

**Software bugs**

**Compensating for human error or neglect**

Which users is authorized to access a certain resource (disk/file/etc)?

Which operations is user XYZ permitted to do with resource ABC?

IT378 - M. Di Pierro @ DePaul CTI

## Why Security ?



**Protecting Information and Resources**

**Ensuring Privacy**

**Facilitating Workflow**

**Software bugs**

**Compensating for human error or neglect**

What information should be considered public?

What should be accessed only by users of this computer?

What should be considered private of each user?

IT378 - M. Di Pierro @ DePaul CTI

## Why Security ?



**Protecting Information and Resources**

**Ensuring Privacy**

**Facilitating Workflow**

**Software bugs**

**Compensating for human error or neglect**

What are the steps required to authorize a certain operation?

What if multi users are involved?

Should root have access to everything?

How do we prevent it?

IT378 - M. Di Pierro @ DePaul CTI

## Why Security ?



**Protecting Information and Resources**

**Ensuring Privacy**

**Facilitating Workflow**

**Software bugs**

**Compensating for human error or neglect**

Can a software bug cause data loss in a separate program?

Can a software bug cause a computer crash?

How do we prevent it?

IT378 - M. Di Pierro @ DePaul CTI

## Why Security?



**Protecting Information and Resources**

**Ensuring Privacy**

**Facilitating Workflow**

**Software bugs**

**Compensating for human error or neglect**

What if the system administrator forgets to some system maintenance task?

What if a user by accident deletes a file?

What if you type  
yes | rm -r /\*  
?

IT378 - M. Di Pierro @ DePaul CTI

## External and Internal Threats



### External Threats:

- Unauthorized access to the system: stolen password, exploited backdoor, exploited buffer overflow, exploited other bug
- Service interruption: Denial of service attacks
- Viruses, worms, unwanted pop-ups, etc

### Internal Threats:

- User access other users' data (accidental read/write/delete or malicious use)
- Root error causes data loss (types wrong command)
- Software error causes data loss or data leak or service interruption

IT378 - M. Di Pierro @ DePaul CTI

## Introduction / Defense Systems



### Methods of Defense:

**Prevent it** by blocking attacks, closing vulnerabilities  
(security settings, good coding, antivirus, firewall)

**Deter it** by making the attack hard and not worthwhile  
(strong encryption, firewall configuration, security policies)

**Deflect it** by making another target more attractive  
(network configuration, decoy programs)

**Detect it** by monitoring your system/network  
(use port scans, monitor network traffic and check logs)

**Recover** from its effect  
(make regular backups and check backups and logs)

IT378 - M. Di Pierro @ DePaul CTI

Topic:

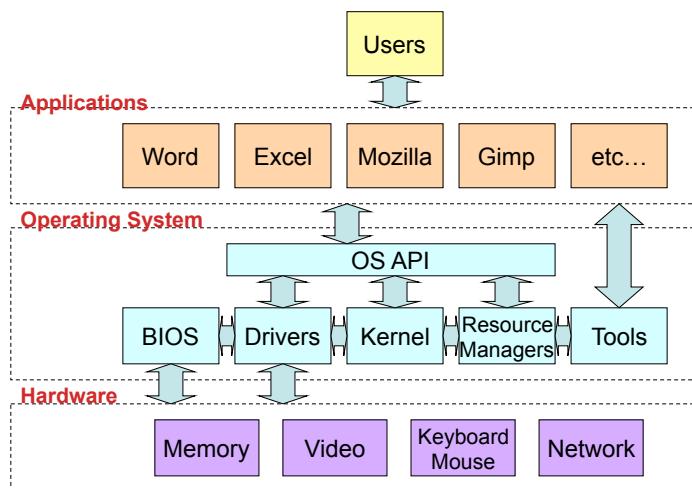


# Operating System Security



IT378 - M. Di Pierro @ DePaul CTI

What is an Operating System?



IT378 - M. Di Pierro @ DePaul CTI

## Windows Blue Screen of Death



The system is either busy or has become unstable. You can wait and see if it becomes available again, or you can restart your computer.

- \* Press any key to return to Windows and wait.
- \* Press CTRL+ALT+DEL again to restart your computer. You will lose unsaved information in any programs that are running.

Press any key to continue . . .

```
Call Trace:
[<>02857ba>] generic_make_request+0xca/0x140 [kernel]
[<>0285896>] submit_bh+0x6/0x140 [kernel]
[<>0285afe>] ll_rw_block+0x1e/0x1a0 [kernel]
[<>0202cf3>] bread+0x63/0xb0 [kernel]
[<>01eb2db>] bdev_read_page+0x1b/0x110 [kernel]
[<>01ea49>] __read_primary_suspend_image+0x49/0x570 [kernel]
[<>0206bbe>] set_blocksize+0x10e/0x110 [kernel]
[<>01efffe>] read_primary_suspend_image+0xde/0x130 [kernel]
[<>01ef5813>] software_resume+0xc3/0x100 [kernel]
[<>01bf24b>] init+0xb/0x160 [kernel]
[<>01c17e3>] arch_kernel_thread+0x23/0x30 [kernel]
[<>01bf240>] init+0x0/0x160 [kernel]

Code: Bad ELF value.
<0>Kernel panic: Attempted to kill init!
```

IT378 - M. Di Pierro @ DePaul CTI

## Operating System History



30 years ago there were no operating systems (think of Commodore 64). Each computer would only execute one program at the time.

Eventually “executive programs” would help assist individual programmers to switch from one user to another (relocation, linking) (think of MS-DOS)

Then parallel use of the CPU became more important (scheduling and sharing files) and multi programmed OS called “monitors program” appeared.

<b>Executives</b>	<b>Monitors</b>
<b>Passive (runs when called)</b>	<b>Active (runs in background)</b>
<b>Code relocation only</b>	<b>Protected memory</b>

IT378 - M. Di Pierro @ DePaul CTI

## Operating System Security



**Physical Separation:** different processes use different physical objects

**Temporal Separation:** different processes run at different time

**Logical Separation:** different processes operate under the illusion that no other processes exist (think Unix/Linux and Windows 2000/XP)

**Cryptographic Separation:** different processes conceal data to other processes. Think of encrypted file systems.

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks (for all modern multitasking OS)



- Authentication of users
- Protection of memory
- File IO
- Allocation and Access of Objects
- Enforcement of Sharing
- Guarantee of Fair Service
- Inter-process Communication Synchronization
- Protection of internal OS data

IT378 - M. Di Pierro @ DePaul CTI

## Secure OS Tasks



- **Memory protection**
- **File protection**
- **General object access control (and auditing)**
- **User authentication**

An Operating System is “trusted” if it provides these four services in a consistent and effective way.

An System (OS, process, product, software) can never be “secure” (i.e. withstand all attacks) but it can be “trusted” or secure enough to serve its purposes.



IT378 - M. Di Pierro @ DePaul CTI

## Why memory protection?



```
emacs
#include <stdio.h>

int main() {
    int x;
    int* p=&x;
    while(1) {
        printf("%x\n",p);
        (*p)=0;
        p++;
    }
}
```

### Output:

**System crush on MSDOS, Segmentation Fault on Linux/Windows XP**

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Memory Protection



Fence, **Segmentation**, Paging

Like multiple fences but no relocation. The memory is divided into segments. Each segment belongs to a process and each memory cell within segmented memory is addressed by a virtual memory address <segment name, offset>.

Every time the memory cell has to be accessed the OS looks up the starting address of the segment corresponding to the segment name and adds the value of the offset.

If the offset exceeds the size of the segment the OS kills the process and report a “**segmentation fault**” error.

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Memory Protection



Fence, Segmentation, **Paging**

Like segmentation but segments have all same size and are therefore called “**pages**”.

Segments can be resized, pages cannot. Paged memory is a bit faster and easier to implement but less memory efficient.

All modern OS have segmented memory. (think of Unix/Linux and Windows 2000/XP. Only supported by 386 and later Intel processors.)

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Kernel Mode Vs. User Mode



All Modern processors support **kernel mode** and **user mode**

User programs run in "**user mode**" and when they perform critical operation such as accessing an object overseen by the OS (for example reading/writing to/from memory or file or device) they perform a "function call" to a function in the OS.

The microprocessor then switches to "**kernel mode**" and executes the function that performs the operations.

Programs/functions running in "**user mode**" do not have access to physical memory addresses while programs/functions running in "**kernel mode**" do have access to physical memory addresses.

IT378 - M. Di Pierro @ DePaul CTI

## Basic Forms of File Protection



### All-None Protection (obsolete):

- Users are trusted
- Passwords used to control all accesses
- System administrator has control over all files

### Problems with All-None Protection:

- Users should NOT be trusted!
- When multitasking problems with file/resource access
- Human intervention is required for file protection

IT378 - M. Di Pierro @ DePaul CTI

## Basic Forms of File Protection



### Group Protection (better):

- Users are authenticated
- Users are divided into groups
- Each file is owned to a group
- Every user in the groups can read/write/delete a file that belongs to the group

### However, there are still problems:

- A single user can only belong to one group
- Forces users to create multiple account
- Users have the responsibility to decide what they share with the group
- Users are unable to share files with set of users different the group itself

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Object Access Control



### Directory

The directory contains information about owner and R/W permissions for each file contained in the disk. Different users see different file permissions (Unix).

### Access Control List

Each object (file) is associated to an access control list that specifies the owner and R/W permissions on the object. (Windows)

### Access Control Matrix

A table of <subject, object, rights>

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Object Access Control



In Windows everything is an **Object**:

- A file is an **object** (has attributes, pardon... “**properties**”)
- A device is an **object** (use control panel)
- A process is an **object**

In Windows a process running in background not connected to the console is called a **Service** (**daemon** in **UNIX/Linux**)

Windows allows for **more sophisticated security policies** than Linux but requires more setup time.

Windows also supports **event auditing**.

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Authentication



Every OS needs to be able to **Authenticate users**.

### Common systems:

- Simple password file (bad)
- Encrypted password file (ok for single system)
- Network Information Service (NIS)
- Pluggable Authentication Modules (PAM) (Linux only)
- Kerberos

**Kerberos** is a computer network authentication protocol designed for use on insecure networks (the Internet for example), based on the key distribution model of Needham and Schroeder. It allows individuals communicating over a network to prove their identity to each other while also preventing eavesdropping or replay attacks, and provides for detection of modification and the prevention of unauthorized reading. Kerberos has become commercially important since Microsoft introduced a version of Kerberos in the Windows 2000 version of the Microsoft Windows operating system. Kerberos is cross-platform and is available for Unix/Linux/BSD as well.

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Authentication



Main weaknesses is the choice of passwords. According to statistics:

- 1% single ascii char (256 combinations)
- 2% two ascii chars (256\*256 combinations)
- 14% three ascii chars (256\*256\*256 combinations)
- 14% fours chars (52\*52\*52\*52 combinations)
- 21% five chars same case (26\*26\*26\*26\*26 combinations)
- 18% six lowercase letters (26\*26\*26\*26\*26\*26 combinations)
- 15% word in dictionary (200000 combinations)
- 86% all of above**

And anyway 80% of the workers would give away password in exchange for a [cheap pen!](#)

### How to create “better” passwords:

Long passwords, mix upper case, lower case and numbers.

IT378 - M. Di Pierro @ DePaul CTI

## Desired Features of a Trusted OS



### Mandatory Access Control:

Owner's of the object do not decide object's permission

### Object Reuse Protection:

When an object is “recycled”, for example memory is freed, the object must be blanked

### Complete Mediation:

Access to all objects should be checked, not only to some objects

### Trusted Paths:

Users/programs need to be able to verify who they are talking to

### Auditing and Audit Log Reduction:

Log important events

### Intrusion Detection:

Try to detect failures in the security system by looking for anomalous patterns

IT378 - M. Di Pierro @ DePaul CTI

## Kernelized Design



**Kernel** = nucleous = core of the OS

It is responsible for:

- Checking every access (reference monitor)
- Isolating security tasks from the users and the users from each other

All security functions performed by a single piece of code easy to identify ([TCB](#))  
possibly a small piece of code so it is easy to check it

**TCB = Trusted Computing Base** takes care of

- Hardware
- Files
- Protected memory
- Inter-process communication

IT378 - M. Di Pierro @ DePaul CTI

## Virtual Machine Design



The best way to achieve security and complete isolation is simulating an Entire computer ([a virtual machine or sandbox](#)) for each process (and not necessarily the same one as the real machine).

This works but the virtual machine may still need to access “real” files, “real” networks and “communicate” with other virtual machines.

**Java Virtual Machine** is an example. It was created to execute un-trusted Java Applets downloaded from the net.

Because Java “needs” the virtual machine, Java programs cannot run in kernel mode and one cannot use Java to write an OS (no access to physical memory address)

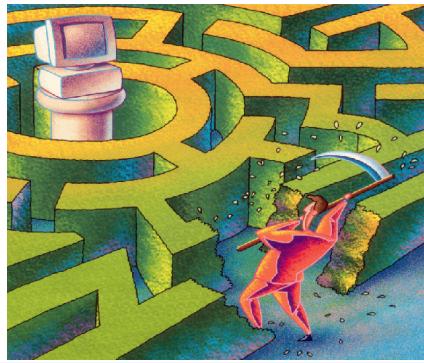
OS are written in C/C++

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks



# Types of Attacks

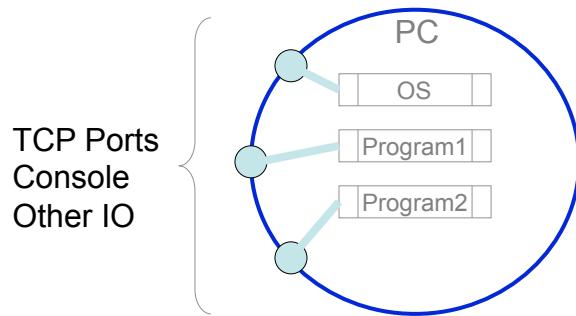


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks



Any computer is connected to the outside via a console, terminals, internet (TCP ports) and other IO devices.



IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks

A diagram illustrating a basic network attack. On the left, a gray circular icon with a sad face and the word "hacker" below it has an arrow pointing to a central node. This central node is a blue circle labeled "PC". Inside the "PC" circle are three rectangular boxes: "OS", "Program1", and "Program2".

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Network Monitoring

All TCP/IP communications between two hosts can be intercepted by somebody monitoring the network.

A diagram illustrating network monitoring. On the left, a gray circular icon with a sad face and the word "hacker" below it is connected by a double-headed arrow to a separate blue circle. This separate blue circle is connected by a double-headed arrow to a central node. The central node is a blue circle labeled "PC". Inside the "PC" circle are three rectangular boxes: "OS", "Program1", and "Program2".

IT378 - M. Di Pierro @ DePaul CTI

## OS Tasks: Object Access Control

A diagram illustrating Object Access Control. On the left, a blue circle represents a hacker. In the center, a blue circle labeled "PC" contains three rectangular boxes: "OS", "Program1", and "Program2". Three light blue arrows point from the "hacker" to each of the three objects within the "PC" circle.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Denial of Service

A diagram illustrating a Denial of Service attack. On the left, two groups of icons are shown: "hackers" (two sad faces) and "Zombie PCs" (four sad faces). Arrows point from both groups towards the central "PC" circle, which contains "OS", "Program1", and "Program2".

Multiple messages jam the program that becomes unavailable.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: DoS from Zombie Network

A zombie computer is a computer attached to the Internet that has a backdoor.

This backdoor allows the computer to be remote-controlled by others.

A Zombie Computer army can then be used for the purpose of Denial of Service attacks (DoS).

A single Zombie Computer can send unsolicited e-mails (spamming).

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: IP Spoofing

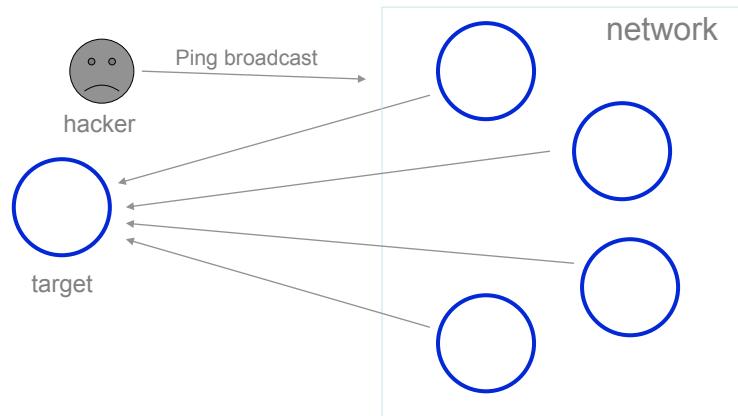
Hacker sends messages using a somebody else's address and sender address

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: IP Spoofing



Hacker broadcasts a message with a spoofed address and target. Target is flooded.

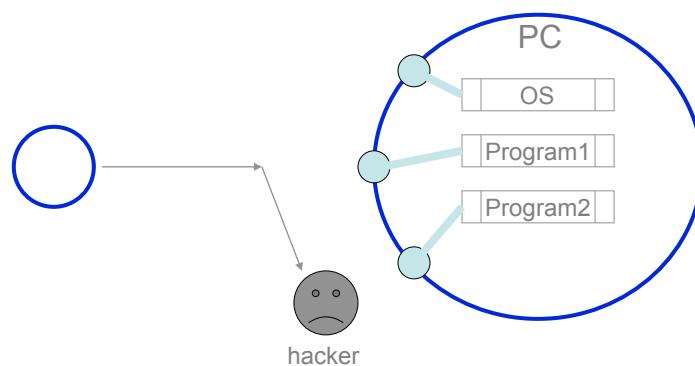


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: DNS poisoning, ARP



An attacker may [pretend to be somebody else](#) and steal (IP, TCP) messages by temporarily changing an entry in a DNS server or by sending fake responses to ARP requests.

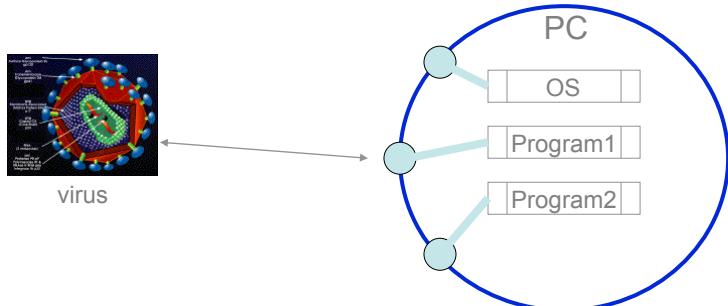


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Viruses



The virus enters via email (you click on the attachment) or exploiting the vulnerability in some program (such as buffer overflow)

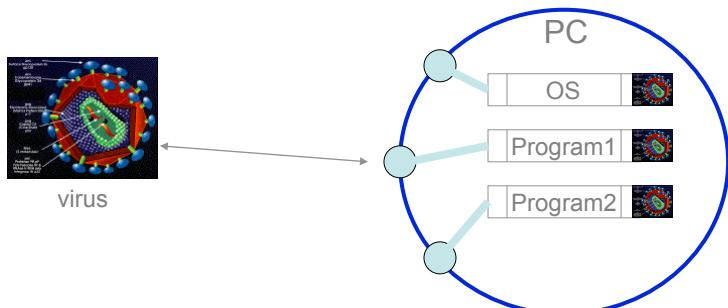


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Viruses



The virus reproduces itself by attaching the virus code to other programs and modifies system registries, on windows.

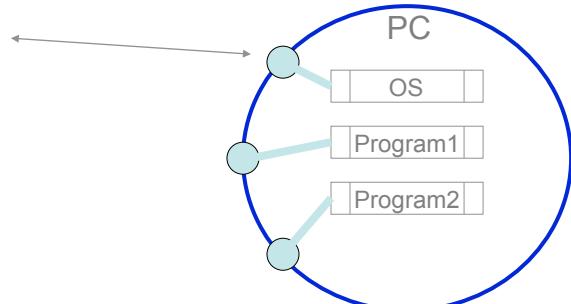


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Worms



The worm enters via email (you click on the attachment) or exploiting the vulnerability in some program (such as buffer overflow)

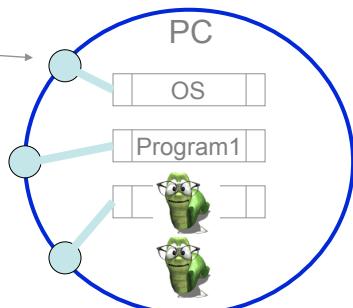


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Worms



The worm creates copies of itself, perhaps replaces some existing programs or files, modifies system registries (on windows).

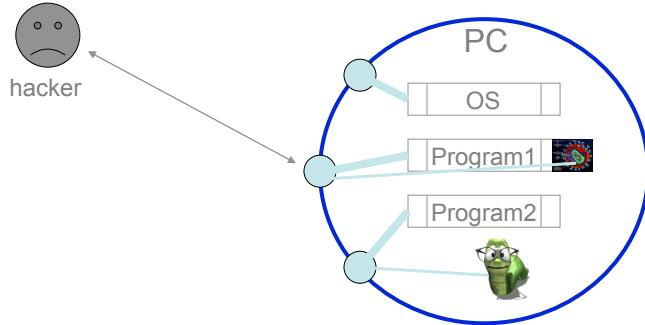


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Backdoors



Malicious programs (such as worms or programs infected by viruses) may install or may act as backdoor to get instructions from hacker, so that PC can be used to perform attacks to other parties (zombie networks)



IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Backdoors



A backdoor in a computer system (or a cryptosystem, or even in an algorithm) is a method of bypassing normal authentication or obtaining remote access to a computer, while intended to remain hidden to casual inspection.

The backdoor may take the form of an installed program (e.g., Back Orifice, *this not a typo*) or could be a modification to a legitimate program.

In 2003 a backdoor in Linux was discovered:

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
retval = -EINVAL;
```

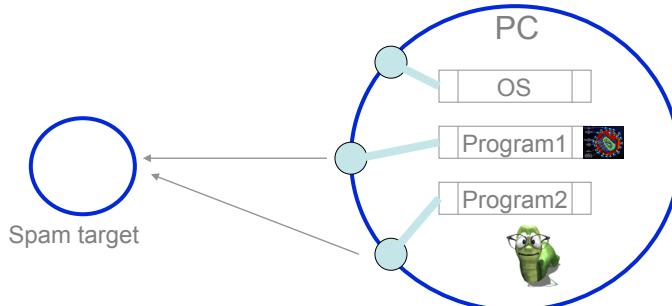
Any user calling sys\_wait4() would become root!

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Spamming



Zombie networks and infected PCs are typically used for spamming or DoS

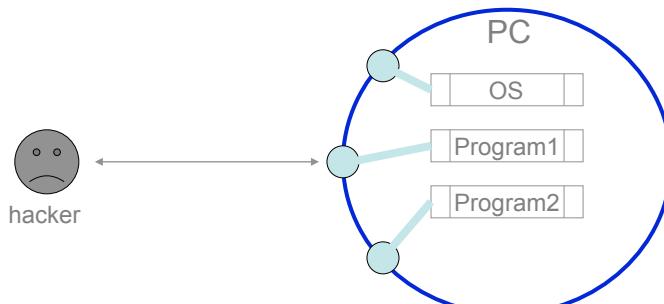


IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow



How does the hacker or a malicious program enters in a computer without a password  
And/or without using email attachments?



It exploits vulnerabilities in programs (or the OS), such as buffer overflow.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow



In computer programming, a **buffer overflow** is an anomalous condition where a program somehow writes data beyond the allocated end of a buffer in memory. Buffer overflows usually arise as a consequence of a bug, and are a commonly exploited computer security risk — since program code often sits in the memory areas adjacent to data buffers, by means of a buffer overflow condition the computer can be made to execute arbitrary (and potentially malicious) code that is fed to the buggy program as data.

[http://www.windowsecurity.com/articles/Analysis\\_of\\_Buffer\\_Overflow\\_Attacks.html](http://www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html)

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow

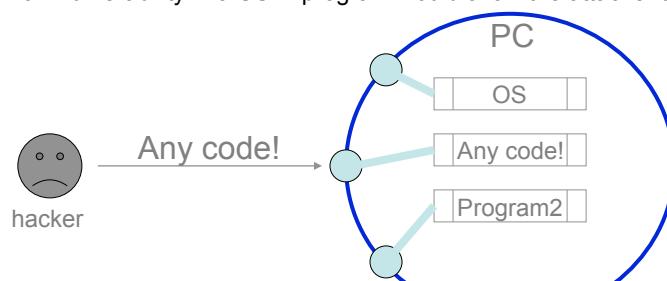


Buffer overflow is the **single most dangerous source of danger**.

Buffer overflow is **consequence of sloppy programming**.

If there is a buffer overflow vulnerability an attacker can execute any code he wants with the privileges of the program that has the vulnerability.

A buffer overflow vulnerability in a SUID program would allow the attacker to become root.



IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow



emacs

```

• void foo(char *s)
{
•     char buf[10];
•     strcpy(buf, s);
•     printf("buf is
          ", buf);
    
```

memory

allocated for buf

thisstringisstolongforfoo

LOC - CROSSED INTO

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow



emacs

```

•     FILE* fstream;
•     ...
•     char buf[10];
•     fscanf(fstream, "%[^\\0]s", buf);
    
```

memory

allocated for buf

thisstringisstolongforfoo

IT378 - M. Di Pierro @ DePaul CTI

## Types of Attacks: Buffer Overflow



### •General Information:

- The general idea is to give servers very large strings that will overflow a buffer.
- For a server with sloppy code – it's easy to crash the server by overflowing a buffer.
- It's sometimes possible to actually make the server do whatever you want (instead of crashing).



IT378 - M. Di Pierro @ DePaul CTI

## Viruses and Worms: History



1981 First “real” virus (spread via Apple II floppy disks)

1984 First Experimental Virus Created (Fred Cohen's)

1988 Two brothers from Pakistan analyzed the **boot sector** of a floppy disk and developed a method of infecting it with a virus dubbed “Brain”

1989 File infectors ([command.com](http://command.com)) and first worms

1992 First internet worms on the wild

2000 Year of the Hacker Hackers attacked Griffith Air Force Base, the Korean Atomic Research Institute, NASA, Goddard Space Flight Center, and the Jet Propulsion Laboratory.

Today VBS worms, RPC worms, etc.

IT378 - M. Di Pierro @ DePaul CTI

## Viruses Vs. Worms



A virus is a self-replicating program that spreads by inserting copies of itself into other executable code or documents

Thus, a computer virus behaves in a way similar to a biological virus, which spreads by inserting itself into living cells. Extending the analogy, the insertion of the virus into a program is termed *infection*, and the infected file (or executable code that is not part of a file) is called a *host*.

While viruses can be intentionally destructive (for example, by destroying data), many other viruses are fairly benign or merely annoying. Some viruses have a delayed payload, which is sometimes called a *bomb*. For example, a virus might display a message on a specific day or wait until it has infected a certain number of hosts. However, the predominant negative effect of viruses is their uncontrolled self-reproduction, which wastes or overwhelms computer resources.

IT378 - M. Di Pierro @ DePaul CTI

## Viruses Vs. Worms



A computer worm is a self-replicating computer program, similar to a computer virus. A virus attaches itself to, and becomes part of, another executable program; however, a worm is self-contained and does not need to be part of another program to propagate itself.

The name 'worm' was taken from *The Shockwave Rider*, a 1970s science fiction novel by John Brunner. Researchers writing an early paper on experiments in distributed computing noted the similarities between their software and the program described by Brunner and adopted the name.

IT378 - M. Di Pierro @ DePaul CTI

## Worm Example: MyDoom



Mydoom, also known as Novarg, Mimail.R and Shimgapi, is a computer worm affecting Microsoft Windows. It was first sighted on January 26, 2004. It became the fastest spreading email worm ever (as of January 2004), exceeding previous records set by the Sobig worm.

Early coverage held that the sole purpose of the worm was to perpetrate a distributed denial-of-service attack (DoS) against SCO Group. 25% of Mydoom.A-infected hosts targeted www.sco.com with a flood of traffic.

Mydoom appears to have been commissioned by e-mail spammers so as to send junk e-mail through infected computers.

IT378 - M. Di Pierro @ DePaul CTI

## Anti-virus Software



An anti-virus software program is a computer program that attempts to identify and eliminate computer viruses and other malicious software (malware). Anti-virus software typically uses two different techniques to accomplish this:

Examining (scanning) files to look for known viruses matching definitions in a virus dictionary

Identifying suspicious behavior from any computer program which might indicate infection

Most commercial anti-virus software uses both of these approaches, with an emphasis on the virus dictionary approach.

A good virus database is maintained by Symantec:

<http://www.symantec.com/avcenter/vinfodb.html>

IT378 - M. Di Pierro @ DePaul CTI

## Types of Viruses



### Polymorphic Viruses

Polymorphic viruses change with each infection. They do this in an attempt to defeat scanners.

Virus writing tool kits have been created to "simplify" creation of new viruses.

### Stealth Viruses

A computer virus that actively hides itself from antivirus software by either masking the size of the file that it hides in or temporarily removing itself from the infected file and placing a copy of itself in another location on the drive, replacing the infected file with an uninfected one that it has stored on the hard drive.

To better find stealth viruses be certain to cold boot from a known-clean (write protected) floppy disk and avoid using generic DOS commands to try to fix them. Use anti-virus software to handle these viruses.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Viruses



### Boot Sector Viruses

Viruses that hide in the boot sector and transfer themselves in memory before the Operating System.

### Memory Resident Viruses

A virus that hides inside the Operating System, a Dynamically Linked Library or any other piece of code that remains in memory for long time.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Viruses



### Fast and Slow Infectors

Viruses that infect in a particular way to try to avoid specific anti-virus software.

A fast infector infects programs when they are accessed, not just when run. This type of virus is designed to ride on the back of anti-virus scanners and can quickly infect an entire disk if not found before the scan is performed.

A slow infector infects programs only when they are created or modified. This type of virus is designed to defeat integrity checkers but can usually be found if the checker has a scanner component or is started properly.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Viruses



### Armored Viruses

Viruses that are programmed to make disassembly difficult.

### Multipartite Viruses

Viruses that may fall into more than one of the top classes.

### Cavity (Spacefiller) Viruses

Viruses that attempt to maintain a constant file size when infecting.

A cavity virus attempts to install itself inside of the file it is infecting.

In the past this was difficult to do properly, but new file formats make it easier.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Viruses



### Tunneling Viruses

Viruses that try to "tunnel" under anti-virus software while infecting.

### Camouflage Viruses

Viruses that attempted to appear as a benign program to scanners.

### NTFS ADS Viruses

Viruses that ride on the alternate data streams in the NT File System (ADT Files)

For more info:

<http://www.dmares.com/maresware/html/ads.htm>  
(very important for forensics)

IT378 - M. Di Pierro @ DePaul CTI

## VBS Viruses and Worms



Basically, think about VBScript as a super batch language. VBScript is an interpreted language (so scripts are really the source code for whatever needs to be done). Scripts can be embedded into such things as web pages or can be standalone files (with the extension .VBS usually).

If you've got Microsoft's Internet Explorer 5 browser on your system it's likely you also have the Windows Scripting Host (WSH) which is the program used to interpret and run VBS scripts.

Even though VBScript is a scaled down language it is quite capable and can be used to, connect to Microsoft's Outlook mail routines and send files to anyone in your address book. This, of course, makes it possible for VBScript to be a language used by worms to spread themselves.

VBH can be disabled on your system

IT378 - M. Di Pierro @ DePaul CTI

## How to secure a host



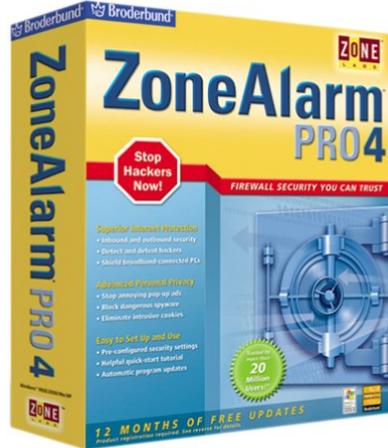
- 1) Implement strong policies (make backups, force users to use strong passwords and change them often, forbid users to install programs, regulate and minimize access to resources, keep system updated)
- 2) Only run trusted programs (do not use unnecessary programs, only get programs from trusted sources, check digital signatures)
- 3) Minimize network access (uninstall unwanted services, close unused ports)
- 4) Monitor and audit suspicious activities
- 5) Do not trust software vendors (easy of use != secure)

IT378 - M. Di Pierro @ DePaul CTI

**Topic:**



## Firewalling Windows



IT378 - M. Di Pierro @ DePaul CTI

## Firewall: Allow/Deny Protocols

For each zone  
Trusted or Internet...

You can specify which protocols should be allowed (always specify what allow and default denied)

Example of secure settings:  
allow out HTTP/HTTPS  
allow in/out SSH

**Attention:**  
Deny incoming NetBIOS connection does not stop unwanted services from running: You have to stop them manually

IT378 - M. Di Pierro @ DePaul CTI

## Firewall: Allow/Deny packets by Time/Source/Destination

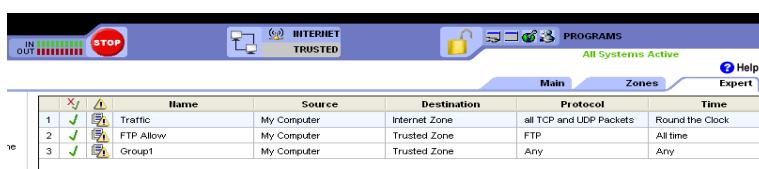
IT378 - M. Di Pierro @ DePaul CTI

**Firewall: Sorting Rules**



**Rules should be sorted by importance:**

- Allow in SSH only from ip address xxx
- Allow out SSH
- Deny out HTTP to forbidden hosts
- Deny out HTTPS to forbidden hosts
- Allow out HTTP
- Allow out HTTPS
- Deny everything else



X	Name	Source	Destination	Protocol	Time
✓	Traffic	My Computer	Internet Zone	all TCP and UDP Packets	Round the Clock
✓	FTP Allow	My Computer	Trusted Zone	FTP	All time
✓	Group1	My Computer	Trusted Zone	Any	Any

IT378 - M. Di Pierro @ DePaul CTI

**Topic:**



# Monitoring Recovery Auditing



IT378 - M. Di Pierro @ DePaul CTI

**Monitoring**

Run fake services to check if somebody tries hacking into your machine



The window title is "NFR BackOfficer Friendly - Warnings". It contains a log of HTTP requests:

```

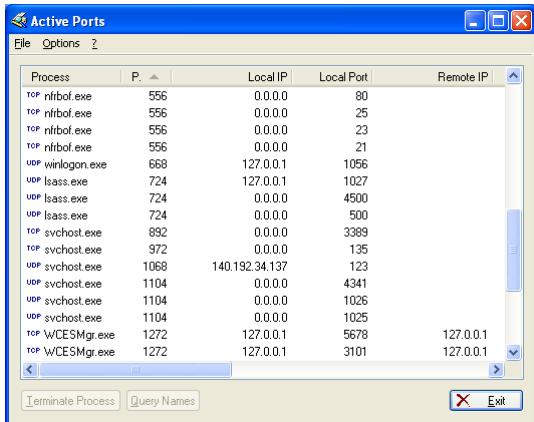
Tue Mar 22 17:24:54 HTTP empty request from 140.192.21.137
Tue Mar 22 17:30:33 HTTP empty request from 140.192.21.137
Mon Mar 28 15:10:48 HTTP request from 127.0.0.1: GET /?_request=fermiqcd/about
Mon Mar 28 15:10:55 HTTP request from 127.0.0.1: GET /?_request=fermiqcd/download
Mon Mar 28 15:10:58 HTTP request from 127.0.0.1: GET /?_request=fermiqcd/bugs

```

IT378 - M. Di Pierro @ DePaul CTI

**Monitoring**

Check for inbound and outbound connections and connecting programs



Process	P.	Local IP	Local Port	Remote IP
TCP nifbot.exe	556	0.0.0.0	80	
TCP nifbot.exe	556	0.0.0.0	25	
TCP nifbot.exe	556	0.0.0.0	23	
TCP nifbot.exe	556	0.0.0.0	21	
UDP winlogon.exe	668	127.0.0.1	1056	
UDP lsass.exe	724	127.0.0.1	1027	
UDP lsass.exe	724	0.0.0.0	4500	
UDP lsass.exe	724	0.0.0.0	500	
TCP svchost.exe	892	0.0.0.0	3389	
TCP svchost.exe	972	0.0.0.0	135	
UDP svchost.exe	1068	140.192.34.137	123	
UDP svchost.exe	1104	0.0.0.0	4341	
UDP svchost.exe	1104	0.0.0.0	1026	
UDP svchost.exe	1104	0.0.0.0	1025	
TCP WCESMgr.exe	1272	127.0.0.1	5678	127.0.0.1
TCP WCESMgr.exe	1272	127.0.0.1	3101	127.0.0.1

IT378 - M. Di Pierro @ DePaul CTI

## Monitoring

Monitor Network traffic

(Live Capture in Progress) - Ethereal

No. | Time | Source | Destination | Protocol | Info

1	0.000000	cisco_55:b2:c5	Spanning-tree-Bridge	comポート	→ 27.0.0.1:52000<– 27.0.0.1:52000
2	0.941410	0000169a.0030c154	0000169a.ffffffff	IPX SA General Query	
3	1.892302	140.192.44.130	140.192.44.191	NBNS	Name query NB PRINTERS<20>
4	1.999078	cisco_55:b2:c5	Spanning-tree-(for	STP	Conf. Root = 32768/00:02:16:84:
5	2.642315	140.192.44.130	140.192.44.191	NBNS	Name query NB PRINTERS<20>
6	2.856325	140.192.34.160	140.192.34.255	BROWSER	Host Announcement PTFAC702-1, w
7	3.000000	140.192.34.160	140.192.34.191	NBNS	Name query NB PRINTERS<20>
8	3.193939	140.192.34.248	Broadcast	ARP	Who has 140.192.34.189? Tell 1
9	3.641358	61.155.86.165	140.192.34.181	TCP	20088 > 14032 [SYN] Seq=0 Ack=0
10	4.000802	cisco_55:b2:c5	Spanning-tree-(for	STP	Conf. Root = 32768/00:02:16:84:
11	5.003993	140.192.34.137	140.192.36.3	DNS	Standard query PTR 248.34.192.1
12	5.004458	140.192.36.3	140.192.34.137	DNS	Standard query response, No suc
13	5.420093	60.163.50.246	140.192.34.181	TCP	4233 > 14032 [SYN] Seq=0 Ack=0
14	5.542658	0000169a.0030c1607	0000169a.ffffffff	IPX SA General Response	
15	5.631561	140.192.34.248	Broadcast	ARP	Who has 140.192.44.161? Tell 1
16	5.999143	cisco_55:b2:c5	Spanning-tree-(for	STP	Conf. Root = 32768/00:02:16:84:

Frame 1 (60 bytes on wire, 60 bytes captured)  
 ▷ IEEE 802.3 Ethernet  
 ▷ Logical-Link Control  
 ▷ Spanning Tree Protocol

0000 01 80 c2 00 00 00 00 06 53 e5 b2 c3 00 26 42 42 ..... SU...&BB  
 0001 03 00 00 00 00 00 80 00 00 02 16 2e a7 00 00 .....SU.....  
 0020 08 00 00 00 06 53 35 b2 e9 80 11 02 00 14 00 .....SU.....  
 0030 02 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 .....

[Device[NPF\_{IC05EIE7-4}]:P: 16 D: 16 M: 0]

IT378 - M. Di Pierro @ DePaul CTI

## Recovery

If you know you have been attacked the safest way out is:

- 1) Disconnect from network
- 2) Save your data
- 3) Format all disks
- 4) Reinstall the OS and all your programs
- 5) Scan the data you backed up for viruses
- 6) Restore the saved data (not the programs)

In fact if somebody got control on your machine there is no way to tell with 100% certainty what has been compromised.

If the attack consists of a known harmless virus or spyware, probably removing the malicious program and changing password is sufficient.

Sometimes one can assess the gravity of the situation by monitoring network traffic, looking for backdoors, auditing the system.

IT378 - M. Di Pierro @ DePaul CTI

## Recovery: How to save data



If the system has been compromised but the HD is working you can [boot Windows in "Safe Mode"](#) by pressing F8 at boot time, then disconnect from network and choose "[Safe Mode with Networking](#)".

Windows will boot but will only install a minimal set of services and, hopefully, they have not been compromised.

Now backup your system.

Even when booting in "Safe Mode" windows will write on disk and will try to repair the disk if damaged. Occasionally this may cause more harm than good.

IT378 - M. Di Pierro @ DePaul CTI

## Recovery: How to save data using Knoppix



**Best Practice:** Do not use windows for recovery. Boot a linux CD and mount the HD read only, then backup data. After your data is safely backed up, try recovery with Windows. If this fails get reinstall the system from scratch and restore backed up data.

To copy /dev/hda1 to /dev/hdb2 boot with Knoppix and execute:

```
knoppix@tty0[knoppix] sudo mount /dev/hda1 /mnt/hda1
knoppix@tty0[knoppix] sudo mount -o rw /dev/hdb1 /mnt/hdb1
knoppix@tty0[knoppix] cd /mnt/hda1
knoppix@tty0[hda1] sudo sh -c "find . -xdev -print0 | cpio
-pa0V /mnt/hdb1"
```

Reading suggestion: <http://www.oreilly.com/catalog/knoppixhks/>

IT378 - M. Di Pierro @ DePaul CTI

## Security Auditing Windows OS



There are different types of auditing:

**Basic:**

Look at the log files (system logs, program logs)

**Pros:** Tells you about past events.

**Cons:** Only tells you about events logged

**Advanced:**

Look at the logical disk content

**Pros:** Tells you everything about present system status.

**Cons:** Nothing about deleted files.

**More Advanced:**

Look at the physical disk content

**Pros:** Same as advanced

**Cons:** Tells you a great deal about deleted files, difficult to do.

**Extremely Advanced:**

Explore the disk at hardware level

**Pros:** Tells you about past history of each bit.

**Cons:** Almost impossible to do

IT378 - M. Di Pierro @ DePaul CTI

## Security Auditing



First steps to auditing a system:

- 1) Switch computer off (do not do proper shutdown)
- 2) Boot computer using a Bootable device such as CD (you can use Knoppix)
- 3) Mount the HDs of the system you want to audit as read only  
(Now you have access to the disk information and you are sure no program is running that is writing on that disk and corrupting information.)
- 4) Make a backup of the entire disk (use CD-R or DVD-R or other write once media)
- 5) Look at log files
- 6) Search files for suspicious activity
- 7) Search disk at physical level for deleted files

IT378 - M. Di Pierro @ DePaul CTI

## Security Auditing



**NTFS** (the file system used by windows XP and 2000) supports multiple data streams in a file. The main data stream is named DATA. The main data stream stores the main content. This is the perfect place for viruses and Trojans to hide.

How does somebody copy data into an ADS?

```
type anyfile > visible.txt:hidden.txt
```

This will create another hidden stream hidden.txt in the file visible.txt.

How does somebody copy data from an ADS into a "normal" file?

```
type visible.txt:hidden.txt > normal.txt
```

This will create a file normal.txt from the hidden stream hidden.exe in the file visible.txt.

How does somebody delete an ADS?

```
ren visible.txt temp.txt
type temp.txt > saved.txt
del temp.exe
ren save.txt > visible.txt
```

Microsoft provides no tool to detect ADS.

IT378 - M. Di Pierro @ DePaul CTI

## Windows Log Auditing



[Control Panel][Administrative Tools][Event Viewer]

Type	Date	Time	Source
Success Audit	3/27/2005	9:59:00 PM	Security
Success Audit	3/27/2005	9:59:00 PM	Security
Success Audit	3/27/2005	9:59:00 PM	Security
Failure Audit	3/27/2005	9:56:04 PM	Security
Success Audit	3/27/2005	9:37:38 AM	Security
Success Audit	3/27/2005	9:37:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	2:22:39 AM	Security

IT378 - M. Di Pierro @ DePaul CTI

## Windows Log Auditing



**Application log**

The application log contains events logged by programs. For example, a database program may record a file error in the application log. Events that are written to the application log are determined by the developers of the software program.

**Security log**

The security log records events such as valid and invalid logon attempts, as well as events related to resource use, such as the creating, opening, or deleting of files. For example, when logon auditing is enabled, an event is recorded in the security log each time a user attempts to log on to the computer. You must be logged on as Administrator or as a member of the Administrators group in order to turn on, use, and specify which events are recorded in the security log.

**System log**

The system log contains events logged by Windows XP system components. For example, if a driver fails to load during startup, an event is recorded in the system log. Windows XP predetermines the events that are logged by system components.

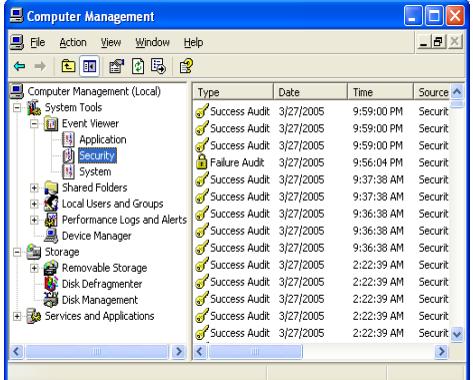
IT378 - M. Di Pierro @ DePaul CTI

## Windows Log Clearing



You should check your logs and clear them often (you may also want to clear them to remove records of some action). You can do so by:

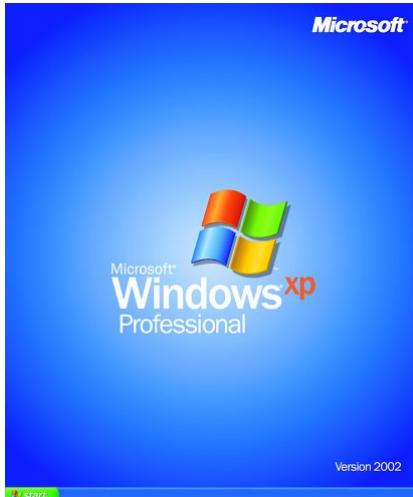
[Control Panel][Administrative Tools][Computer Management]select [Security]  
in the menu select [Action][Clear All]



Type	Date	Time	Source
Success Audit	3/27/2005	9:59:00 PM	Security
Success Audit	3/27/2005	9:59:00 PM	Security
Success Audit	3/27/2005	9:59:00 PM	Security
Failure Audit	3/27/2005	9:56:04 PM	Security
Success Audit	3/27/2005	9:37:38 AM	Security
Success Audit	3/27/2005	9:37:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	9:36:38 AM	Security
Success Audit	3/27/2005	2:22:39 AM	Security
Success Audit	3/27/2005	2:22:39 AM	Security
Success Audit	3/27/2005	2:22:39 AM	Security
Success Audit	3/27/2005	2:22:39 AM	Security

IT378 - M. Di Pierro @ DePaul CTI

**Topic:**



A screenshot of a Windows XP Professional desktop. The desktop is blue with a Microsoft logo in the top right corner. In the center is the Windows XP logo with the text "Microsoft Windows xp Professional". At the bottom, there's a taskbar with a "Start" button.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS



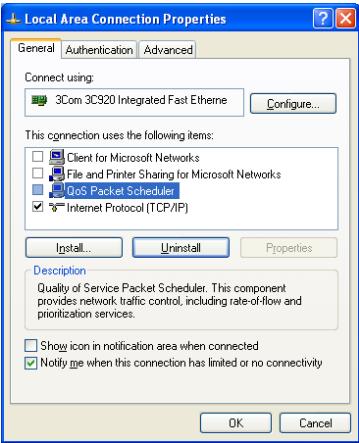
**Important:** All security settings should be done immediately after a fresh windows installation and before connecting the system to the network or installing any program. Login as Administrator.

- 1) Close all unused ports
- 2) Turn on default firewall (or better install your own firewall)
- 3) Secure IE settings (later install Mozilla Firefox and try not to use IE)
- 4) Secure Outlook (oops... uninstall Outlook Express)
- 5) Deal with users (create/edit/disable accounts)
- 6) Create Secure logins
- 7) Disable unnecessary services and unwanted programs
- 8) Set file permissions and encrypt data folders
- 9) Other settings (for example disable VBScript)
- 10) Install anti-virus, anti-spyware and intrusion detection software and other tools
- 11) Reboot
- 12) Perform a port scan and security scans of your machine periodically
- 13) Periodically install security updates and change passwords
- 14) Setup a restore point

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports

[Control Panel][Network Connections]  
for each item [Properties]  
for each item other than TCP/IP [Uninstall]



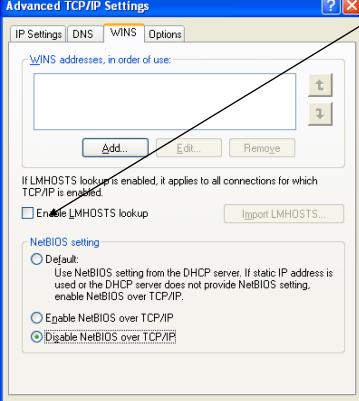
**What's QoS ?**

In the context of networking, Quality of Service (QoS) refers to a combination of mechanisms that cooperatively provide a specific quality level to application traffic crossing a network or multiple, disparate networks. Without QoS, each of these network devices treat all data equally and provide service on a first-come, first-served basis.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports

[Control Panel][Network Connections]  
for each item [Properties]  
for the TCP/IP item [Properties][Advanced][WINS]  
Remove: Enable LMhosts lookup



**What's LMHOSTS?**

Specific to [Windows](#), the LMHOSTS file is a plain text file (without a [file extension](#)) that tells your computer where to find another computer on a [network](#). The file resides in the Windows directory, and it lists the computer names ([NetBIOS](#)) and [IP addresses](#) of machines you access on a regular basis.

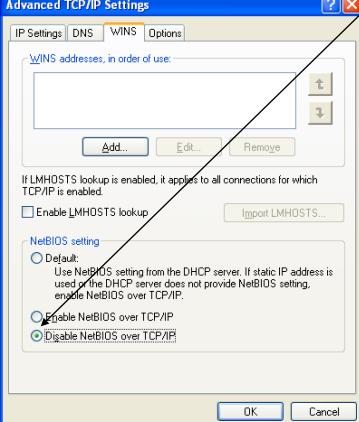
IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :1) Close unused ports**



[Control Panel][Network Connections]  
for each item [Properties]  
for the TCP/UP item [Properties][Advanced][WINS]

**Advanced TCP/IP Settings**



Select: Disable NetBIOS over TCP/IP

**What's NetBIOS?**

NetBIOS is a [networking protocol](#), co-developed by [IBM](#) and Sytec in the early 1980s. As PC-Network hardware is no longer used, having been replaced by [TokenRing](#) and [Ethernet](#) networks, the NetBIOS protocol might not be needed anymore. However, because a lot of software was written for NetBIOS's API, it has since been adapted to work over various other protocols such as [IPX/SPX](#) and [TCP/IP](#). NetBIOS over TokenRing or Ethernet is now referred to as [NetBEUI](#). NetBIOS over TCP/IP is referred to as [NBT](#). It intends to provide a NetBIOS-based PC-Network LAN emulation over a routed IP-based internetwork. It has been introduced with Microsoft Windows 2000 and is now the preferred NetBIOS transport.

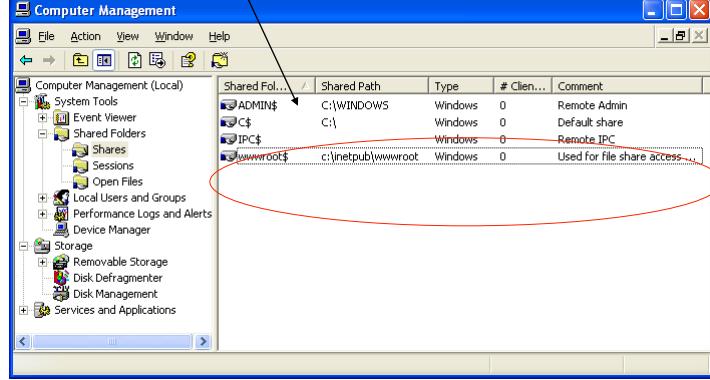
IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :1) Close unused ports**



[Control Panel][Administrative tools][Computer management][Shared folders]  
[Shares]  
delete everything inside

**Computer Management**



Shared Fol...	Shared Path	Type	# Clien...	Comment
ADMIN\$	C:\WINDOWS	Windows	0	Remote Admin
C\$	C:\	Windows	0	Default share
IPC\$		Windows	0	Remote IPC
wwwroot\$	c:\inetpub\wwwroot	Windows	0	Used for file share access

New viruses and worms can use weak passwords on shared network volumes to spread. You want to remove shared folders to avoid spreading or receiving viruses.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports



run `regedit.exe` then

in `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Ole`  
**EnableDCOM** (REG\_SZ) Set to: N

Name	Type	Data
(Default)	REG_SZ	(value not set)
DefaultLaunchP...	REG_BINARY	01 00 04 80 5c 00 00 00 6c 00 00 00 00 00 00 14 0...
EnabledCOM	REG_SZ	Y
MachineAccessB...	REG_BINARY	01 00 04 80 44 00 00 00 54 00 00 00 00 00 00 14 0...
MachineLaunchR...	REG_BINARY	01 00 04 80 48 00 00 00 58 00 00 00 00 00 00 14 0...

What is **DCOM**?  
In Windows, applications (such as word processors and spreadsheets) can be written to expose their internal functions as COM objects, allowing them to be "automated" instead of manually selected from a menu. COM can be run locally or remotely via DCOM. The terms COM object, ActiveX object and ActiveX component are synonymous.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports



in `HKLM\SYSTEM\CurrentControlSet\Services\NetBT\Parameters`  
**SmbDeviceEnabled** (REG\_DWORD) Set to: 0

in `HKLM\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters`  
**MaxCachedSockets** (REG\_DWORD) Set to: 0

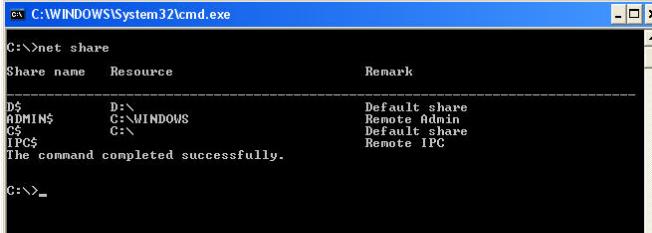
What is **Dnscache**?  
When you say "www.yahoo.com" this is translated into the IP address "68.142.226.45" by the DNS. Your machine runs a service called dnscache whose job is to request translations to the DNS server. If you do not set MaxCachedSockets to 0 then your machine always uses the same port (1026) to talk to the DNS server and this is a hazard.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports



in HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\  
 REG\_DWORD AutoShareServer Set to: 0 and AutoShareWks Set to: 0  
 in HKLM\System\CurrentControlSet\Services\LanManServer\Parameters\NullSession  
 Pipes  
 NullSessionPipes (Delete all value data INSIDE this key)  
 NullSessionShares (Delete all value data INSIDE this key)



Every Windows NT/W2K/XP/2003 machine automatically creates a share for each drive on the system. These shares are hidden, but available with full control to domain administrators. The drive letter, followed by the \$ sign is the name, and it is shared from the root.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :1) Close unused ports



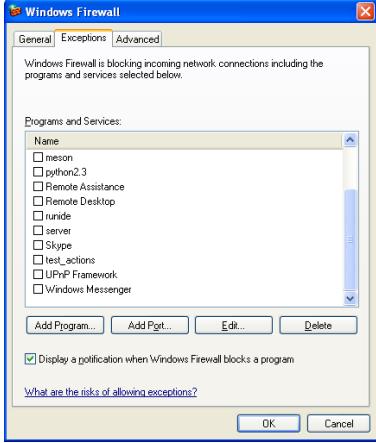
in HKLM\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg\  
 \AllowedPaths\  
 Machine (Delete all value data INSIDE this key)

The items listed here are registry keys for which you allow remote access. You do not want to give remote address to any registry key!

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :2) Enable Firewall

[Control Panel][Network Connections]  
 for each connection [Properties][Advanced][Internet Firewall]Enable  
 [settings]everuthing should be disabled



**What is a **firewall**?**

It is a program that perform packet filtering and blocks un-authorized or malformed packets from being received.

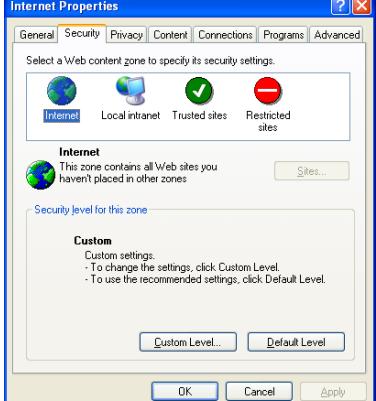
It can block packets from a specific source or block packets for specific destination ports.

Windows has a default firewall but there are better firewall around. A good and free one is ZoneAlarm.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :2) Enable Firewall

[Control Panel][Internet Options][Security]  
 [Internet] Set Custom level to high and reset  
 [Local intranet][Sites]remove everything  
 [Trusted sites][Sites]remove everything you do not trust



**Always assume you cannot trust unless you are absolutely sure you must trust in order to obtain a functionality you need.**

Never assume you can trust an unknown source.

Anyway IE suffers is a very complex browser and it tends to execute scripts in pages which may contain malicious code (such as ActiveX controls)

Do not use IE. Use Mozilla.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :3) Secure IE

[Control Panel][Internet Options]

[Privacy][Advanced] set as shown below (left)  
[Content][Autocomplete] set as shown below (right), click the clear buttons

**Advanced Privacy Settings**

You can choose how cookies are handled in the Internet zone. This overrides automatic cookie handling.

**Cookies**

Override automatic cookie handling

First-party Cookies      Third-party Cookies

Accept       Accept  
 Block       Block  
 Prompt       Prompt

Always allow session cookies

**OK**    **Cancel**

**AutoComplete Settings**

AutoComplete lists possible matches from entries you've typed before.

Use AutoComplete for

Web addresses  
 Forms  
 User names and passwords on forms  
 Prompt me to save passwords

Clear AutoComplete history

**Clear Forms**    **Clear Passwords**

To clear Web address entries, on the General tab in Internet Options, click Clear History.

**OK**    **Cancel**

**What are Cookies?**  
Web sites may ask your browser to store data on your machine for later retrieval. This data is stored in cookies. Some cookies are used to maintain continuity when navigating a web site (session cookies). Other cookies are set to track you.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :3) Secure IE

[Control Panel][Internet Options]

[advanced] disable everything but items in the list on the right

**Internet Properties**

General Security Privacy Content Connections Programs Advanced

**Settings:**

- Accessibility
  - Always expand ALT text for images
  - Move system caret with focus/selection changes
- Browsing
  - Always send URLs as UTF-8 (requires restart)
  - Automatically check for Internet Explorer updates
  - Close unused folders in History and Favorites (requires restart)
  - Disable Script Debugging (Internet Explorer)
  - Disable Script Debugging (Other)
  - Display a notification about every script error
  - Enable Folder View for FTP sites
  - Enable Install On Demand (Internet Explorer)
  - Enable offline items to be synchronized on a schedule
  - Enable page transitions
  - Enable Personalized Favorites Menu

**Always send URL:s as UTF-8**  
**Disable script debugging**  
**Enable folder view on FTP sites**  
**Enable page transitions**  
**Show friendly http error messages**  
**Show go button in address bar**  
**Use passive ftp**  
**Use smooth scrolling**  
**Use http 1.1**  
**Use http 1.1 through proxy connections**  
**Dont display online media content in the media bar**  
**Play animations in webpages**

**Play sounds in webpages**  
**Lay videos in webpages**  
**Show pictures**  
**Smart image dithering**  
**Check for publishers certificate revocation**  
**Check for server certificate revocation**  
**Check signatures on downloaded programs**  
**Do not save encrypted pages to disk**  
**Use SSL 3.0**  
**Use TLS 1.0**  
**Warn about invalid site certificates**  
**Warn if form submittal is being redirected**

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :4) Deal with Users

[Control Panel][Users](or run control userpasswords2)[Advanced][Advanced]

Remover every user other than Administrator and Guest (perhaps disable) in the menu [Action][new user] to add new users make sure users below to the right Group

In windows each group has members. One user can belong to more of one group. When granting or forbidding access to a resource (file, device, etc) you can do it by user or by group.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :5) Secure login...

[Control Panel][Users](or run control userpasswords2)[Advanced]

check secure login

Some program mimic the login and can steal your password

With Windows secure login you are asked to press [Ctrl]+[Alt]+[Del] before the system prompts the login.

This combination of keys causes a system interrupt to go on and the OS switches to kernel mode so that you can be sure the OS is prompting you.

Attention: You can only trust the OS as long the OS itself has not been compromised.

IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :5) Secure login...**

Run: syskey.exe  
[Encryption enabled][Update][Store key locally]

This should be the default but you may want to check.

The OS needs to store login key somewhere. You can store then locally on HD or on a floppy disk (the floppy would be required at start-up). Passwords should be stored encrypted so that whoever has access to the file with the keys cannot read them. Nevertheless it is easy to crack passwords with programs such as Jack the Ripper.

IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :6) Disable Unnecessary services**

[Control Panel][Administrative Tools][Services]  
disable every service by clicking on properties,  
except those listed in the following two slides.

**What is a service?**

In windows a service is a program that runs in background (as opposed to in a window) and is not connected to console IO. Services perform various functions from maintenance to networking. They respond and cause events both internal and external to the computer. Multiple services access the network and this is a hazard if the service is not necessary.

In Unix/Linux a "service" is called a "daemon".

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :6) Disable Unnecessary services



### Services to keep:

- + **Application Layer Gateway Service** (allows application to speak to each other)
- + **Application Management** (provides access to Add/Remove of applications)
- + **Automatic Updates** (connects to microsoft and downloads windows updates)
- + **Cryptographic Services** (name says it all)
- + **DHCP Client** (allows your PC to automatically get an IP address when on network)
- + **DNS Client** (converts hostnames to IP addresses)
- + **Event Log** (logs the events such as "user attempt to perform illegal operation")
- + **Help and support** (provide the help interface for the windows OS)
- + **Human Interface Device Access** (enables generic input access such as hotkeys)
- + **HTTP SSL** (allows you to access https pages and does encryption/decryption)
- + **Internet Connection Firewall** (filters incoming packets/connections)
- + **Network Connections** (make your network cards function)
- + **Network Location Awareness** (NLA) (collects network information)
- + **Plug and Play** (if you install a new device it loads the driver to use it)
- + **Print Spooler** (queues documents in print)

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :6) Disable Unnecessary services



### Services to keep:

- + **Remote Access Connection Manager** (if you want to be able to connect remotely)
- + **Remote Procedure Call** (RPC) (used by almost everything)
- + **System Event Notification** (applications that use COM need this to be notified by events)
- + **Task Scheduler** (performs various registered tasks at scheduled time)
- + **Telephony** (deals with dialup, faxes and occasionally cable modems)
- + **Themes** (performs the fancy animations that comprise your theme)
- + **Windows Audio** (plays audio)
- + **Windows Image Acquisition** (reads from your webcam/scanner/etc.)
- + **Windows Installer** (add/remove applications provided as msi files)
- + **Windows Management Instrumentation** (provides a software interface to many OS functions)
- + **Windows Management Instrumentation Driver Extensions** (same as above)

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :6) Disable Unnecessary services



**Services to definitively delete:**

- **Remote Registry** (you do not want other PC on the net to access your registries)
- **(everything with NetBIOS)** (an old protocol, just a security hazard)
- **(network services you know you are not using)** for example FTP, Telnet, IIS, etc.)

If you are not sure about services you are using disable it. If you notice that something breaks put it back.

You will also notice services associated to programs you installed (such as antivirus, firewall, intrusion detection, etc.) If you can clearly identify the program a service is associated to and you want that program running, you probably want the corresponding service.

For more info: <http://www.theelddergeek.com/>

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :7) Disable unwanted programs



run **msconfig.exe [startup]**  
 disable all programs that you do not want to start automatically  
 and/or you do not recognize

Startup Item	Command	Location
SAGUI	C:\Program Files\PREV...	HKEY_SOFTWARE\Microsoft\Windows\CurrentVer...
gcaServ	"C:\Program Files\Micr..."	HKEY_SOFTWARE\Microsoft\Windows\CurrentVer...
WCESCOMM	"C:\Program Files\Micr..."	HKEY_SOFTWARE\Microsoft\Windows\CurrentVer...
mssmsgs	"C:\Program Files\Mes..."	HKEY_SOFTWARE\Microsoft\Windows\CurrentVer...
Photo Loader supe...	C:\PROGRA~1\CASIO...	Common Startup

**What is startup?**

Every time windows boots it automatically starts some programs (generally stuff you have installed, often spyware and viruses appear in this list).

If a malicious program has taken control of your PC you will not be able to remove it. In this case you need to reboot in secure mode (later)

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :8) Set File Permissions

As part of your security policy you may want to allow/deny access to specific files/folder to different users/groups.

To do this:

select any file and click on [Properties][Security]

for each user or group of users you can allow or deny various types of access.

Permissions for Administrator	Allow	Deny
Full Control	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Modify	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read & Execute	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Write	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Special Permissions	<input type="checkbox"/>	<input type="checkbox"/>

For special permissions or for advanced settings, click Advanced.

OK Cancel Apply

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :8) Encrypt Folders

Some folders are going to contain sensitive. Encrypt those folders

select any file and click on [Properties][Advanced]  
click on Encrypt contents to secure data

Advanced Attributes

Choose the settings you want for this folder  
When you apply these changes you will be asked if you want the changes to affect all subfolders and files as well.

Archive and Index attributes

Folder is ready for archiving  
 For fast searching, allow Indexing Service to index this folder

Compress or Encrypt attributes

Compress contents to save disk space  
 Encrypt contents to secure data

OK Cancel Details

IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :9) Other Settings**



90% of all viruses propagate as email attachments written in **Visual Basic Script** (VBS) and VBS is used almost exclusively by viruses

Do you know of any serious program written in VBS?

**Disable VBS as follows:**

find the file wscript.exe and remove it from disk!

wscript.exe is pure evil!

IT378 - M. Di Pierro @ DePaul CTI

**Securing Windows OS :9) Other Settings**



**Set your Local Security Policy**  
 [Control Panel][Administrative Tools][Local Security Policy]

- a) Force users to change password every 3 months
- b) Force users to set a long password
- c) Force users to use complex passwords
- d) Automatically lock accounts every 3 login failures
- e) Audit all login and success and failures
- f) Prevent remove access
- g) Prevent some users from accessing  
some files/services



IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :9) Other Settings

The best security tool for managing security policies is run mmc.exe

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :10) Install Antivirus...

A **good antivirus**. If you are cheap get a good free antivirus (for example AVG)

A **good firewall** (a good one if **ZoneAlarm**)

**Intrusion detection tool** (for example **prevx**, it notifies of intrusion attempts and notifies when a program attempts to edit the windows registry)

**Anti-spyware** tools. None is perfect so you may want to scan your computer periodically with more than one and keep one running resident.  
 Suggested: **Ad-Aware, Spybot, Microsoft Anti Spyware** (the latter is quite good so keep it running in backgrounds).

(Attention MS Anti Spyware requires DCOM service.)

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :11) Network Scans



Check your computer for open access by using (remotely) a port scanner.

Each to use port scanners: **Superscan, Cerberus, nmap**

Intrusion Detection: **Snort**

Check for open TCP connections: **ActivePorts, netstat**

Monitor network traffic directly: use **windump/tcpdump, Ethereal**

Monitor suspicious activity by faking open vulnerabilities: **BackOfficer**

IT378 - M. Di Pierro @ DePaul CTI

## Securing Windows OS :12) Make Backups



[Start][Accessories][System Tools][Backup]

You should consider getting a **proper backup program**.  
 You should also consider getting a program like **PartitionMagic** or **BinHex** that can do a physical disk image.  
 Incremental backup for users' data (we'll see rsync for Linux and Windows)

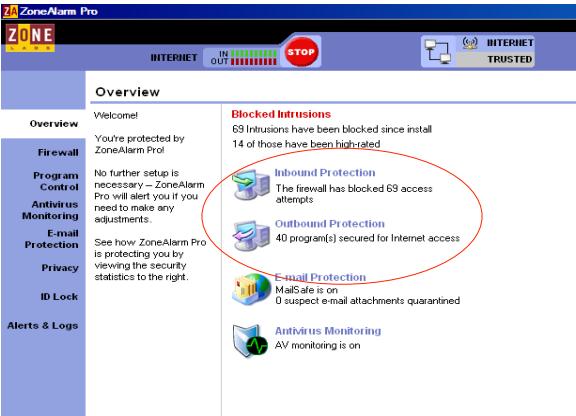
**Where to backup to?**

You can backup to another partition but the disk may fail.  
 It is better to backup to another HD and, even better, backup to a RAID system (you can get one for \$600)

IT378 - M. Di Pierro @ DePaul CTI

## Firewall: Installing Zone Alarm

A good free personal windows firewall is **ZoneAlarm**



IT378 - M. Di Pierro @ DePaul CTI

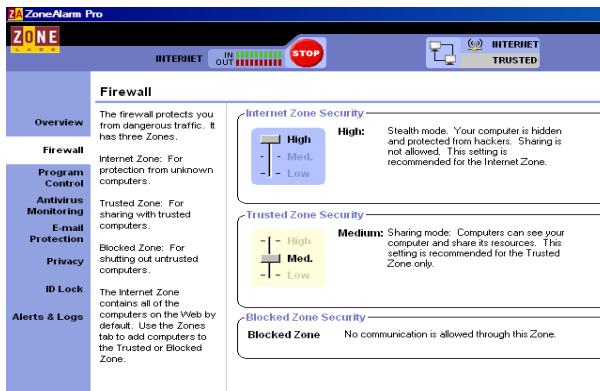
## Firewall: General configuration

You can set some default configurations  
(Deny all connections/Allow all connections/etc.)

ZoneAlarm distinguishes two zones:

- Trusted Zone**  
(set of IP address of computers in your company)
- Internet**  
(all other IP addresses)

You can set different rules for different zones.



IT378 - M. Di Pierro @ DePaul CTI

**Topic:**



**slackware**  
l i n u x

# Introduction to Linux




IT378 - M. Di Pierro @ DePaul CTI

**Linux and GNU**



Manipulated from the wikipedia:

The term **Linux** strictly refers to the Linux kernel developed by Linus Torvalds in 1991, but is commonly used to describe entire Unix-like operating systems (also known as **GNU/Linux**) that are based on the Linux kernel combined with libraries and tools from the GNU project.

It was originally developed for Intel 386 microprocessors and now supports a variety of computer architectures. It is deployed in applications ranging from personal computers to supercomputers and embedded systems such as mobile phones and personal video recorders.




IT378 - M. Di Pierro @ DePaul CTI

## Linux and GNU



**GNU** is a recursive acronym for "GNU's Not Unix". The **GNU project** was launched in 1983 by Richard Stallman with the goal of creating a complete operating system -- called the **GNU system** or simply **GNU** -- that is free software, meaning that users are allowed to copy, modify and redistribute it. The GNU project is now carried out under the auspices of the Free Software Foundation (FSF)

The most important part of GNU is the **GNU Compiler Collection (GCC)**



IT378 - M. Di Pierro @ DePaul CTI

## GNU Public License



Two of the most common free software licenses are the **BSD** and **GPL** licenses.

Traditionally, Linux associated software is licensed under the **GPL**

The GPL grants the recipients of a computer program the following rights:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and modify it.  
(Access to the source code is a precondition for this)
- The freedom to redistribute copies.
- The freedom to improve the program, and release the improvements to the public. (Access to the source code is a precondition for this)

In contrast, the end-user licenses that come with proprietary software **rarely grant the end-user any rights**, and even attempt to restrict activities normally permitted by law, such as reverse engineering. (license written by Richard Stallman)

IT378 - M. Di Pierro @ DePaul CTI

## GNU Public License



The primary difference between the GPL and more "permissive" free software licenses such as the BSD License is that the GPL seeks to ensure that the above freedoms are preserved in copies and in derivative works. It does this using a legal mechanism known as **copyleft**, invented by Stallman, which requires derivative works of GPL-licensed programs to also be licensed under the GPL. In contrast, BSD-style licenses allow for derivative works to be redistributed as proprietary software.

The **BSD license** essentially allows the user of source code released under this license to be free to do whatever that user wishes to do with the code, with very few restrictions. This means that code released under this license can be used in both open source *and* closed source situations.

IT378 - M. Di Pierro @ DePaul CTI

## GNU/Linux: Components and Applications



Linux

**Kernel**

**Posix** (API for threads and sockets/networking)

**XFree86** (API for Client/Server X applications)

**Other libraries:** **Qt, etc.**

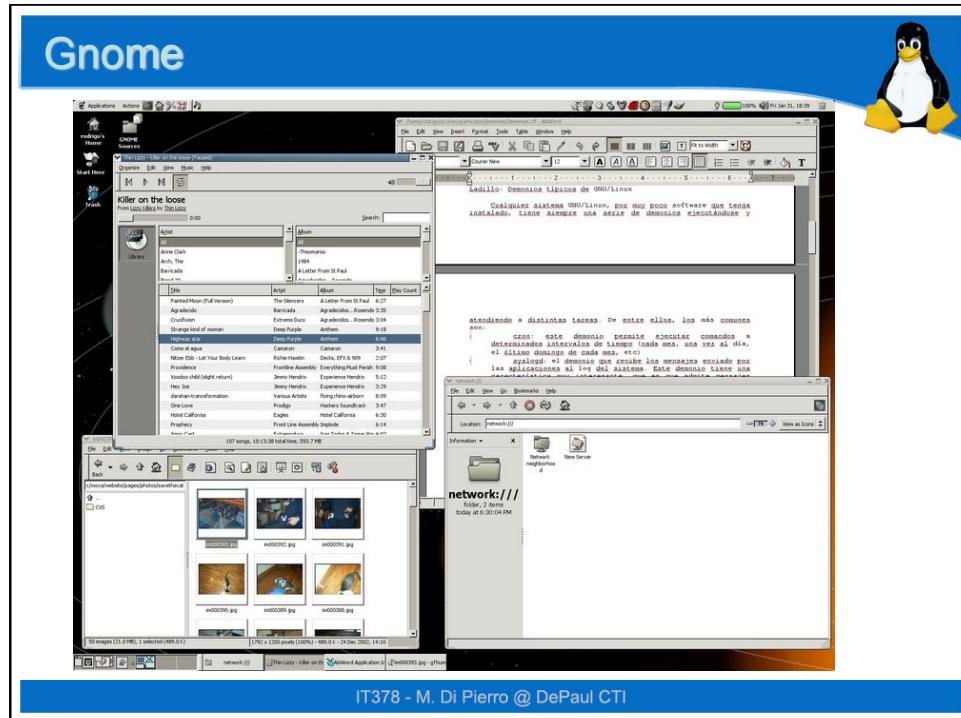
**Window Managers:** **Gnome, KDE, etc.**

**Shells:** **sh, bash, tcsh, ksh, etc.**

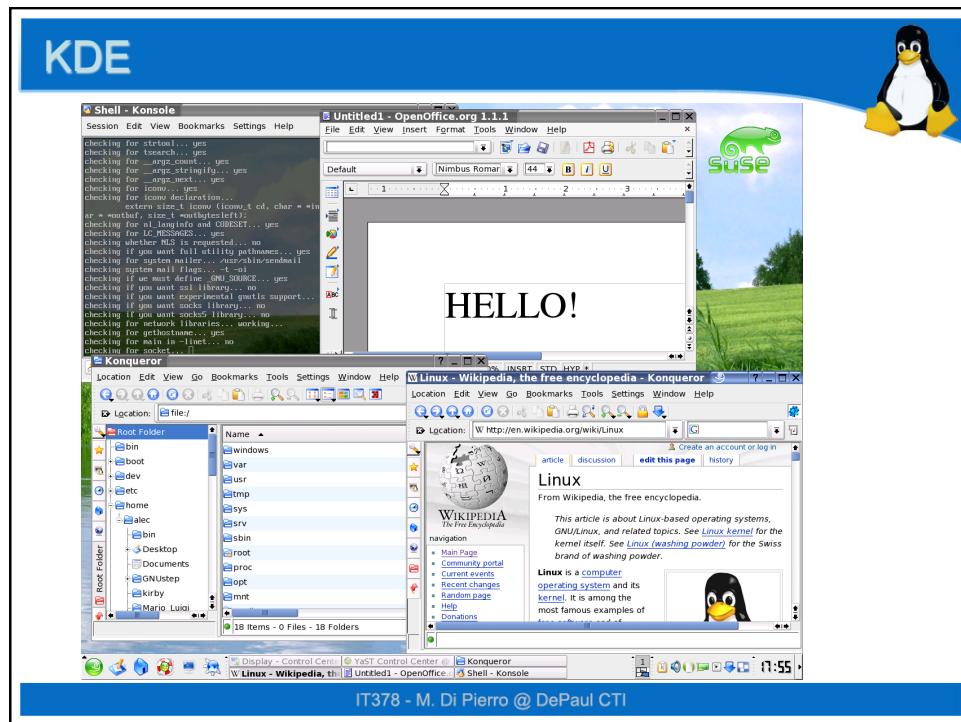
**Tools:** **make, gawk, grep, etc.**

**Applications:** **Gimp, Mozilla, KOffice, OpenOffice, Apache, etc.**

IT378 - M. Di Pierro @ DePaul CTI



IT378 - M. Di Pierro @ DePaul CTI



IT378 - M. Di Pierro @ DePaul CTI



## GNU/Linux: Distributions

**RedHat** Founded in 1993, the company now has more than 700 employees and 22 locations worldwide. They sell a business version of Linux, which is called Red Hat Enterprise Linux and a free version, Fedora Linux, which is an RPM-based Linux distribution, based on GPL license. RedHat also makes cygwin.

**S.u.S.E** was founded in late 1992 in Germany as a UNIX consulting group, which among other things regularly released software packages that included SLS and Slackware, and printed UNIX/Linux manuals. SuSE is now owned by Novell.

**Slackware** is a Linux distribution with a policy of incorporating only stable applications, and the absence of distribution-specific GUI configuration tools found in other varieties of Linux.

IT378 - M. Di Pierro @ DePaul CTI

## GNU/Linux: Distributions



**Gentoo Linux** is a GNU/Linux distribution designed to be modular, portable and optimized for the user's machine. This is accomplished by building all tools and utilities from source code, although, for convenience, several large software packages are also available as precompiled binaries for various architectures.

**Debian** is known for its adherence to the Unix and free software philosophies. It is also known for its abundance of options: the current release contains over eight and a half thousand software packages for many computer architectures, ranging from the ARM architecture commonly found in embedded systems and the IBM s390 mainframe architecture to the more common x86 and PowerPC architectures found in modern personal computers.

**Knoppix** is based on Debian but allows boot from CD. The good: you can try it without installing it, you can use it as a recovery tool. The bad: if you do not install it is slow and you cannot install pages.

IT378 - M. Di Pierro @ DePaul CTI

## GNU/Linux: Kernels



### 2.6 vs 2.4:

- Added support for embedded processors for handheld and other miniature devices
- Increasing scalability: 4095 vs 255 devices, 64 vs 16 GB memory
- Support for even more processors, NUMA, Hyper-Threading
- Changes to scheduling, security, networking, etc, etc.
- SECURITY:
  - Native IPSec
  - Modular security mechanism for greater level of granularity

### SELinux - National Security Agency

Mandatory Access Control:

- Protection decisions must not be decided by the object owner.
- The system must enforce the protection decisions  
(i.e., the system enforces the security policy over the wishes of object owner)

<http://selinux.sourceforge.net/>

IT378 - M. Di Pierro @ DePaul CTI

<h2>Important Basic Commands</h2>	
<code>bash</code>	Start bash shell (we'll use it, same as cygwin)
<code>pwd</code>	Prints your directory
<code>mkdir dirname</code>	Make directory dirname
<code>cd dirname</code>	Move into directory dirname ("cd .. " to move up)
<code>touch filename</code>	Create empty filename or update date if exists
<code>ls -l filename</code>	List directory content
<code>cp filename filename1</code>	Copy filename into filename1
<code>my filename filename1</code>	Rename filename as filename2
<code>rm filename</code>	Delete filename (attention, there is no recycle bin)
<code>less filename</code>	Dump a file to the screen (enter=nextpage, b=back q=quit)
<code>nano filename</code>	Console editor
<code>xemacs filename</code>	X windows editor (only use if you know it already)
<code>ps aux</code>	List all processes running under this shell
<code>ps -aux</code>	List all processes running
<code>pstree</code>	List processes
<code>kill -9 pid</code>	Kills process with process id == pid
<code>top</code>	List processes who are taking most CPU time
<code>man command</code>	Help page about command
<code>which command</code>	Tells you full path of command

<h2>Important Basic Commands</h2>	
<pre>&gt; ls -l /sbin/ -rwxr-xr-x 1 root root lrwxrwxrwx 1 root root -rwsr-xr-x 1 root root</pre>	 permissions owner group size last modified filenames

IT378 - M. Di Pierro @ DePaul CTI

## Important Basic Commands



```
> ls -l /sbin/
-rwxr-xr-x 1 root root 18856 Mar 21 2002 fsck
lrwxrwxrwx 1 root root 6 Feb 9 2004 ksyms -> insmod
-rwsr-xr-x 1 root root 14508 Jan 21 2002 unix_chkpwd
```

**permissions**

**symbolic link:**  
this is not a regular file but a link to a file located somewhere else

**Examples:**

- wx----- owner can write/read/execute
- wrx--- group members can write/read/execute
- wx everybody can write/read/execute
- |----- it is not a file, it is symbolic link
- d----- it is not a file, it is a directory
- s---- it a special file called SUID

**SUID** files run under credentials of the owner and not under credential of the user executing it. A buffer overflow vulnerability in such file may allow user to become root.

IT378 - M. Di Pierro @ DePaul CTI

## Important Basic Commands



```
echo 'hello world' > filename
grep 'hello' < filename
echo 'hello world' | grep 'hello'
```

**Examples:**

<code>ls &gt; mydir.txt</code>	list directory and store it in a file
<code>ls -l   grep 'test'</code>	list all files in the directory that contain 'test' in the filename
<code>ps aux   grep 'root'</code>	list all processes running as "root"

Redirect output of first command (echo 'hello world') to filename  
 Redirect filename to input of the first command (grep 'hello')  
 Redirect the output of the first command to the input of the second command

IT378 - M. Di Pierro @ DePaul CTI

## Linux Directory Structure



```
> ls /
bin etc home lib mnt proc sbin usr
boot dev floppy lost+found opt root tmp var
```

**dev** in Linux/Unix everything is a file. A device is a file. /dev/ contains files that correspond possible devices supported by the Operating System

**lib** contains shared system object libraries required by system programs (example: libc.so, .so is the Linux equivalent of Windows .dll)

**sbin** contains binary programs (executable) for system administration (example: shutdown, turns off the system)

**bin** critical programs (sort of equivalent to /WINDOWS/)

**etc** system configuration files  
(example: passwd, contains system account/password info)

IT378 - M. Di Pierro @ DePaul CTI

## Linux Directory Structure



```
> ls /
bin etc home lib mnt proc sbin usr
boot dev floppy lost+found opt root tmp var
```

**proc** interface to Kernel data structures  
(Example: cpufreqinfo, used to identify the CPU type and speed)

**tmp** temporary storage should go here

**var** should contain files that may have variable size such as log files

**root** home directory of user root

**home** where home directories for other users are stored

**usr** where applications are stored (sort of equivalent to Windows' Program Files)

IT378 - M. Di Pierro @ DePaul CTI

## Linux Directory Structure



```
> ls /
bin etc home lib mnt proc sbin usr
boot dev lost+found root tmp var
```

**mnt** when a device is "connected" and it is "seen" by the OS, we say the device is mounted. Once a device is mounted it appears as a file one can read and write to. This is where such files are usually located.

**lost+found** when system cleans up the file system and finds broken files, they are stored in here.

IT378 - M. Di Pierro @ DePaul CTI

## Other Important Commands: md5sum



`md5sum filename` generates or checks MD5 checksums of filename

**MD5** is a hash function, i.e. a function that is difficult to invert.

If two files have the same MD5 checksum, it is likely the two files are identical.

If you download a file and want to make sure the file was not compromised during the download (for example because a man in the middle attack), compare the MD5 check sum of the file you downloaded with the posted checksum of the file.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Linux



1. You download all software for free, verify integrity of downloads (md5sum)
2. Manage users and file permission
3. Restrict what users can do
4. Force users to use long/complex passwords and change them frequently
5. Restrict hosts who can connect to ours (hosts.allow, hosts.deny)
6. Check SUID/GUID executable
7. Disable root login via ssh
8. Find out which services are running and disable unnecessary services
9. Avoid using su, configure sudo instead.
10. Periodically check for rootkits (ask system to emails you a report every night)
11. Install and configure firewall and intrusion detection software
12. Scan your machine for known vulnerabilities (nmap or nessus)
13. Check your logs periodically
14. Patch your system periodically (apt-get upgrade)
15. Make regular backups

IT378 - M. Di Pierro @ DePaul CTI

## Package Management: configure/make



Red Hat & SuSE	<b>configure/make</b>	rpm
Gentoo	<b>configure/make</b>	portage
Debian	<b>configure/make</b>	apt-get

```

mkdir package
cd package
wget http://somewhere/package.tar.gz
md5sum
gunzip package.tar.gz
tar xvf package.tar
cd package-version
less README
./configure [options]
make all
su
make install

```

create a dir for the package  
 move under the dir  
 download the package  
 verify the md5 hash  
 unzip it  
 untar it  
 move under the dir  
 read instructions  
 configure package  
 compile the package  
 become super user  
 install it

IT378 - M. Di Pierro @ DePaul CTI

## Package Management: rpm



Red Hat & SuSE	configure/make	<b>rpm</b>
Gentoo	configure/make	portage
Debian	configure/make	apt-get

```
mkdir package
cd package
wget http://somewhere/package.rpm
md5sum package.rpm
su
rpm -i package.rpm
```

create a dir for the package  
move under the dir  
download the package  
verify the md5hash  
become root  
install the package

```
rpm -e package
rpm -Vp package.rpm
rpm -Va
```

uninstall the package  
verify installed package  
verify all installed packages  
against DB

The good: rpm checks and notifies about package dependencies

IT378 - M. Di Pierro @ DePaul CTI

## Package Management: apt-get



Red Hat & SuSE	configure/make	<b>rpm</b>
Gentoo	configure/make	portage
Debian	configure/make	<b>apt-get</b>

```
su
apt-get install package
apt-get [-purge] remove
package
apt-get update
apt-get -u upgrade
```

create a dir for the  
package

uninstall the package

updates /etc/apt/  
sources.list

**The good:** apt-get checks package dependencies and automatically installs them too!  
upgrades all packages

List of all Debian packages <http://www.debian.org/distrib/packages>

IT378 - M. Di Pierro @ DePaul CTI

## Log Files (the BOOT log)



To view the boot logs use the command

`dmesg | less` dumps the boot logs

All other log files are in `/var/log` and are configured in `/etc/syslog.conf`  
Typically almost all logs go in `/var/log/messages`

for more info:

<http://www.cert.org/security-improvement/implementations/i041.08.html>

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files (PASSWORDS)



`/etc/passwd`

```
root:x:0:0:root:/bin/bash
mdipierro:x:1001:1001:Massimo DiPierro,,,:/home/mdipierro:/bin/bash
```

**Name: Password: UserID: PrincipleGroup: Name: HomeDirectory: Shell**

Attributes in an entry are separated by a : (colon).

For this reason, you should not use a : (colon) in any attribute.

**Name** Specifies the user's login name. See the **adduser** command for more information

**Password** x if password in /etc/shadow

**UserID** Specifies the user's unique numeric ID.

**PrincipleGroup** Specifies the user's principal group ID.

**Name** real name of user (you may want to conceal it)

**HomeDirectory** Specifies the full path name of the user's home directory.

**Shell** Specifies the initial program or shell that is executed when user logs in.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files (PASSWORDS)



### /etc/shadow

```
root:$1$Nu1p$BlgMBY5up9F5ES.fMmCr1:12478:0:99999:7:::  
mdipierro:$1$D0zHTNplGuu1r7cE4sDI/:12478:0:99999:7:::
```

**Username**, up to 8 characters. A direct match to the username in the /etc/passwd file.

**Password**, 13 character encrypted. A blank entry indicates a password is not required to log in (usually a bad idea), and a `'' entry indicates the account has been disabled.

**The number of days** (since January 1, 1970) since the password was last changed.

**The number of days before password may be changed** (0 = any time)

**The number of days after which password must be changed** (99999 = forever)

**The number of days** to warn user of an expiring password (7 for a full week)

**The number of days** after password expires that account is disabled

**The number of days** since January 1, 1970 that an account has been disabled  
A reserved field for possible future use

IT378 - M. Di Pierro @ DePaul CTI

## Adding a User



```
adduser mdp      add a user
```

Adding user mdp...

Adding new group mdp (1022).

Adding new user mdp (1022) with group mdp.

Creating home directory /home/mdp.

Copying files from /etc/skel

Enter new UNIX password: \*\*\*\*\*

<b>su mdp</b>	become the new user
<b>passwd</b>	change the password

IT378 - M. Di Pierro @ DePaul CTI

## Adding / Managing groups



### **/etc/group**

Contains group information. Edit with **addgroup**, **adduser**, **delgroup**, **deluser**:

```
addgroup mygroup
adduser mdp mygroup
deluser mdp mygroup
delgroup mygroup
deluser mdp
```

create a group  
add a user to group  
delete user from group  
delete group from system  
delete user from system

```
ls -l /sbin/
```

-rwxr-xr-x 1 root root	18856 Mar 21 2002 fsck
lrwxrwxrwx 1 root root	6 Feb 9 2004 ksyms -> insmod
-rwsr-xr-x 1 root root	14508 Jan 21 2002 unix_chkpwd

permissions owner group size last modified filenames

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files (Password Aging)



### **/etc/login.defs**

Contains the following default values for account aging:

**PASS\_MAX\_DAYS 60** Maximum number of days a password is valid.

**PASS\_MIN\_DAYS 7** Minimum number of days before a user can change the password since the last change.

**PASS\_WARN\_AGE 7** Number of days when the password change reminder starts.

```
chage mdp
```

display password aging information for user mdp

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files (PAM)



**/etc/pam.d/passwd**

Configuration for the Pluggable Authentication Modules. Default and no good:

```
password required pam_unix.so nullok obscure min=4 max=8 md5
```

TO DO:

get /lib/security/pam\_cracklib.so and configure it

```
apt-get install libpam_cracklib
apt-get install wenglish
/etc/cron.daily/cracklib
touch /etc/security/opasswd
```

CONTINUE....

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files (PAM)



CONTINUE....

edit **/etc/pam.d/passwd**

**emacs**

```
auth    required pam_env.so
auth    required pam_tally.so onerr=fail no_magic_root
auth    sufficient pam_unix.so likeauth nullok
auth    required pam_deny.so
account required pam_unix.so remember=10
account required pam_tally.so per_user deny=5 no_magic_root reset
account sufficient pam_succeed_if.so uid < 100 quiet
account required pam_permit.so
password requisite pam_cracklib.so retry=3 minlen=6 difffok=3
password sufficient pam_unix.so nullok use_authtok md5 shadow
password required pam_deny.so
session required pam_limits.so
session required pam_unix.so
```

read more here: man faillog, usermod  
[http://www.deer-run.com/~hal/sysadmin/pam\\_cracklib.html](http://www.deer-run.com/~hal/sysadmin/pam_cracklib.html)

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



Given a file you change its owner or group with chown

```
chown mdp:testmdp filename
```

set filename owner to mdp and group to testmdp

and change what users (u=owner, g=group, o=other) can do with it.

```
chmod 750 filename  
chmod u-x filename  
chmod o+r filename  
chmod u+s filename  
chmod u-s filename
```

owner can read+write,execute (4+2+1=7) this file  
members of the group can read+execute (4+1=5)  
other users can do nothing (0)  
revoke owner's right to execute this file  
give every user permission to read this file  
make the file executables as SUID  
remove SUID

Important: SUID means the file when executed runs under credentials of owner and not under credentials of user who is running it.

IT378 - M. Di Pierro @ DePaul CTI

## \$PATH Environment Variable



The you type a command the terminal searches for it in the directory specified in the **\$PATH variable**. You can display and change the \$PATH

```
echo $PATH  
PATH=$PATH:/home/mdp
```

displays the current PATH environment variable  
Add /home/mdp to PATH environment variable

### To execute a local file filename

```
./filename
```

executes filename

reads as 'here'

IT378 - M. Di Pierro @ DePaul CTI

## Adding / Managing User Limitations



Two commands

**ulimit**  
**quota**

tells/set user  
limitations

checks user disk  
usage

Read more:

<http://www.yolinux.com/TUTORIALS/LinuxTutorialQuotas.html>

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



**hostent -a ipaddress -h hostname**  
**hostent -d ipaddress**

add an entry into the host  
table

create a dir for the package

### **/etc/host.conf**

tells the network domain server how to look up hostnames.  
(Normally /etc/hosts, then name server; it can be changed through netconf.)

### **/etc/hosts**

Contains a list of known hosts (in the local network).

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/hosts.allow**

list of services and hosts that have access to them.

Example:

```
sshd: .depaul.edu
httpd: ALL
```

(allow sshd access from depaul.edu domain only and httpd access from everybody)

### **/etc/hosts.deny**

list of services and hosts that do not have access to them (exceptions in hosts.allow)

ALL: ALL

→ we'll see names of services later

(means deny all TCP connection except for hostnames explicitly allowed)

IT378 - M. Di Pierro @ DePaul CTI

## More on Services



Entries in /etc/hosts.equiv and /etc/hosts.lpd are **critical**

They allow users from those hosts to connect without supplying a password.

Also, users can create **.rhosts** and **.netrc** files in their home directories, which function similarly. Find these as well

IT378 - M. Di Pierro @ DePaul CTI

## Su (NO) and sudo (YES)



Permitting login access is dangerous. Disable it

edit the file `/etc/ssh/sshd_config`

1. Find the line: Protocol 2, 1  
Change it to: Protocol 2
2. Find the line: PermitRootLogin yes  
Change it to: PermitRootLogin no
3. Restart SSH deamon: `/etc/init.d/ssh restart`

**Never give the root password.** Use sudo instead: <http://linsec.ca/syshardening/sudo.php>

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Commands



The single most important command is **find**

```
find / -ctime 3 -print
find / -iname *.txt -print
find / -nouser -print
find / -user mdp -perm 700 -print

find / -user mdp -perm 700 -ok rm {} ;
find [where] [for what] [what to do]
```

print all files that where changed less than 3\*24 hours ago  
 print all files that have a .txt extension  
 print all files with an invalid user  
 print all files owned by mdp and that only mdp can read/write/execute

remove all files owned by mdp and that only mdp can read/write/execute

The following command finds all SUID and SGID files on the system

```
find / -type f \( -perm -04000 -o -perm -02000 \| ) -exec ls -l {} \;
```

IT378 - M. Di Pierro @ DePaul CTI

Topic:



# Introduction to Linux Services and Configuration



IT378 - M. Di Pierro @ DePaul CTI

## Linux Runlevels



**Runlevel 0:** Halt System - To shutdown the system

**Runlevel 1:** Single user mode

**Runlevel 2:** Basic multi user mode without NFS

**Runlevel 3:** Full multi user mode (text based)

**Runlevel 4:** unused

**Runlevel 5:** Multi user mode with Graphical User Interface

**Runlevel 6:** Reboot System

Runlevels 1 and 2 are generally used for debugging purposes only, and are not used during normal operations. Most desktop linux distributions boot into runlevel 5, which starts up the Graphical Login Prompt. This allows the user to use the system with X-Windows server enabled. Most servers boot into runlevel 3, which starts the text based login prompt.

Linux runlevels can be changed on the fly using the init tool.

If you want to switch from text based operations to the Graphical Interface, you just have to type in `telinit 5` in the root prompt. This will bring up the Graphical Interface in your system.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Runlevels



Each runlevel can be configured by the system administrator. The **/etc/inittab** file has information on which runlevel to start the system at and lists the processes to be run at each runlevel.

Each runlevel has its own directory structure where you can define the order in which the services start. These directories are located in the **/etc/rc.d/** directory, under which you have rc1.d, rc2.d, rc3.d.... rc6.d directories where the number from 0 through 6 that corresponds to the runlevel. Inside each directory are symbolic links that point to master initscripts found in **/etc/init.d** or **/etc/rc.d/init.d**.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



Services are started by **/etc/rc.d** scripts and Xinetd services are configured by individual files in **/etc/xinetd.d/**

```
>netstat -antp
>> pstree
init+-apache---5*[apache]
|-atd
|-cron
|-dhclient-2.2.x
|-6*[getty]
|-keventd
|-kflushd
|-klogd
|-kswapd
|-kupdate
|-miniserv.pl
|-safe_mysqld---mysqld---mysqld---2*[mysqld]
|-sendmail
|-sshd---sshd---sshd---bash---bash---pstree
`-syslogd
```

tells you about open ports  
tells you about running programs

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



`/etc/init.d/apache start`  
`/etc/init.d/apache stop`  
`/etc/init.d/apache restart`

Debian only

`update-rc.d apache default`  
`update-rc.d -f apache remove`

Red Hat only

`chkconfig apache on`  
`chkconfig apache off`

starts apache now  
stops apache now  
stops and starts apache now

adds apache to rc  
removes apache from rc

adds apache to rc  
removes apache from rc

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**amd**  
Runs the automount daemon for remote filesystem mounting such as nfs.

**apmd**  
Monitors battery status and can shut down the system if power is low.

**arpwatch**  
Keeps track of ethernet IP address pairings what are resolved using the ARP protocol. This allows system administrators to note new IP addresses being used. It maintains a database in /var/arpwatch/arp.dat.

**atd**  
Runs commands scheduled by the "at" program at their scheduled times. Jobs are stored in /var/spool/at

**autofs**  
Also called the automount daemon, it is used to automatically mount filesystems on demand. It is especially worthwhile for working with removable media such as floppies or CD ROM disks.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**crond**  
A daeman that executes scheduled commands according to the /etc/crontab file. It can be used to clean up temporary files in /tmp and /var/tmp and other places.

**dhcpd**  
Provides DHCP services to "lease" out IP addresses to remote

**gpm**  
Provides mouse support to Linux.

**httpd**  
The Apache hypertext transfer protocol Web server.

**identd**  
Server implementing the TCP/IP proposed standard IDENT user identification protocol in RFC 1413. It returns user information to a remote host that a user is requesting a service from. Also called auth.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**inet**  
The internet super daemon (inetd) that provides all the services specified in /etc/inetd.conf.

**innd**  
The Usenet news server

**ldap**  
Lightweight directory access protocol package which provides client/server based directory database services which runs on UNIX platforms over TCP/IP. RFC is 1777.

**lpd**  
Provides printing services to Linux. It is a print spooler daemon.

**named and bind**  
Provide DNS services. It is a name server used to translate local IP addresses to names and vice-versa.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**netfs**  
Mounts and unmounts Network File System (NFS), Windows (SMB), and Netware (NCP) file systems. The mount command is used to perform this operation and no daemon is run in the background.

**network**  
Brings up all the network interfaces under the directory /etc/sysconfig/network-scripts. Also controls IP\_forwarding and IP\_defrag.

**nfs**  
Provides Network File System server services

**nfslock**  
NFS file locking service. Starts the daemons rpc.lockd and rpc.statd. The rpc.statd daemon implements the Network Status Monitor (NSM) RPC protocol which is a reboot notification service used to implement file lock recovery when an NFS server crashes and reboots.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**pcmcia**  
Provides access to PCMCIA (PC Cards) services configured in the /etc/exports file.

**portmap**  
Provides Remote Procedure Call (RPC) support for other protocols like NFS.

**postgresql**  
Runs the postgres database and provides SQL services. It runs the daemon postmaster.

**pxe**  
A preboot execution environment (PXE) server. Allows network boot of other PXE machines. PXE is a standard developed by Intel for a means of booting various operating systems on remote machines.

**random**  
Saves and restores a random value used to generate better random numbers for security. No daemon program is invoked for this operation.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**routed**  
Provides for automatic router table updates using the RIP dynamic routing information protocol.

**rstatd**  
The rstat protocol allows users on a network to get performance information for any machine on the network. Runs the rpc.rstdt daemon which provides performance statistics retrieved from the kernel usually by using the "rup" command.

**sendmail**  
The sendmail mail transport agent daemon used to move e-mail from one machine to another.

**smb**  
Provides SMB (Samba) client/server services which include file and print services. It allows Linux computers to exchange file and printer services with Microsoft Windows based systems.

**snmpd**  
Provides Simple Network Management Protocol support to Linux

IT378 - M. Di Pierro @ DePaul CTI

## Linux Services



**squid**  
Runs the squid proxy web server

**sshd**  
provides ssh server connectivity

**syslog**  
System logging daemon which records system events to log files usually in the directory "/var/log". The actual name of the daemon is klogd.xfs X font file server

**xntpd**  
Starts the Network Time Protocol NTPv3 daemon.

**ypbind**  
Binds YP/NIS clients to a yellow pages server. NIS (Network Information Service) is a name service created by Sun.yppasswddAllows users to change their passwords on systems running YP/NIS

**ypserv**  
This daemon provides the YP/NIS (Network Information System) server functions.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/rc.d/rc**

Normally run for all run levels with level passed as argument. For example, to boot your machine in the Graphics mode (X-Server), run the following command from your command line: init 5. The runlevel 5 is starts the system in graphics mode.

### **/etc/rcX.d/**

Scripts run from rc (X stands for any number from 1 to 5). These directories are "run-level" specific directories. When a system starts up, it identifies the run-level to be initiated, and then it calls all the startup scripts present in the specific directory for that run-level. For example, the system usually starts up and the message "entering run-level 3" is shown after the boot messages; this means that all the init scripts in the directory /etc/rc.d/rc3.d/ will be called.

### **/etc/mtab**

This changes continuously as the file /proc/mount changes. In other words, when filesystems are mounted and unmounted, the change is immediately reflected in this file.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/fstab**

Lists the filesystems currently "mountable" by the computer. This is important because when the computer boots, it runs the command mount -a, which takes care of mounting every file system marked with a "1" in the next-to-last column of fstab.

### **/etc/group**

Contains the valid group names and the users included in the specified groups. A single user can be present in more than one group if he performs multiple tasks. For example, if a "user" is the administrator as well as a member of the project group "project 1", then his entry in the group file will look like: user: \* : group-id : project1

### **/etc/nologin**

If the file /etc/nologin exists, login(1) will allow access only to root. Other users will be shown the contents of this file and their logins refused.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/rpmrc**

rpm command configuration. All the rpm command line options can be set together in this file so that all of the options apply globally when any rpm command is run on that system.

### **/etc/shells**

Holds the list of possible "shells" available to the system.

### **/etc/motd**

Message Of The Day; used if an administrator wants to convey some message to all the users of a Linux server.

### **/etc/networks**

Lists names and addresses of networks accessible from the network to which the machine is connected. Used by route command. Allows use of name for network.

### **/etc/protocols**

Lists the currently available protocols.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/resolv.conf**

Tells the kernel which name server should be queried when a program asks to "resolve" an IP Address.

### **/etc/rpc**

Contains instructions/rules for RPC, which can be used in NFS calls, remote file system mounting, etc.

### **/etc/exports**

The file system to be exported (NFS) and permissions for it.

### **/etc/services**

Translates network service names to port number/protocol. Read by inetd, telnet, tcpdump, and some other programs. There are C access routines.

### **/etc/inetd.conf**

Config file for inetd. Holds an entry for each network service for which inetd must control daemons or other services. Note that services will be running, but comment them out in /etc/services so they will not be available even if running.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/sendmail.cf**

The Mail program sendmail's configuration file.

### **/etc/lilo.conf**

Contains the system's default boot command line parameters and also the different images to boot with. You can see this list by pressing Tab at the LILO prompt.

### **/etc/logrotate.conf**

Maintains the log files present in the /var/log directory.

### **/etc/identd.conf**

Identd is a server that implements the TCP/IP proposed standard IDENT user identification protocol as specified in the RFC 1413 document. identd operates by looking up specific TCP/IP connections and returning the user name of the process owning the connection. It can optionally return other information instead of a user name. See the identd man page.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/inittab**

This is chronologically the first configuration file in UNIX. The first program launched after a UNIX machine is switched on is init, which knows what to launch, thanks to inittab. It is read by init at run level changes, and controls the startup of the main process.

### **/etc/syslogd.conf**

The configuration file for the syslogd daemon. syslogd is the daemon that takes care of logging (writing to disk) messages coming from other programs to the system. This service, in particular, is used by daemons that would not otherwise have any means of signaling the presence of possible problems or sending messages to users.

### **/etc/httpd.conf**

The configuration file for Apache, the Web server. This file is typically not in /etc. It may be in /usr/local/httpd/conf/ or /etc/httpd/conf/, but to make sure, you need to check the particular Apache installation.

IT378 - M. Di Pierro @ DePaul CTI

## Linux Important Files



### **/etc/conf.modules or /etc/modules.conf**

The configuration file for kerneld. This is not the kernel "as a daemon". It is rather a daemon that takes care of loading additional kernel modules "on the fly" when needed.

IT378 - M. Di Pierro @ DePaul CTI

## Your Security Tools



### **Bastille**

```
apt-get install bastille
```

Performs most of the security configurations above using a GUI and step-by-step explanations

<http://www.sans.org/rr/whitepapers/linux/195.php>

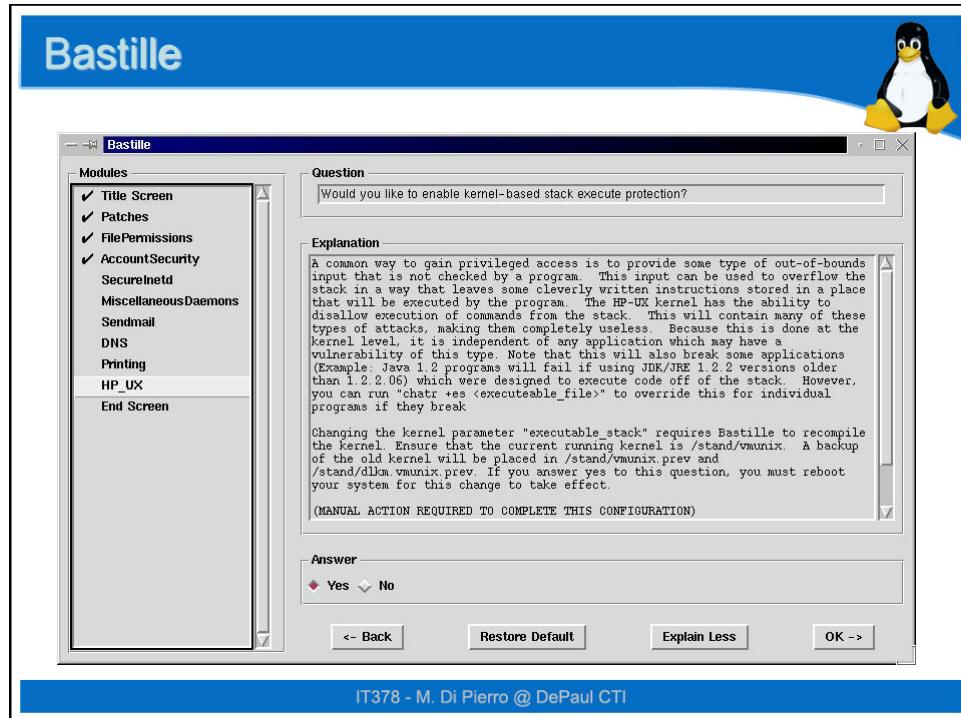
### **chkrootkit**

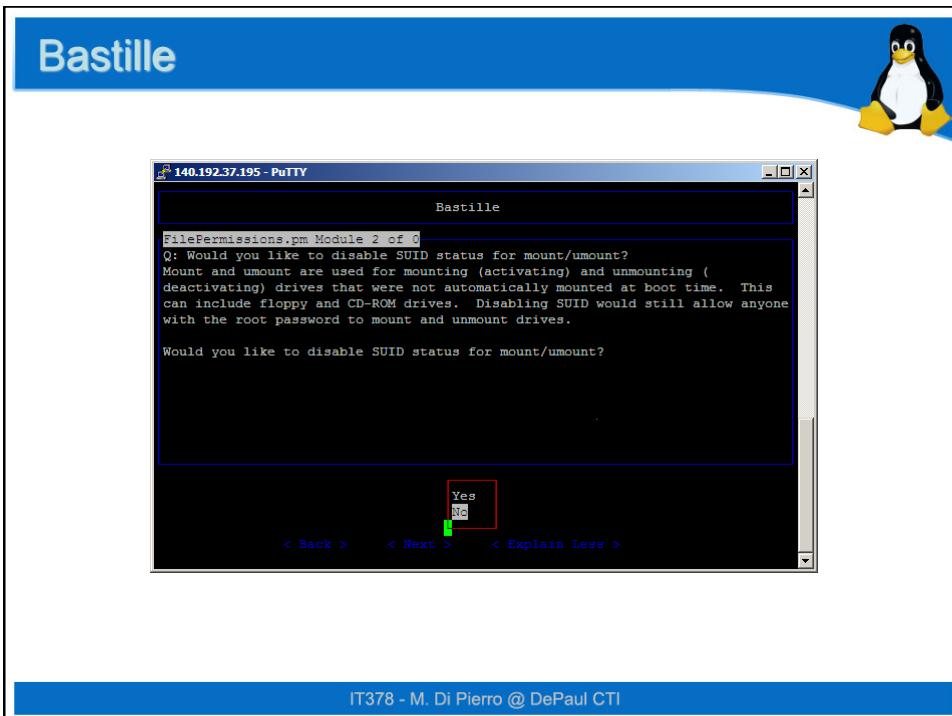
```
apt-get install chkrootkit
```

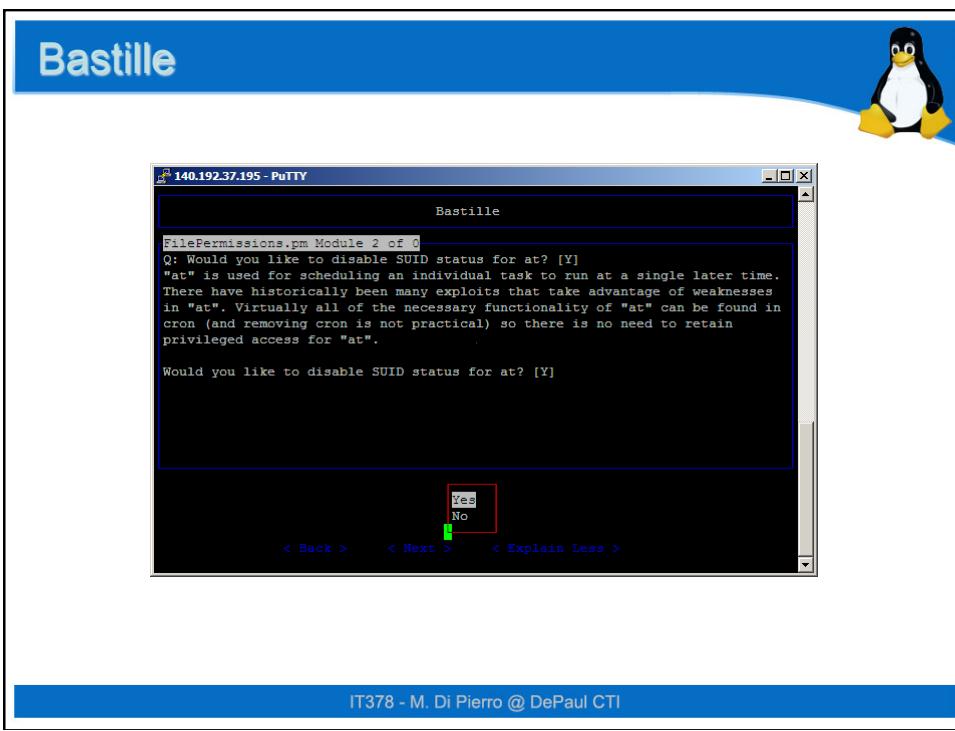
Checks for the anomalies such as known viruses/worms/trojans/backdoors and suspicious log entries.

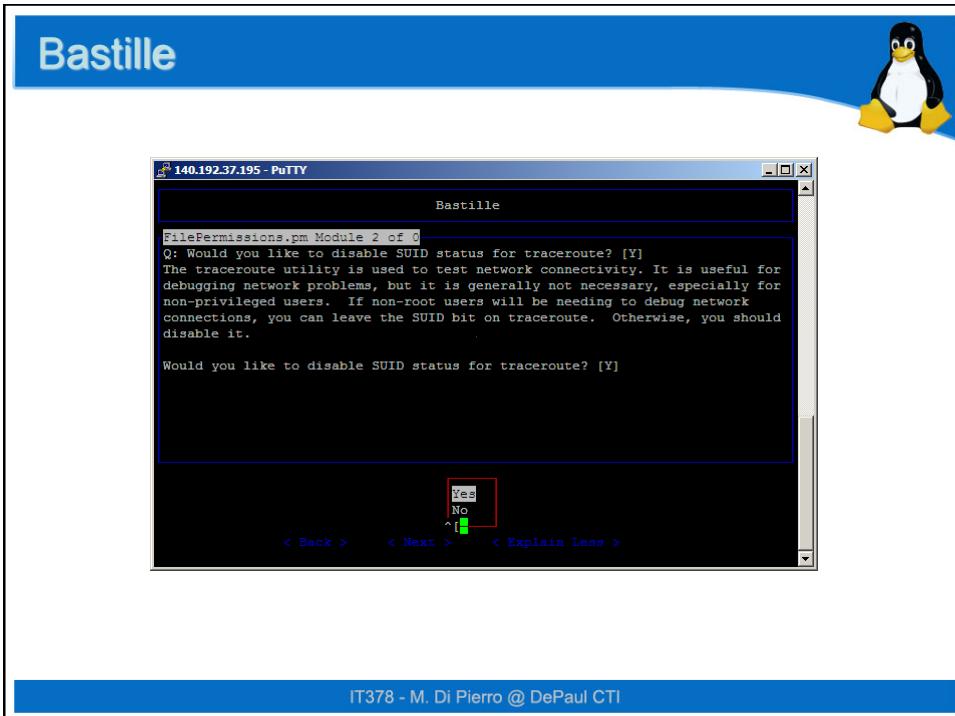
<http://www.chkrootkit.org>

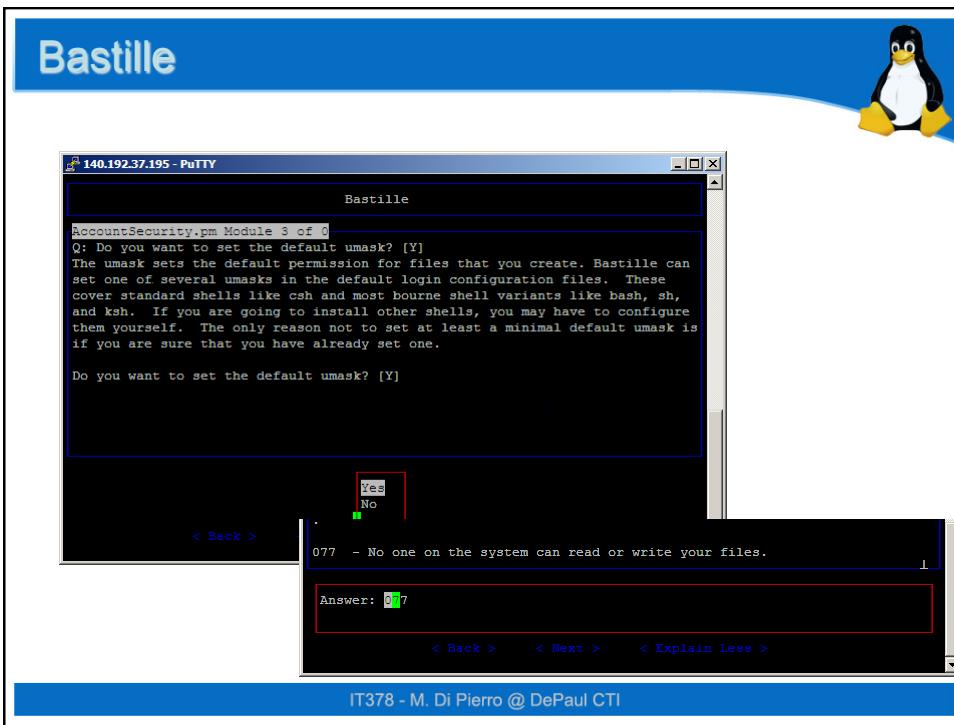
IT378 - M. Di Pierro @ DePaul CTI

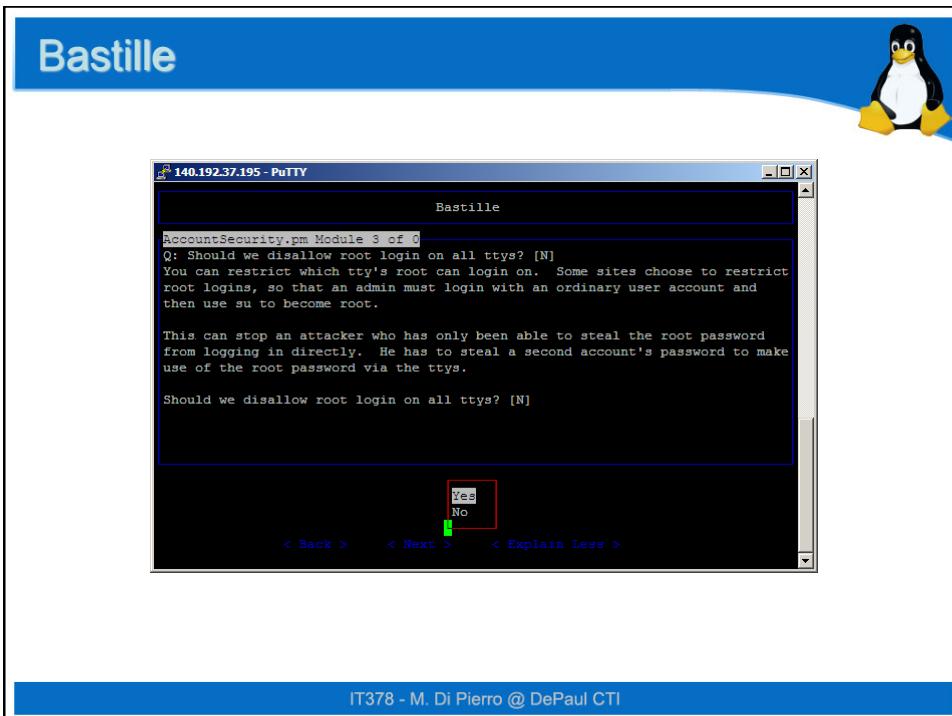


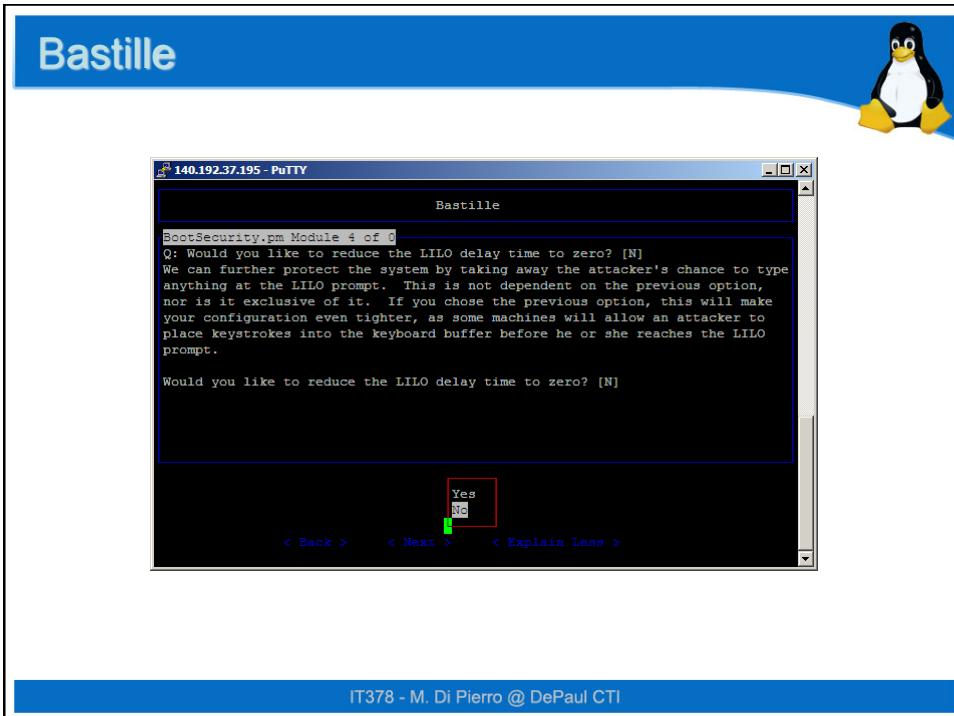


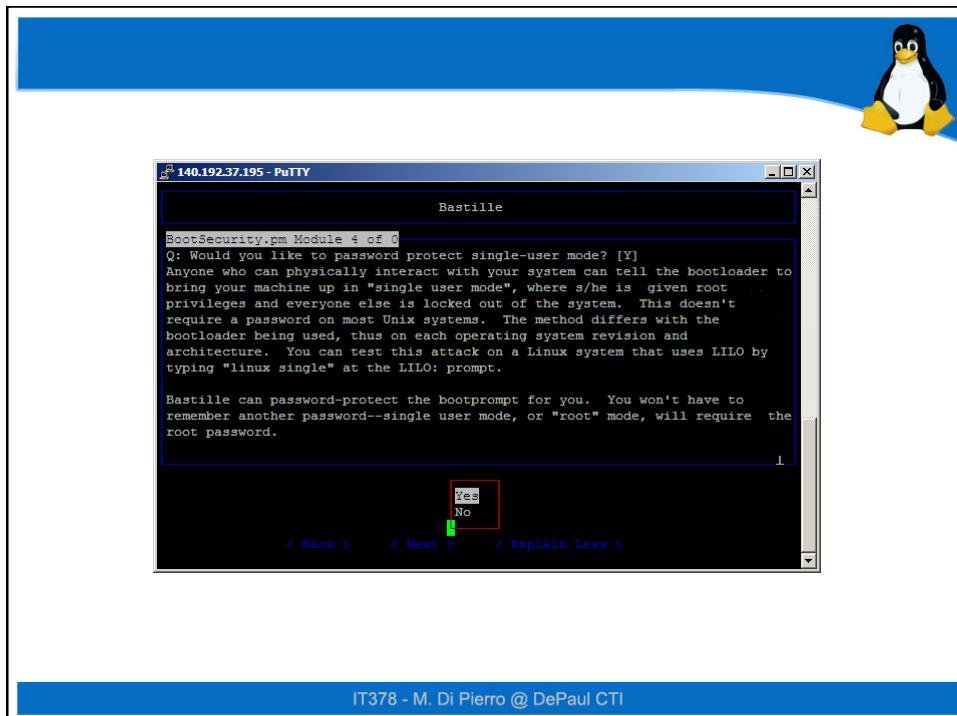


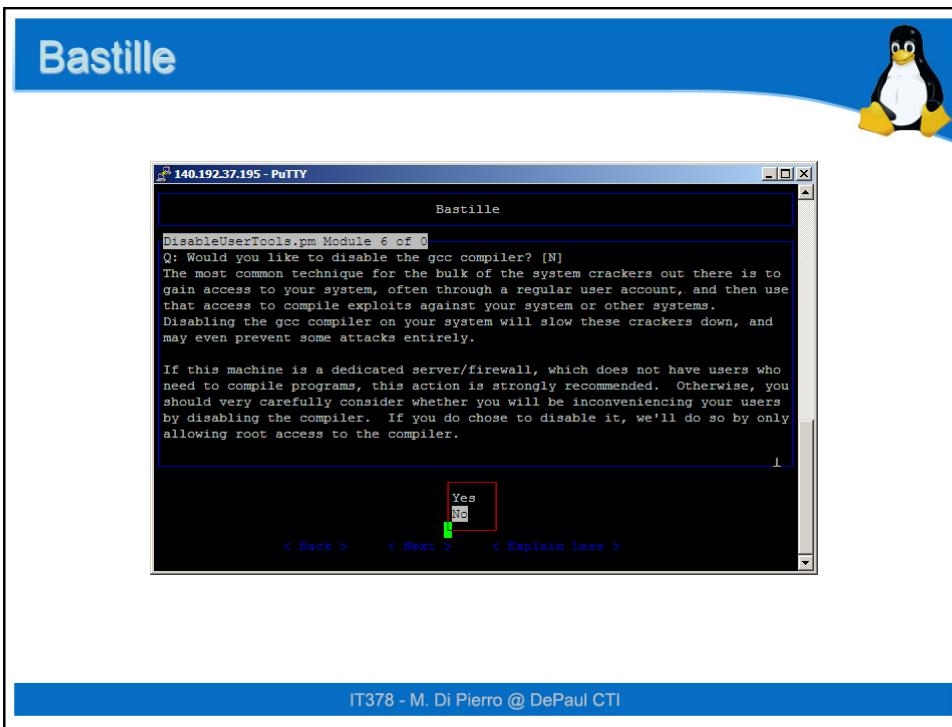


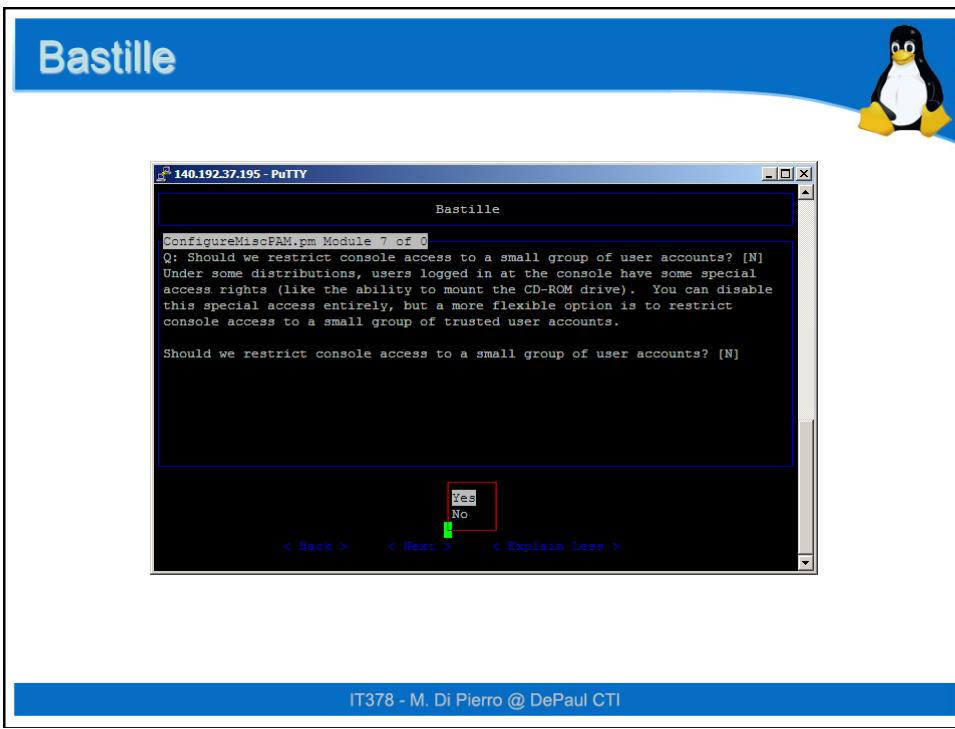








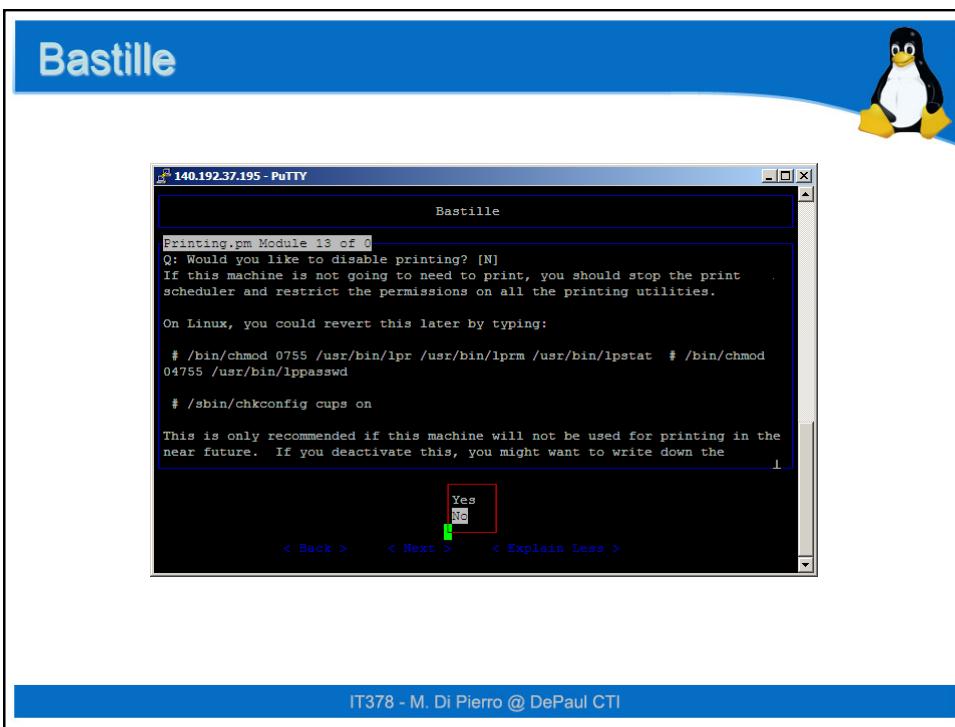
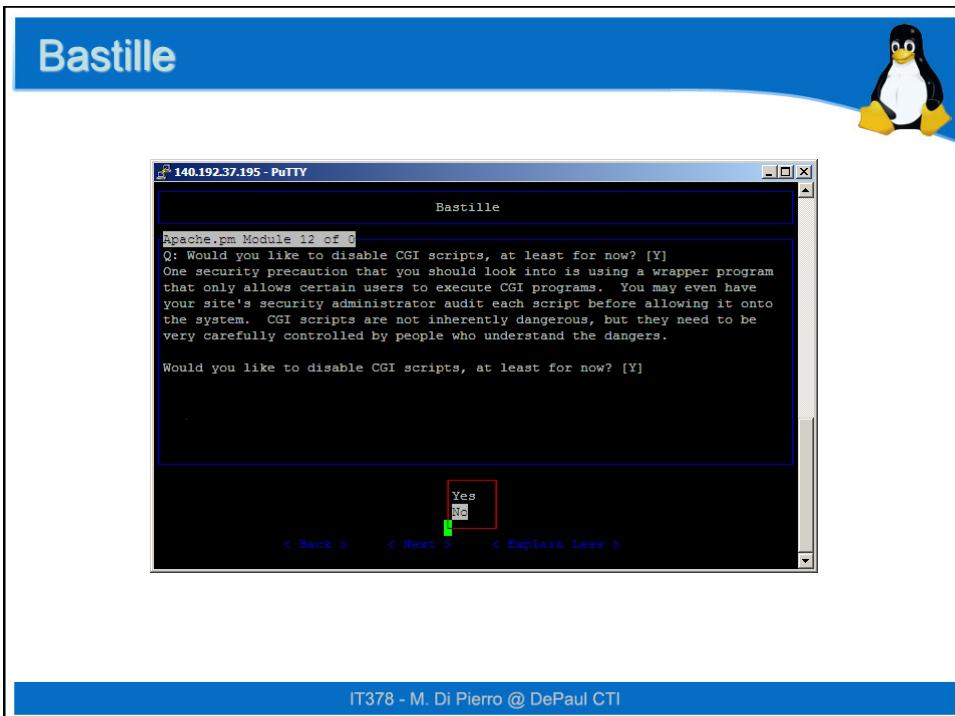


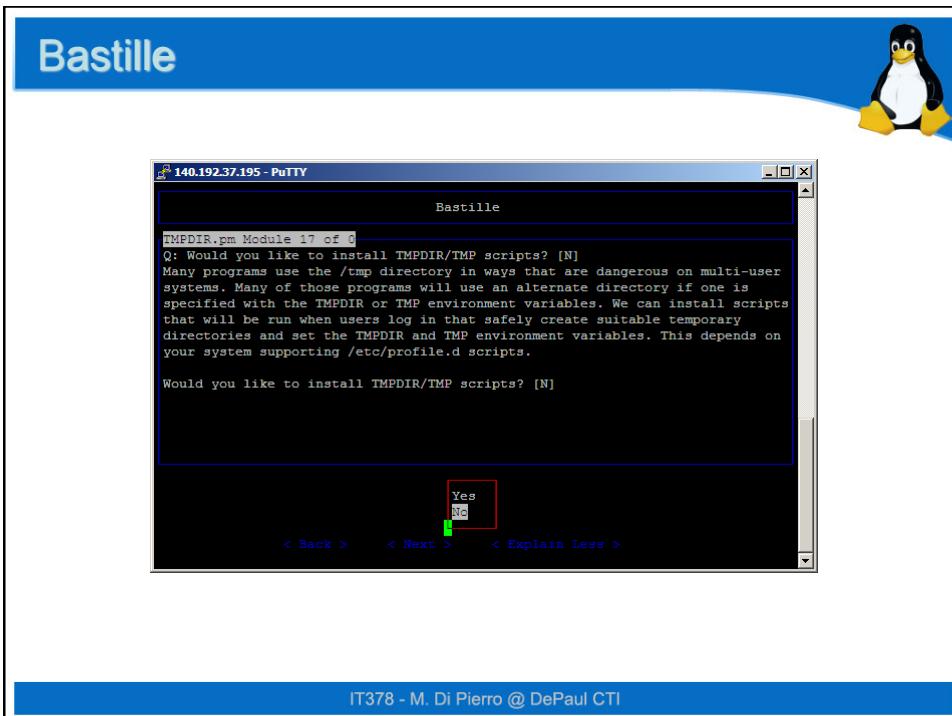












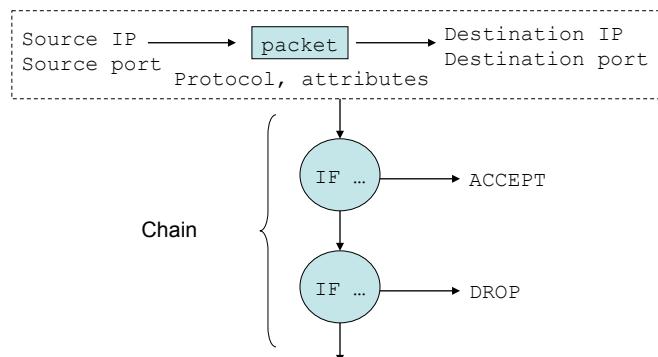
Topic:



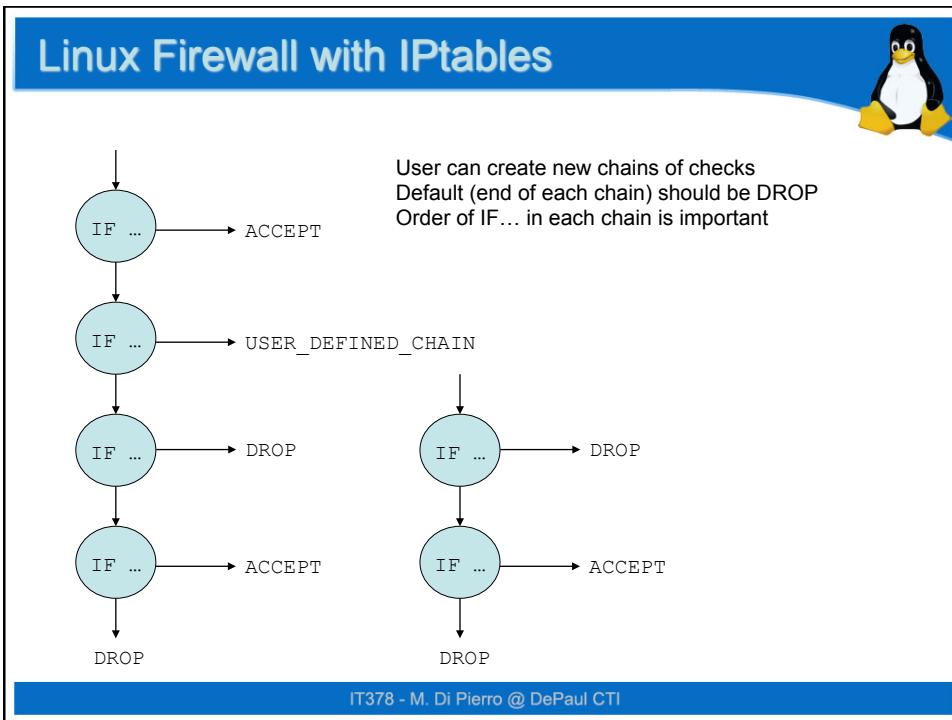
# Linux firewall: iptables

IT378 - M. Di Pierro @ DePaul CTI

Linux Firewall with IPtables



IT378 - M. Di Pierro @ DePaul CTI



## Linux Firewall with IPtables

```
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes 64 bytes from 127.0.0.1:
icmp_seq=0 ttl=64 time=0.2 ms --- 127.0.0.1
ping statistics --- 1 packets transmitted, 1 packets received,
0% packet loss round-trip min/avg/max = 0.2/0.2/0.2 ms

# iptables -A INPUT -s 127.0.0.1 -p icmp -j DROP

# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes --- 127.0.0.1
ping statistics --- 1 packets transmitted, 0 packets received,
100% packet loss
#
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPtables



There are several different things you can do with **iptables**. First the operations to manage whole chains. You start with three built-in chains input, output and forward which you can't delete.

- **Create a new chain** (-N).
- **Delete an empty chain** (-X).
- **Change the policy for a built-in chain**. (-P).
- **List the rules in a chain** (-L).
- **Flush the rules out of a chain** (-F).
- **Zero the packet and byte counters on all rules in a chain** (-Z).

There are several ways to manipulate rules inside a chain:

- **Append a new rule to a chain** (-A).
- **Insert a new rule at some position in a chain** (-I).
- **Replace a rule at some position in a chain** (-R).
- **Delete a rule at some position in a chain** (-D).
- **Delete the first rule that matches in a chain** (-D).

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPtables



Important files:

**/etc/sysconfig/iptables-precursor**

Where you would put the iptable commands you want to be executed at startup

**/etc/sysconfig/iptables**

Automatically generated from the file iptable-precursor and input for the daemon

```
/etc/rc.d/init.d/iptables stop
source /etc/sysconfig/iptables-precursor
iptables-save > /etc/sysconfig/iptables
/etc/init.d/iptables restart
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPTables



Example of file */etc/sysconfig/iptables-precursor*

```
emacs
# Configure default policies (-P)

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# Flush (-F) all specific rules

iptables -F INPUT
iptables -F FORWARD
iptables -F OUTPUT
iptables -F -t nat
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPTables



Example of file */etc/sysconfig/iptables-precursor* (CONTINUED)

```
emacs
# Allow output of HTTP, HTTPS and SSH

iptables -A OUTPUT -p tcp -s ME -d 0/0 --dport 80 --syn -j
ACCEPT
iptables -A OUTPUT -p tcp -s ME -d 0/0 --dport 443 --syn -j
ACCEPT

iptables -A OUTPUT -p tcp -s ME -d 0/0 --dport 22 --syn -j
ACCEPT

# Permit packets in and out of firewall if they are part of
existing and related connections.

iptables -A OUPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPtables



Example of file */etc/sysconfig/iptables-precursor* (CONTINUED)

**emacs**

```
# If running a web server allow in HTTP, HTTPS
iptables -A INPUT -p tcp -s 0/0 -d ME --dport 80 --syn -j ACCEPT
iptables -A INPUT -p tcp -s 0/0 -d ME --dport 443 --syn -j ACCEPT

# if running SSH deamon
iptables -A INPUT -p tcp -s 0/0 -d ME --dport 22 --syn -j ACCEPT
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPtables



Example of file */etc/sysconfig/iptables-precursor* (CONTINUED)

**emacs**

```
# Deny any packet coming in on the public internet interface
eth0 which has a spoofed source address from our local
networks:
iptables -A INPUT -i eth0 -s ME -j DROP
iptables -A INPUT -i eth0 -s 127.0.0.1 -j DROP

iptables -A OUTPUT -i eth0 -d ME -j DROP
iptables -A OUPUT -i eth0 -d 127.0.0.1 -j DROP
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPTables



Example of file */etc/sysconfig/iptables-precursor* (CONTINUED)

**emacs**

```
# Forward all packets from eth1 (internal network) to eth0
# (the internet).

iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

# Forward packets that are part of existing and related
# connections from eth0 to eth1.

iptables -A FORWARD -i eth0 -o eth1 -m state --state
ESTABLISHED,RELATED -j ACCEPT

#allow input from inside network

iptables -A INPUT -i eth1 -s 0/0 -d 0/0 -j ACCEPT
iptables -A INPUT -i lo -s 0/0 -d 0/0 -j ACCEPT
```

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPTables



### Warnings:

Before any iptables commands have been run (be careful: some distributions will run iptables in their initialization scripts), there will be no rules in any of the built-in chains ('INPUT', 'FORWARD' and 'OUTPUT'), the INPUT and OUTPUT chains will have a policy of ACCEPT, and the FORWARD chain will have a policy of DROP (you can override this by providing the 'forward=1' option to the iptables module).

IT378 - M. Di Pierro @ DePaul CTI

## Linux Firewall with IPTables: iptables vs. ipchains



- Firstly, the names of the built-in chains have changed from lower case to UPPER case, because the INPUT and OUTPUT chains now only get locally-destined and locally-generated packets. They used to see all incoming and all outgoing packets respectively.
- The `'-i' flag now means the incoming interface, and only works in the INPUT and FORWARD chains. Rules in the FORWARD or OUTPUT chains that used `'-i' should be changed to `'-o'.
- TCP and UDP ports now need to be spelled out with the --source-port or --sport (or --destination-port/-dport) options, and must be placed after the `'-p tcp' or `'-p udp' options, as this loads the TCP or UDP extensions respectively (you may need to insert the ipt\_tcp and ipt\_udp modules manually).
- The TCP -y flag is now --syn, and must be after `'-p tcp'.
- The DENY target is now DROP, finally.
- Zeroing single chains while listing them works.
- Zeroing built-in chains also clears policy counters.
- Listing chains gives you the counters as an atomic snapshot.
- REJECT and LOG are now extended targets, meaning they are separate kernel modules.
- Chain names can be up to 16 characters.
- MASQ and REDIRECT are no longer targets; iptables doesn't do packet mangling. There is a separate NAT subsystem for this: see the ipnatctl HOWTO.
- Probably heaps of other things I forgot.

IT378 - M. Di Pierro @ DePaul CTI

## Topic:



# Linux syslog

IT378 - M. Di Pierro @ DePaul CTI

## syslogd



Logging is done by the syslogd daemon. The configuration file for this daemon is [`/etc/syslog.conf`](#)

Each entry in this file has the form:

`facility.level action`

The line specifies the action to be taken about messages generated by facility at a given level. Example:

```
kern.crit /var/log/messages  
(log into /var/log/messages critical messages from Kernel)
```

```
auth.err /var/log/auth.log  
(log into /var/log/auth.log authentication errors)
```

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf - Facilities



### Facility Description

auth	Authentication information, but has been deprecated in favor of authpriv
authpriv	Authentication information which may contain data like usernames and other privileged information
console	Messages written to /dev/console by the kernel console output driver
cron	Messages from the cron and at daemons
daemon	Messages from daemons, such as inetd, xinetd, etc.
ftp	Messages from FTP daemons
kern	Messages from the kernel;
lpr	Messages from printer services like lpd, LPRng, etc.
mail	Messages from mail-related daemons such as sendmail or postfix
mark	An internal facility for syslog to generate timestamps
news	Messages from NNTP daemons such as innd or leafnode
syslog	Messages from syslog itself
user	Messages generated by user-space programs and default
uucp	Messages from UUCP
local0-local7	Facilities used by customized programs
*	All facilities except mark

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf - Levels



<b>Level</b>	<b>Description</b>
emerg	The system is unusable
alert	A condition exists that needs to be corrected immediately
crit	Program or subsystem may no longer be useable
err	A component of a program or subsystem may no longer be useable
warning	A warning message
notice	A normal condition with significance
info	An informational message
debug	A debug message, usually does not indicate any problem
none	No level, usually used as a facility exemption when using the wildcard
*	All levels, except none

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf - Examples



```
kern.crit /var/log/messages
(log into /var/log/messages critical messages from Kernel)

authpriv.*      /var/log/auth.log
(log into /var/log/auth.log all authentication messages)

authpriv.err     /var/log/auth.log
(log into /var/log/auth.log authentication errors and higher)

authpriv.=err    /var/log/auth.log
(log into /var/log/auth.log authentication errors only)

authpriv.=err,=debug   /var/log/auth.log
(log into /var/log/auth.log authentication errors and higher and debug info)

kern,auth.crit; auth.=err  /var/log/messages
(log into /var/log/messages all critical messages from kernel and auth, also log auth errors)
```

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf - Caveats



kern.crit /var/log/messages  
 (log into /var/log/messages critical messages from Kernel)

kern.crit -/var/log/messages  
 (same as above but logfile is not flushed and log entry can be lost in case of system crash. DO NOT USE "-" option for critical logs)

kern.crit /dev/console  
 (same as above but log entry is sent to console)

kern.crit mdipierro  
 (same as above but log entry is sent to user mdipierro)

kern.crit @logging.depaul.edu  
 (same as above but log entry is sent remote syslogd daemon, if you administer a network you may want all logs stored by a single host: logging.depaul.edu)

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf – Good Policies (BSD)



# Log all kernel messages, authentication messages of  
 # level notice or higher and anything of level err or  
 # higher to the console.  
 # Don't log private authentication messages!

\*.err;kern.\*;auth.notice;authpriv.none /dev/console

# Log anything (except mail) of level info or higher.  
 # Don't log private authentication messages!

**kern.\*;\*.info;mail.none;authpriv.none /var/log/messages**

# Log daemon messages at debug level only

daemon.\*,!info /var/log/daemon.log

# Log ALL authentication info including success and failures

**authpriv.\* /var/log/secure.log**

IT378 - M. Di Pierro @ DePaul CTI

## Syslogd.conf – Good Policies II (BSD)



```
# Log all the mail messages in one place.  
  
mail.* /var/log/mail.log  
  
# Everybody gets emergency messages, plus log them on another  
# machine. *.emerg  
  
*.emerg @yourfavouritehost.com  
  
# Root and mdipierro get alert and higher messages.  
  
.alert root,mdipierro  
  
# Save mail and news errors of level err and higher in a special file.  
  
uucp,news.crit /var/log/news.log  
  
# Pipe all authentication messages to a filter  
  
auth.* |exec /usr/local/sbin/authfilter
```

IT378 - M. Di Pierro @ DePaul CTI

## Topic:



# Using Sudo

IT378 - M. Di Pierro @ DePaul CTI

## Using SUDO



Example:

You want to prevent everybody but mdipierro and friend to use the command mount

- 1) remove SUID (chmod u-s /bin/mount)
- 2) Edit file **/etc/sudoers** and add entries

```
mdipierro ALL= NOPASSWD: /bin/mount
friend ALL= NOPASSWD: /bin/mount
```

- 3) They can do by using the command

```
sudo /bin/mount
```

IT378 - M. Di Pierro @ DePaul CTI

## Using SUDO



Example:

mdipierro can kill programs started friend

Edit file **/etc/sudoers** and add entries

```
mdipierro ALL= (friend) /bin/kill
```

Now mdipierro can kill a program run by friend using the command:

```
sudo -u friend /bin/kill pid
```

**/etc/sudoers** syntax

```
user1 hosts=(user2) program
```

reads "user1 on host can run program as user2"

IT378 - M. Di Pierro @ DePaul CTI

## Good Sudoers



1) Add entry:

Defaults logfile=/var/log/sudolog  
(everything will be logged in /var/log/sudolog)

2) Create a user Administrator and give him sudo permission to do maintenance tasks: adduser, addgroup, apt-get, at, chkrootkit, mount, ppp, etc.

3) Give users permission to run only those commands they need, no more

4) Administrator does not need to have root password.

Read more: <http://www.courtesan.com/sudo/sudo.html>

IT378 - M. Di Pierro @ DePaul CTI

## Topic:



# Using Cron

IT378 - M. Di Pierro @ DePaul CTI

## Using Cron



Look into */etc/crontab*

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

```
# m h dom mon dow user command
17 * * * * root run-parts --report /etc/cron.hourly
25 6 * * * root run-parts --report /etc/cron.daily
47 6 * * 7 root run-parts --report /etc/cron.weekly
52 6 1 * * root run-parts --report /etc/cron.monthly
```

All files in */etc/cron.hourly/* are executed every hour at \*:17

All files in */etc/cron.daily/* are executed every day at 6:25am

All files in */etc/cron.weekly/* are executed every week at 6:47am on Saturday

All files in */etc/cron.monthly/* are executed every month at 6:52am on the 1st.

They are all executed as root.

IT378 - M. Di Pierro @ DePaul CTI

## Topic:



# Chkrootkit

IT378 - M. Di Pierro @ DePaul CTI

## Chrootkit



Chkrootkit is a program that checks for suspicious files and/or configuration on your system. It consists of a single script with a lot of "find ..." commands

create a **/etc/cron.daily/chkrootkit.sh** file and ask to be notified daily

```
#!/bin/bash  
cd /usr/sbin/  
.chkrootkit | mail -s "subject line" email_address
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache



Getting apache: <http://httpd.apache.org/>

List of Apache modules (Included): <http://httpd.apache.org/docs/mod/>

List of Apache modules (optional): <http://modules.apache.org/search>

On Debian:  
    apt-get install apache  
    apt-get install mod\_ssl     (for https)

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache



[http://httpd.apache.org/docs/misic/security\\_tips.html](http://httpd.apache.org/docs/misic/security_tips.html)

<http://www.apacheweek.com/security/>

<http://www.linuxplanet.com/linuxplanet/tutorials/1527/1>

<http://www.oreilly.com/catalog/apache2/chapter/ch13.html>

<http://apache-server.com/tutorials/LPauth1.html>

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache



The apache daemon (httpd) starts as root and then forks a number of child processes as a different user (apache or www-data or ...) that listen for connections.

The apache configuration file is ***httpd.conf*** (find it!)

The apache log file is ***access.log*** (find it!)

- 1) If the web server is hijacked it should not be able to edit/modify the executables, the configuration files and/or the logs
- 2) Minimize chances of hijacking by removing unnecessary modules/functionality (SSI, mod\_....)
- 3) Set permissions/authorization requirements (<Directory>, htaccess, htpasswd)
- 4) Use chroot, suEXEC
- 5) Use mod\_ssl to use public key to protect communications between client-server.
- 6) Write safe CGI scripts!

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache



- 1) If the web server is hijacked it should not be able to edit/modify the executables, the configuration files and/or the logs

```
chown root:root apache apache/bin
chown root:root apache apache/conf
chown root:root apache apache/logs
chmod 755 apache apache/bin apache/conf apache/log

chmod 511 apache apache/bin/httpd
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



2) Minimize chances of hijacking by removing unnecessary modules/functionality

**SSI:** this allows <--#include .... --> tags in html that instruct the web server to replace the include with some text and or the output of an exec statement. Poses same risk as CGI cripts in general but insidious because hidden in regular web pages.

**Prevent it.**

*emacs*

```
<Directory "/var/www/public">
    # stuff
    Options Includes IncludesNoExec FollowSymLink ....
    #stuff
</Directory>
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



2) Minimize chances of hijacking by removing unnecessary modules/functionality

### Protect System Settings:

Protect all files by default

stop users from setting up their own access policies.

*emacs*

```
<<Directory /> # "/" means default - applies to all files/dirs
    Order Allow, Deny
    AllowOverride None
</Directory>
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



2) Minimize chances of hijacking by removing unnecessary modules/functionality

### Allow/Deny access to folder from a specific IP/domain:

Protect all files by default

stop users from setting up their own access policies.

**emacs**

```
<Directory /myhtmldir> # or <Files foo.html>
    Order Deny, Allow
    Deny from All
    Allow from 140.192.37.195
    AllowOverride None
</Directory>           # or </Files>

<Directory /myhtmldir> # or <Files foo.html>
    Order Deny, Allow
    Deny from 140.192.37.195
    Allow from All
    AllowOverride None
</Directory>           # or </Files>
```

## Securing Apache – httpd.conf



2) Minimize chances of hijacking by removing unnecessary modules/functionality

### Allow users to post their own html files/dirs

And allow them to restrict access within those dirs

**emacs**

```
<Directory /home/*/*www>
    AllowOverride AuthConfig Limit
    #AllowOverride All
</Directory>
```

Just in case users were to post a password in their public www area prevent these files from being downloaded.

**emacs**

```
<Files ~ "\.ht">
    Order Allow, Deny
    Deny from All
</Files>
```

## Securing Apache – httpd.conf



2) Minimize chances of hijacking by removing unnecessary modules/functionality

### Add remove modules (mod\_...)

Modules you probably want:

```
LoadModule config_log_module /usr/lib/apache/1.3/mod_log_config.so
LoadModule mime_magic_module /usr/lib/apache/1.3/mod_mime_magic.so
LoadModule mime_module /usr/lib/apache/1.3/mod_mime.so
LoadModule negotiation_module /usr/lib/apache/1.3/mod_negotiation.so
LoadModule status_module /usr/lib/apache/1.3/mod_status.so
LoadModule autoindex_module /usr/lib/apache/1.3/mod_autoindex.so
LoadModule dir_module /usr/lib/apache/1.3/mod_dir.so
LoadModule cgi_module /usr/lib/apache/1.3/mod_cgi.so
LoadModule userdir_module /usr/lib/apache/1.3/mod_userdir.so
LoadModule alias_module /usr/lib/apache/1.3/mod_alias.so
LoadModule rewrite_module /usr/lib/apache/1.3/mod_rewrite.so
LoadModule access_module /usr/lib/apache/1.3/mod_access.so
LoadModule auth_module /usr/lib/apache/1.3/mod_auth.so
LoadModule expires_module /usr/lib/apache/1.3/mod_expires.so
LoadModule unique_id_module /usr/lib/apache/1.3/mod_unique_id.so
LoadModule setenvif_module /usr/lib/apache/1.3/mod_setenvif.so
LoadModule ssl_module /usr/lib/apache/1.3/mod_ssl.so
LoadModule php3_module /usr/lib/apache/1.3/libphp3.so
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



3) Set permissions/authorization requirements (<Directory>, htaccess, htpasswd)

Create file **/etc/apache/.htpasswd** somewhere not public (use htpasswd)

```
emacs
<Directory /var/www>
    Order Allow, Deny
    AuthName "Restricted Access"
    AuthType Basic
    AuthUserFile /etc/apache/.htpasswd
    Require valid-user
    Satisfy Any
    Deny from All
</Directory>
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – htpasswd



The **.htpasswd** file can be created manually or, better, via the command:

```
htpasswd [-cmdps] passwordfile username
htpasswd -b[cmdps] passwordfile username password
```

- c Create a new file.
  - n Don't update file; display results on stdout.
  - m Force MD5 encryption of the password.
  - d Force CRYPT encryption of the password (default).
  - p Do not encrypt the password (plaintext).
  - s Force SHA encryption of the password.
  - b Use the password from the command line rather than prompting for it.
- On Windows, TPF and NetWare systems the '-m' flag is used by default.  
On all other systems, the '-p' flag will probably not work.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



3) Set permissions/authorization requirements (<Directory>, htaccess, htpasswd)

```
emacs
<Directory /var/www>
Order Deny,Allow
Allow from All
AuthName "Restricted Access"
AuthType Basic
AuthUserFile /etc/apache/.htpasswd
Require valid-user
Satisfy Any # means satisfy both Allow and Require
</Directory>
```

Modules involved in authentication: mod\_access (Allow/Deny), mod\_auth (checks if user is in .htpasswd and whether password matches).

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



3) Set permissions/authorization requirements (<Directory>, htaccess, htpasswd)

Using the htaccess. One can decentralize the access control from httpd.conf to a local .htaccess file in the directory in question.

```
<Directory /var/www>
    Order Allow, Deny
    AuthName "Restricted Access"
    AuthType Basic
    AuthUserFile /etc/apache/.htpasswd
    Require valid-user
    Satisfy Any # means satisfy both Allow and Require
    Deny From All
</Directory>
```

Then in /var/www create a file **.htaccess** containing

```
AuthName "Restricted Access"
AuthType Basic
AuthUserFile /etc/apache/.htpasswd
Require valid-user
Satisfy Any
Deny From All
```

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



3) chroot

Load module mod\_security and In your httpd.conf add the line

```
SecChrootDir /chroot/apache
```

Unlike external chrooting mod\_security chrooting requires *no additional files to exist in jail*. The chroot call is made after web server initialisation but before forking. Because of this, all shared libraries are already loaded, all web server modules are initialised, and log files are opened. You only need your data in jail.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache – httpd.conf



The **suEXEC** feature -- introduced in Apache 1.2 -- provides Apache users the ability to run **CGI** and **SSI** programs under user IDs different from the user ID of the calling web-server. Normally, when a CGI or SSI program executes, it runs as the same user who is running the web server.

Used properly, this feature can reduce considerably the security risks involved with allowing users to develop and run private CGI or SSI programs. However, if suEXEC is improperly configured, it can cause any number of problems and possibly create new holes in your computer's security. If you aren't familiar with managing setuid root programs and the security issues they present, we highly recommend that you not consider using suEXEC.

IT378 - M. Di Pierro @ DePaul CTI

## Securing Apache



<http://httpd.apache.org/docs/mod/index-bytype.html>

### DO NOT DO THE FOLLOWING

IIS DAV Exploit (FIXED BUT PATCH)

<http://lists.virus.org/bugtraq-0305/msg00339.html>

Javascript Password Protection (DO NOT USE EVER)

<http://www.pageresource.com/javascript/jex8.htm>

IT378 - M. Di Pierro @ DePaul CTI

## Useful Tools



	Linux	Windows
Configuration tool:	<b>bastille</b>	<b>mmc</b>
	[still need to know where configuration files are and edit them]	
Check for rootkits/etc.:	<b>chkrootkit</b>	<b>antivirus/anispyware</b>
	[OK as long you update them]	
Firewall:	<b>iptables</b>	<b>zone-alarm?</b>
	[Better if connected with Intrusion Detection]	
Port scanner:	<b>nmap</b>	<b>nmap, superscan</b>
	[Use only to check open ports on your systems/networks]	
Intrusion detection (IDS):	<b>snort</b>	<b>snort, prevx</b>
	[Connect to firewall and update rules]	
Vulnerability scanner:	<b>nessus</b>	<b>nessus (commercial)</b>
	[using nessus outside your network is illegal!!!]	
Forensics:	<b>sleuth kit</b>	<b>winhex</b>
	[switch system off, copy disk, then do forensics on copy]	

IT378 - M. Di Pierro @ DePaul CTI

Topic:



# Port Scanning with nmap



IT378 - M. Di Pierro @ DePaul CTI

## What's a port scan?



A **port scanner** is a piece of software designed to search a network host for open ports. This is often used by administrators to check the security of their networks and by hackers to compromise it.

Example of TCP Ports:

20,21	FTP (close)	22	SSH
23	Telnet (close)	25	SMTP
79	Finger (close)	80	HTTP
110	Post Office	113	Ident (close)
137-139	Netbios (close)	443	HTTPS
989,990	FTP with SSL	8080	HTTP Proxy

Results of scan can be:

- **Accepted or Open:** Reply indicates service is listening on the port.
- **Denied or Closed:** Reply indicates connections will be denied to the port.
- **Dropped or Blocked:** There was no reply from the host.

IT378 - M. Di Pierro @ DePaul CTI

## How do I perform a port scan?



Simplest type of scan

```
nmap 140.192.37.195
nmap -v -sS -O -p 1-1024 140.192.37.195
```

- v Verbose. Its use is recommended.
- sS TCP SYN stealth port scan (best all-around TCP scan)
- O Guess remote operating system
- p <range> ports to scan.

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### TCP connect() scanning

This is the most basic form of TCP scanning. The `connect()` system call provided by your operating system is used to open a connection to every interesting port on the machine. If the port is listening, `connect()` will succeed, otherwise the port isn't reachable. One strong advantage to this technique is that you don't need any special privileges. Any user on most UNIX boxes is free to use this call. Another advantage is speed. While making a separate `connect()` call for every targeted port in a linear fashion would take ages over a slow connection, you can hasten the scan by using many sockets in parallel. Using non-blocking I/O allows you to set a low time-out period and watch all the sockets at once. This is the fastest scanning method supported by nmap. The big downside is that this sort of scan is easily detectable and filterable. The target hosts logs will show a bunch of connection and error messages for the services which take the connection and then have it immediately shutdown.

```
nmap -v -sT 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### TCP SYN scanning

This technique is often referred to as "half-open" scanning, because you don't open a full TCP connection. You send a SYN packet, as if you are going to open a real connection and wait for a response. If you receive a SYN|ACK packet then the port is listening. A RST is indicative of a non-listener. If a SYN|ACK is received, you immediately send a RST to tear down the connection (actually the kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets.

```
nmap -v -sS 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### TCP FIN scanning

Some firewalls and packet filters watch for SYNs to restricted ports, and programs like synlogger and Courtney are available to detect these scans. FIN packets, on the other hand, may be able to pass through unmolested.

The idea is that closed ports tend to reply to your FIN packet with the proper RST. Open ports, on the other hand, tend to ignore the packet in question. As Alan Cox has pointed out, this is required TCP behavior. However, some systems (notably Windows boxes), are broken in this regard. They send RST's regardless of the port state, and thus they aren't vulnerable to this type of scan. It works well on most other systems I've tried. It is often useful to discriminate between a \*NIX and NT box, and this can be used to do that.

```
nmap -v -sF 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### TCP reverse ident scanning

It was noted by Dave Goldsmith in a 1996 Bugtraq post that the ident protocol (rfc1413) allows for the disclosure of the username of the owner of any process connected via TCP, even if that process didn't initiate the connection. So you can, for example, connect to the http port and then use identd to find out whether the server is running as root (does not always work). This can only be done with a full TCP connection to the target port.

```
nmap -v -sT -I 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### UDP ICMP port unreachable scanning

Uses UDP protocol instead of TCP. Based on fact that most hosts send an ICMP PORT UNREACH error when you send a packet to a closed UDP port.

Thus you can find out if a port is NOT open, and by exclusion determine which ports which are. Neither UDP packets, nor the ICMP errors are guaranteed to arrive, so UDP scanners of this sort must also implement retransmission of packets that appear to be lost. This scanning technique is slow.

Also, you will need to be root for access to the raw ICMP socket necessary for reading the port unreachable.

Example: Solaris rpcbind hole. rpcbind can be found hiding on an undocumented UDP port somewhere above 32770. So it doesn't matter that 111 is blocked by the firewall. But can you find which of the more than 30,000 high ports it is listening on? With a UDP scanner you can!

```
nmap -v -sU 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Types of Scanning



### ACK scan

This advanced method is usually used to map out firewall rulesets. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets.

```
nmap -v -sA 127.0.0.1
```

IT378 - M. Di Pierro @ DePaul CTI

## Why nmap?



**Best free port scanner** that runs under Unix/Linux and Windows

**Easy to use.** Figures out most info without need to ask user

**Flexible.** You can specify many parameters (type of scan, ports, IP ranges, etc.)

**Dynamic delay time calculations.** Some scanners require that you supply a delay time between sending packets (?). Nmap tries to determine the best delay time for you by dynamically keeping track of packet retransmissions.

**Retransmission.** If a port does respond to a UDP or FIN scan, it can be a false positive. Nmap retries ports that do not respond.

**Parallel port scanning.** Nmap can scan multiple ports at the same time (open multiple connections for example) instead of naive looping.

IT378 - M. Di Pierro @ DePaul CTI



## Intrusion Detection with Snort



IT378 - M. Di Pierro @ DePaul CTI

## What does SNORT do?



Snort monitors network traffic and detects patterns such those described in

attack-responses.rules	backdoor.rules
bad-traffic.rules	sql.rules
telnet.rules	ddos.rules
netbios.rules	tftp.rules
dns.rules	policy.rules
virus.rules	dos.rules
porn.rules	web-attacks.rules
exploit.rules	rpc.rules
web-cgi.rules	finger.rules
rservices.rules	ftp.rules
scan.rules	web-iis.rules
smtp.rules	

These are rule files stored in */etc/snort/snort.conf*

Usually snort just logs bad traffic but it can work with iptables to block it.

IT378 - M. Di Pierro @ DePaul CTI

## SNORT – in sniffer's mode



Snort can run in three modes:

- **from console**
- **as daemon**
- **with iptables**

### From console, examples:

If you just want to print out the TCP/IP packet headers to the screen (i.e. sniffer mode)

`snort -v`

If you want to see the headers and the application data in transit:

`snort -v -d`

If you want an even more descriptive display, showing the data link layer headers too:

`snort -v -d -e`

IT378 - M. Di Pierro @ DePaul CTI

## Sniffer's mode and NIDS mode



```
snort -dev -l ./log -h 192.168.1.0
```

With the -h you specify the home network

With the -l switch you specify the log directory. Snort uses the address of the remote computer as the directory in which it places packets and sometimes it uses the local host address.

To enable Network Intrusion Detection (NIDS) mode so that you don't record every single packet sent down the wire need a configuration file **/etc/snort/snort.conf**

```
snort -dev -l ./log -h 192.168.1.0/24 -c snort.conf
```

And to start as daemon, guess what?

```
/etc/init.d/snort start
```

IT378 - M. Di Pierro @ DePaul CTI

## SNORT configuration file



A peek into **/etc/snort/snort.conf**

```
preprocessor stream4: detect_scans
```

Includes module stream4 with option "detect scans". This module enable TCP stream reassembly and stateful analysis capabilities. Has the ability to track up to 256 simultaneous TCP streams.

IT378 - M. Di Pierro @ DePaul CTI

## SNORT configuration file



A peek into */etc/snort/snort.conf*

preprocessor portscan: \$HOME\_NET 4 3 portscan.log

Includes module portscan and every time the same remote server tries to connect to more than 4 ports on \$HOME\_NET (localhost) in 3 seconds it logs the event as a scan in the file portscan.log.

IT378 - M. Di Pierro @ DePaul CTI

## SNORT configuration file



A peek into */etc/snort/snort.conf*

preprocessor http\_decode: 80

Includes module http\_decode module.

http\_decode normalizes HTTP requests from remote machines by converting any %XX character substitutions to their ASCII equivalent. This is very useful for doing things like defeating hostile attackers trying to stealth themselves from IDSs by mixing these substitutions in with the request.

Specify the port numbers you want it to analyze as arguments.

Options:

- unicode turns off detection of UNICODE directory traversal, etc attacks.
- cginull turns off detection of CGI NULL code attacks.

IT378 - M. Di Pierro @ DePaul CTI

## SNORT configuration file



A peek into */etc/snort/snort.conf*

```
include bad-traffic.rules
include exploit.rules
include scan.rules
include finger.rules
include ftp.rules
#...etc.
```

They include snort rule files. A rule file says what to look for in a packet, what to log.  
Example:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP BOO";)
```

Reads as "load as "FTP BOO' any attempt to connect to local port 21".

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI view-
source directory traversal";flags: A+; uricontent:"/view-source"; nocase; content:"../";
nocase; reference:cve,CVE-1999-0174;classtype:web-application-attack; sid:848; rev:
2;)
```

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...
... url contains "/hsx.cgi" and

IT378 - M. Di Pierro @ DePaul CTI

**SNORT Alerts**



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...  
... url contains "/hsx.cgi" and  
... content contains "..../" and

IT378 - M. Di Pierro @ DePaul CTI

**SNORT Alerts**



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...  
... url contains "/hsx.cgi" and  
... content contains "..../" and  
... content conatins "%00"

IT378 - M. Di Pierro @ DePaul CTI

**SNORT Alerts**



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...  
... url contains "/hsx.cgi" and  
... content contains "..../" and  
... content contains "%00"  
The log as "WEB-CGI HyperSeek directory traversal attempt" (user defined)

IT378 - M. Di Pierro @ DePaul CTI

**SNORT Alerts**



A peek into */etc/snort/web\_cgi.rules*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...  
... url contains "/hsx.cgi" and  
... content contains "..../" and  
... content contains "%00"  
The log as "WEB-CGI HyperSeek directory traversal attempt" (user defined)

Give a reference number to this rule (for use with compatible databases)

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into `/etc/snort/web_cgi.rules`

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI
HyperSeek directory traversal attempt"; uricontent:"/hsx.cgi"; content:"..../";
content:"%00"; flags:A+; reference:bugtraq,2314; reference:cve,CAN-2001-0253;
classtype:web-application-attack; sid:803; rev:2;)
```

Alert on all incoming connections to port 80 if ...

- ... url contains "/hsx.cgi" and
- ... content contains "..../" and
- ... content contains "%00"

The log as "WEB-CGI HyperSeek directory traversal attempt" (user defined)

Give a reference number to this rule (for use with compatible databases)  
And a snort rule ID.

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into `/etc/snort/viruses.rules`

```
alert tcp any 110 -> any any (msg:"Virus - SnowWhite Trojan Incoming";
content:"Suddlently"; sid:720; classtype:misc-activity; rev:3;)
```

```
alert tcp any 110 -> any any (msg:"Virus - Possible pif Worm"; content:".pif";
nocase; sid:721; classtype:misc-activity; rev:3;)
```

```
alert tcp any 110 -> any any (msg:"Virus - Possible NAVIDAD Worm"; content:
"NAVIDAD.EXE"; nocase; sid:722; classtype:misc-activity; rev:3;)
```

```
alert tcp any 110 -> any any (msg:"Virus - Possible CheckThis Trojan"; content:"|6E
61 6D 65 20 3D 22 6C 69 6E 6B 73 2E 76 62 73 22|"; sid:774; classtype:misc-
activity; rev:3;)
```

Binary

IT378 - M. Di Pierro @ DePaul CTI

## SNORT Alerts



A peek into `/etc/snort/dos.rules`

```
alert udp any 19 <> $HOME_NET 7 (msg:"DOS UDP Bomb"; classtype:attempted-dos; sid:271; rev:1;)

alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"DOS IGMP dos attack";
content:"|02 00|"; depth: 2; ip_proto: 2;
fragbits: M+; classtype:attempted-dos; sid:272; rev:1;)
```

Checks for the More Fragments bit in the packet header, which forces receiver to wait for more packets to follow.

IT378 - M. Di Pierro @ DePaul CTI

## Using SNORT with IPtables



You need to send traffic to snort\_inline using the QUEUE target. For example,

```
iptables -A OUTPUT -p tcp --dport 80 -j QUEUE
```

(sends all TCP traffic leaving the firewall going to port 80 to the QUEUE target)

This is what sends the packet from kernel space to user space (snort\_inline). Or look for firewall scripts at <http://www.honeynet.org/papers/honeynet/tools/>

Finally, start snort\_inline.

```
snort_inline -QD -c /etc/snort.conf -l /var/log/snort
```

You can use the following command line options:

- Q - Gets packets from iptables.
- D - Runs snort\_inline in daemon mode. PID is stored at /var/run/snort\_inline.pid
- c - Reads the following configuration file.
- l - Logs to the following directory.

IT378 - M. Di Pierro @ DePaul CTI

# MDP Guide to Network Programming

Massimo Di Pierro

October 11, 2007

## **Abstract**

This notes are a work in progress, please report any error.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Header Files . . . . .	4
1.2	File descriptors . . . . .	5
<b>2</b>	<b>Dealing with IP addresses</b>	<b>6</b>
2.1	Representation of addresses . . . . .	6
2.2	Storing an IP address (ipv4) . . . . .	7
2.3	Retrieving an IP address . . . . .	7
2.4	Convert host names to IP address . . . . .	8
<b>3</b>	<b>TCP Sockets (SOCK_STREAM)</b>	<b>8</b>
3.1	TCP server . . . . .	9
3.2	TCP client . . . . .	10
3.3	Staying alive . . . . .	11
3.4	How to copy a socket . . . . .	11
3.5	Dealing with SIGHUP . . . . .	11
3.6	Terminators and marshalling . . . . .	12
<b>4</b>	<b>UDP Sockets (SOCK_DGRAM)</b>	<b>12</b>
4.1	Sending a datagram with sendto . . . . .	13
4.2	Receiving a datagram with recvfrom . . . . .	13
4.3	Non-blocking IO . . . . .	14
4.4	Asynchronous IO . . . . .	15
4.5	Using poll . . . . .	16
4.6	Using select . . . . .	17
4.7	UDP broadcasting with sendto . . . . .	17
4.8	UDP broadcasting and recvfrom . . . . .	17
4.9	UDP multicasting and sendto . . . . .	18
4.10	UDP multicast and recvfrom . . . . .	19
<b>5</b>	<b>Signals, Processes and Threads</b>	<b>19</b>
5.1	Using Alarms . . . . .	20
5.2	Ignore Signals . . . . .	21
5.3	Blocking a signal . . . . .	21
5.4	Creating a process . . . . .	22
5.5	Killing zombies . . . . .	23

5.6	Creating a deamon process (fork twice) . . . . .	23
5.7	Running commands withing programs: system and execl . . . . .	24
5.8	Creating a deamon process for inetd . . . . .	24
5.9	Locking a file . . . . .	25
5.10	Getting file info . . . . .	26
5.11	Creating threads . . . . .	26
<b>6</b>	<b>Appendix A - Working Examples</b>	<b>28</b>
6.1	Gethostbyname . . . . .	28
6.2	TCP server with fork . . . . .	28
6.3	TCP client . . . . .	30
6.4	UDP server . . . . .	31
6.5	UDP client . . . . .	32
6.6	UDP server using non-blocking IO . . . . .	33
6.7	UDP server using asynchronous IO . . . . .	34
6.8	UDP server using poll . . . . .	36
6.9	UDP server using select . . . . .	37
6.10	Setting an alarm . . . . .	39
6.11	Killing zombies . . . . .	40
<b>7</b>	<b>Appendix B - OSI Model</b>	<b>41</b>
7.1	Physical . . . . .	42
7.2	Data Link . . . . .	42
7.3	Network . . . . .	42
7.4	Transport . . . . .	42
7.5	Session . . . . .	42
7.6	Presentation . . . . .	43
7.7	Application . . . . .	43
<b>8</b>	<b>Appendix B - UNIX Commands</b>	<b>43</b>

# 1 Introduction

I use both C and C++. Compile with

```
g++ filename.cpp -o filename.exe -lpthread
```

## 1.1 Header Files

I will assume the following headers and included

```
// BEGIN FILE: mdp_all.h
// C headers
#include "sys/types.h"
#include "sys/socket.h"
#include "sys/time.h"
#include "time.h"
#include "netinet/in.h"
#include "arpa/inet.h"
#include "errno.h"
#include "fcntl.h"
#include "netdb.h"
#include "signal.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "sys/stat.h"
#include "sys/uio.h"
#include "unistd.h"
#include "sys/wait.h"
#include "sys/un.h"
#include "sys/select.h"
#include "poll.h"
#include "strings.h"
#include "pthread.h"

// C++ headers and STL headers
#include <iostream>
#include <string>
#include <vector>
```

```

#include "deque"
#include "map"
using namespace std;

#ifndef HAVE_INET_NTOP
#define inet_ntop(a,b) inet_ntoa(b)
#define inet_pton(a,b,c) inet_aton(b,c)
#endif

void exit_message(int en, string message) {
    cerr << "FROM PROCESS PID: " << getpid() << endl;
    cerr << "CHILD OF PROCESS PID: " << getppid() << endl;
    cerr << "FATAL ERROR: " << message << endl;
    cerr << "EXITING WITH ERROR NUMBER: " << en << endl;
    exit(en);
}

```

In all the examples below we will use `exit_message` to notify the user when an error occurs. In real life we may want to deal with error in some more sophisticated way.

## 1.2 File descriptors

In Unix/Linux everything is a file (a file is a file, a socket is a file, a device is a file). From the point of view of a program a file (including a socket or a device) are identified by a single integer number called file descriptor. For example to write a buffer to the file “myfile.dat”:

```

int main() {
    int fd=open("myfile.dat",O_WRONLY);
    if(fd<0)
        exit_message(1,"unable to open file");
    char buffer[128];
    int buffer_size=128;
    if(write(fd,buffer,buffer_size)!=buffer_size)
        exit_message(1,"write failuer");
    close(fd);
}

```

In the example fd is the file descriptor associated to the file. If you have a pointer to file (FILE\* fp) and not an integer file descriptor you can get it with

```
fd=fileno(fp);
```

## 2 Dealing with IP addresses

Relevant commands:

- **htonl**: converts 32bits int from host to network
- **ntohl**: converts 32bits int from network to host
- **htons**: converts 16bits int from host to network
- **ntohs**: converts 16bits int from network to host
- **inet\_pton**: converts IP address (char[]) to 32bits network
- **inet\_aton**: same as above (only ipv4)
- **inet\_ntop**: converts 32bits network to IP address (char[])
- **inet\_ntoa**: same as above (only ipv4)
- **inet\_addr**: do not use, problems with broadcasting.
- **gethostbyname**: get the ip address from the hostname

### 2.1 Representation of addresses

There are at least four representations for an address:

- Host Name: www.yahoo.com
- IP Address: 216.109.118.77
- 32bits long network:  $216 \cdot 2^{24} + 109 \cdot 2^{16} + 118 \cdot 2^8 + 77 = 36310\,52365$
- 32bits long host: the host representation of the long integer above.

Functions that deal with socket need to know: the address, the port number and the socket type. The structure `sockaddr_in` has member variables to hold the socket type, the port number and the ip address in 32bits long network representation.

```
struct sockaddr_in {  
    short          sin_family; // address family  
    u_short        sin_port;  // socket port  
    struct in_addr sin_addr; // address (32bits net)  
    char           sin_zero[8]; // padding  
};
```

## 2.2 Storing an IP address (ipv4)

```
// input variables:  
char ip_address[16] = "216.109.118.77";  
int port_number=80;  
// ouput variable:  
struct sockaddr_in address;  
// store IP address in 'address':  
memset(&address, 0, sizeof(address));  
address.sin_family=AF_INET;  
address.sin_port=htons(port_number);  
if/inet_pton(AF_INET, ip_address, &address.sin_addr)<=0  
    exit_message(1, "inet_pton");
```

## 2.3 Retrieving an IP address

```
// input variable:  
struct sockaddr_in address; // as returned by recvfrom  
// output variables:  
char ip_address[16];  
int port_number=80;  
// retrieve IP address from 'address':  
port_number=ntohs(address.sin_port);  
strncpy(ip_address, inet_ntop(AF_INET, address.sin_addr), 16);
```

## 2.4 Convert host names to IP address

```
// output variable:  
char ip_address[16];  
// get host ip:  
struct hostent* hptr=gethostbyname(name.c_str());  
if(hptr==0)  
    exit_message(1,"gethostbyname, invalid address");  
if(hptr->h_length!=4)  
    exit_message(1,"gethostbyname, not supported");  
strncpy(ip_address,  
        inet_ntop(family,*((struct in_addr*)  
            *(hptr->h_addr_list))),16);
```

## 3 TCP Sockets (SOCK\_STREAM)

TCP is a reliable protocol. A connection is established between a client and a server. When the client sends something to the server the server acknowledges receiving and vice versa. Telnet, ssh, tcp, pop, imap are all based on TCP.

Relevant commands:

- **socket**: creates a socket
- **bind**: binds socket to a port
- **listen**: a server listen for connection
- **accept**: a server accepts a client
- **connect**: a client attempt to connect to a server
- **recv**: read from a TCP socket
- **read**: same as above but default flags (avoid)
- **send**: write to a TCP socket
- **write**: same as above but default flags (avoid)
- **setsockopt**: set socket options.

### 3.1 TCP server

Any TCP server performs the following operations in the same order:

- Create socket (family AF\_INET for ipv4 and AF\_INET6 for ipv6)
- Bind socket to local port
- *Optional:* set socket options (using setsockopt or fcntl)
- Listen for connection
- *Optional:* fork or create\_pthread
- Accept a connection (blocking unless otherwise specified)
- Read/write or recv/send from/to socket
- Close socket

```
int sfd=socket(AF_INET,SOCK_STREAM,0);
if(sfd<0) exit_message(1,"socket");
int port_number=80; // port you want to server from
int maxn=10; // max number of connections
struct sockaddr_in address;
memset(&address,0,sizeof(address));
address.sin_family=AF_INET;
address.sin_port=htons(port_number);
address.sin_addr.s_addr=htonl(INADDR_ANY);
if(bind(sfd,(struct sockaddr*)&address,sizeof(address)))
    exit_message(1,"bind");
if(listen(sfd,maxc))
    exit_message(1,"bind, too many connections?");
struct sockaddr_in client_address;
socklen_t length(sizeof(client_address));
int cfd=accept(sfd,(struct sockaddr*)&client_address,&length);
if(cfd<0) exit_message(1,"accept");
char buffer[128];
int buffer_size;
if(send(cfd,buffer,buffer_size,0)!=buffer_size)
    exit_message(1,"send");
```

```

if(recv(sockfd,buffer,buffer_size,0)!=buffer_size)
    exit_message(1,"recv");
close(sockfd); // close connection to client
close(listenfd); // stop listening

```

NOTE: We bind to local address INADDR\\_ANY because we assume we have only one network card. If this is not the case we have to specify the IP address associated to the network card from which we wish to serve.

### 3.2 TCP client

Any TCP client performs the following operations in the same order:

- Create socket (family AF\\_NET for ipv4 and AF\\_INET6 for ipv6)
- *Optional:* set socket options (setsockopt or fcntl)
- Connect to server
- read/write or recv/send from/to socket
- Close socket

```

int sfd=socket(AF_INET,SOCK_STREAM,0);
if(sfd<0) exit_message(1,"bind");
// fill address of remote server
memset(&address,0,sizeof(address));
address.sin_family=AF_INET;
address.sin_port=htons(port_number);
if(inet_pton(AF_INET,ip_address,&address.sin_addr)<=0)
    exit_message(1,"inet_pton");
int ce=connect(sfd,(struct sockaddr*)&address,
               sizeof(address));
if(ce!=0 && errno==ECONNREFUSED) exit_message(1,"connection refused");
if(ce!=0 && errno==ETIMEDOUT) exit_message(1,"timeout");
// else if ce==0
char buffer[128];
int buffer_size;
if(recv(sockfd,buffer,buffer_size,0)!=buffer_size)
    exit_message(1,"recv");

```

```
if(send(sfd,buffer,buffer_size,0)!=buffer_size)
    exit_message(1,"send");
close(sfd); // close connection to client
```

### 3.3 Staying alive

When TCP connections can be idle for long time they may timeout. To prevent timeout, after creating a socket, set the SO\_KEEPALIVE option of the socket to ON.

```
static int alive_on=1;
setsockopt(sfd,SOL_SOCKET,SO_KEEPALIVE,
           &alive_on,sizeof(alive_on));
```

One should check if the return value of setsockopt is different from zero and eventually return error.

### 3.4 How to copy a socket

It is unsafe to copy a file descriptor (that may be associated to a socket) in the following way

```
int sfd2=sfd;
```

because one may lose track of the copies and accidentally close the same file descriptor twice, close(sfd) and close(sfd2). This would be incorrect. To prevent the problem, always copy a file descriptor using the dup function:

```
int sfd2=dup(sfd);
```

Now one can safely close(sfd) and close(sfd2) as if they were different file descriptors, even if they are connected to the same file or socket.

### 3.5 Dealing with SIGHUP

When a process tries to perform IO on a socket while the connection is closed by the peer, the process receives a SIGHUP signal. If this happens one may want to try to connect again or just abort. See examples below on how to catch a signal.

### 3.6 Terminators and marshalling

If a host (let's say the client) sends data to a peer (let's say the server) how does the server knows the size of the data to read? There are two solutions to the problem:

- Using a terminator character or a sequence of characters as a terminator sequence.
- Using marshalling, i.e. send the size of the data as an int (32 bits) and then the data.

Different solutions correspond to different protocols. Typically for text based protocols (such as HTML and XML) the former solution is adopted. For ASCII based protocols (such as ASCII transfer in FTP) the latter solution is adopted.

## 4 UDP Sockets (SOCK\_DGRAM)

UDP is an unreliable protocol. No connection is established between a client and a server. When the client sends something to the server there is no guarantee that the message has been received, and vice versa. UDP is typically used in internet radio/video broadcasting or for some other type of broadcasting when the failure rate is considered acceptable (for example, skipping a video frame once in a while).

Relevant commands:

- `sendto`: send a datagram to a remote host
- `recvfrom`: receive a datagram from a remote host
- `fcntl`: the most general way to set options on a socket or any file description
- `poll`: wait for something to happen on a set of file descriptors
- `select`: same as poll but old and cumbersome
- `sigemptyset`, `sigaddset`, `sigaction`, `sigprocmask`: deal with signals

## 4.1 Sending a datagram with sendto

To send a UDP datagram contained in buffer

- Create Socket
- Bind socket to port (port 0 if you do not care)
- Store destination (remote) address
- call sendto

```
sfd=socket(AF_INET, SOCK_DGRAM, 0);
// if sfd<0 error
// set local port and bind to it
local_address.sin_family = AF_INET;
local_address.sin_port = htons(local_port);
local_address.sin_addr.s_addr = INADDR_ANY;
if(bind(sfd,(const sockaddr*)&local_address,
        sizeof(local_address))<0)
    exit_message(1,"bind");
// set remote address and store it
remote_address.sin_family = AF_INET;
remote_address.sin_port = htons(remote_port_number);
if/inet_pton(AF_INET,remote_ip_address,
             &remote_address.sin_addr)<=0)
    exit_message(1,"inet_pton");
// send buffer to remote address
char buffer[128];
int buffer_size=128;
sendto(sfd,buffer,buffer_size,0,
       (const sockaddr*)&remote_address,sizeof(remote_address));
// close socket
close(sfd);
```

## 4.2 Receiving a datagram with recvfrom

To receive a UDP datagram and store it in buffer

- Create Socket

- Bind socket to port (port 0 if you do not care)
- Call recvfrom
  - *Optional:* retrieve destination (remote) address

```

sfd=socket(AF_INET, SOCK_DGRAM, 0);
// if sfd<0 error
// set local port and bind to it
local_address.sin_family = AF_INET;
local_address.sin_port = htons(local_port);
local_address.sin_addr.s_addr = INADDR_ANY;
if(bind(sfd,(const sockaddr*)&local_address,
        sizeof(local_address))<0)
    exit_message(1,"bind");
socklet_t addrsize=sizeof(remote_address);
char buffer[128];
int buffer_size=128;
recvfrom(sfd,buffer,buffer_size,0,
          (const sockaddr*)&remote_address,&addrsize);
// Optional retrieve remote address
int remote_port_number;
char remote_ip_address[16];
remote_port_number=ntohs(remote_address.sin_port);
strncpy(remote_ip_address,
        inet_ntop(AF_INET,remote_address.sin_addr),16);
// close socket
close(sfd);

```

### 4.3 Non-blocking IO

Non blocking IO works for both UDP and TCP, but it is more commonly used for UDP.

To set a socket nonblocking, immediately after creating the socket

```
fcntl(sfd,F_SETFL,O_NONBLOCK);
```

When one does this than one should put the recv/recvfrom in a loop and periodically try to read from the socket. If there is nothing to read

from the socket recv/recvfrom will return with an error status different from EWOULDBLOCK. If recv/recvfrom return EWOULDBLOCK than a socket error has occurred.

```
while(1) {
    int e=recvfrom(sfd,buffer,buffer_size,0,
                  (struct sockaddr*)&address,&address_size);
    if(e<0 && errno=EWOULDBLOCK)
        exit_message(1,"recvfrom->EWOULDBLOCK");
    // do something such as sleep(1) and try again
    ...
}
```

## 4.4 Asynchronous IO

Asynchronous IO works for both UDP and TCP, but it is more commonly used for UDP. Asynchronous IO means that the socket is set to non-blocking and the OS is requested to notify the process when there is something to read from the socket. The notification is a SIGIO signal.

To set a socket asynchronous, immediately after creating the socket

```
fcntl(sfd,F_SETFL,O_NONBLOCK);
fcntl(sfd,F_SETOWN,getpid());
fcntl(sfd,F_SETFL,FASYNC);
```

If something happens to the socket the process itself, getpid(), receives a SIGIO signal. To catch the signal:

```
void sigio_handler(int singnum) {
    // take action
}

...
struct sigaction action;
action.sa_handler=sigio_handler;
if(sigemptyset(&action.sa_mask)<0)
    exit_message(1,"sigemptyset");
if(sigaddset(&action.sa_mask,SIGIO))
    exit_message(1,"sigaddset");
```

```

if(sigaction(SIGIO,&action,NULL)<0)
    exit_message(1,"sigaction");

```

## 4.5 Using poll

The poll function is a blocking function that allows execution to wait for something to happen to any of a set of the file descriptors.

Given two file descriptors (sfda and sfdb)

```

struct pollfd fds[2];
fds[0].fd=sfda;
fds[0].events=POLLRDNORM | POLLERR;
fds[1].fd=sfdb;
fds[1].events=POLLRDNORM | POLLERR;

while(1) {
    int n=poll(fds,2,-1); // -1 means poll forever
    if(n== -1) exit_message(1,"poll");
    if(n==0) exit_message(1,"timeout");
    if(fds[0].revents & POLLRDNORM) {
        // something to read from sfda, deal with it
    }
    if(fds[0].revents & POLLERR) {
        // error from sfda, deal with it
    }
    if(fds[1].revents & POLLRDNORM) {
        // something to read from sfdb, deal with it
    }
    if(fds[1].revents & POLLERR) {
        // error from sfdb, deal with it
    }
}

```

The lines:

```

fds[0].fd=sfda;
fds[0].events=POLLRDNORM | POLLERR;

```

indicate that we are registering `sfda` with poll and poll will return when POLLRDNORM (something to read) or POLLERR (socket error).

## 4.6 Using select

## 4.7 UDP broadcasting with sendto

Broadcasting is a way to send a single datagram to every machine in a subnetwork. All machines that listen for the broadcast will receive the same datagram. Each network has a broadcast address. To get the broadcast address

```
> ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:51:9D:F7:AC:A0
          inet addr:140.192.36.40  Bcast:140.192.36.255  Mask:255.255.255.0
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:162132 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:78186 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:69309232 (66.0 MiB)  TX bytes:9676649 (9.2 MiB)
                  Interrupt:18 Base address:0x9000 Memory:fc000000-fc020000
```

Notice:

Bcast:140.192.36.255

That is by sending a datagram to this address all nodes in the subnetwork will receive the dgram.

To enable broadcasting on a socket, after creating the socket, before sendto

```
static int bc_on=1;
setsockopt(sfd,SOL_SOCKET,SO_BROADCAST,
           &bc_on,sizeof(bc_on));
```

## 4.8 UDP broadcasting and recvfrom

In order to be able to receive broadcasted datagrams, after creating the socket

```
static int reuse_on=1;
setsockopt(sfd,SOL_SOCKET,SO_REUSEADDR,
           &reuse_on,sizeof(reuse_on));
```

also, very importantly, bind the socket to the broadcasting address (`bc_ip_address`, `bc_port_number`):

```
// set local port and bind to it
struct sockaddr_in bc_address;
bc_address.sin_family = AF_INET;
bc_address.sin_port = htons(bc_port);
if(inet_pton(AF_INET,bc_ip_address,
    &bc_address.sin_addr)<=0)
    exit_message(1,"inet_pton");
if(bind(sfd,(const sockaddr*)&bc_address,
    sizeof(local_address))<0)
    exit_message(1,"bind");
```

Routers may reroute broadcasted datagrams. The old way to broadcast to a subnetwork was to sendto 255.255.255.255. If you use this address do not use the function `inet_addr()`. In general, if you can, avoid using 255.255.255.255.

## 4.9 UDP multicasting and sendto

Multicasting is based on the same idea of broadcasting but it allows a program to send one datagram to a set of hosts, including hosts across the sub-network. Multicasting is essentially a method to implement a virtual network (referred to as a “group”) and broadcast within the group. Every IP multicast has a group address. It is not necessary to be a member of the group to send a datagram to the group.

Multicast is not supported by old routers and is an optional feature in IPv4.

To send a multicast datagram one sends a regular datagram to the multicast address. The only difference is that one has to set the Time-To-Live option of the datagram using the `setsockopt`

```
unsigned char ttl=1; // 0,1,32,64,128 or 255
setsockopt(sfd,IPPROTO_IP,IP_MULTICAST_TTL,&ttl,sizeof(ttl));
```

If `ttl` is set to 0 the multicast is restricted to the host; if `ttl` is set to 1 the multicast is restricted to the subnet; if `ttl` is set to 32 to the site, 64 to the region, 128 to the continent and 255 to the world. Yes, you sent `ttl` to 255 and you are sending the datagram to everybody.

Note that multicast address are class D addresses (range 224.0.0.0 to 239.255.255.255).

## 4.10 UDP multicast and recvfrom

To receive a multicast address one has:

- to be in the are reached by the datagram (in the same subnet if tt was set to 1 by the sender);
- join the group (identified by a multicast\_ip\_address and a multicast\_port\_number).

One joins the group by

```
// set socket reusable
static int reuse_on=1;
setsockopt(sfd,SOL_SOCKET,SO_REUSEADDR,&reuse_on,sizeof(reuse_on));
// setup local address and bind to port
local_address.sin_family=AF_INET;
local_address.sin_port=htons(multicast_port_number);
local_address.sin_port.s_addr=htonl(INADDR_ANY);
if(bind(sfd,(struct sockaddr*)&local_address,
        sizeof(local_address))<0)
    exit_message(1,"bind");
// request kernel to join group
struct ip_mreq mreq;
mreq.imr_multiaddr.s_addr=inet_addr(multicast_ip_address);
mreq.imr_interface.s_addr=htonl(INADDR_ANY);
setsockopt(sfd,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq));
```

Now you recvfrom in the usual way.

Note that a multicast protocol is identified by a port (to which the clients/receivers must bind) and a multicast address (the same address used by the server to multicast a message and used by the clients/receivers to join the group).

## 5 Signals, Processes and Threads

Relevant commands:

- **alarm**: sets an alarm
- **fork**: fork a process
- **system**: executes a shell command
- **exec1**: replaces process with a shell command
- **pthread\_create**, **pthread\_detach**, **pthread\_join**, **pthread\_cancel**: threads
- **getpeerbyname**: get the address and port of the peer connected to a socket (typically used in daemon).

## 5.1 Using Alarms

Alarm is a system to ask the OS to send us a signal SIGALRM after xxx seconds in time.

To set an alarm:

```
// create handler
void alarm_handler(int signum) {
    // this is called when the alarm occurs
    // do something
}

...
// register signal SIGALRM
struct sigaction action;
action.sa_handler=alarm_handler;
if(sigemptyset(&action.sa_mask)<0)
    exit_message(1,"sigemptyset");
if(sigaddset(&action.sa_mask,SIGALRM))
    exit_message(1,"sigaddset");
if(sigaction(SIGALRM,&action,NULL)<0)
    exit_message(1,"sigaction");
// set alarm
int seconds=5
alarm(seconds);
// do something while you wait for the alarm
```

...

Note that all blocking functions return when the alarm occurs. Each of them may return in a different way. Do not set alarms that may go off during console IO functions.

## 5.2 Ignore Signals

To ignore a signal, just register the signal with the macro SIG\_IGN as handler.

```
...
// ignore signal SIGALRM
struct sigaction action;
action.sa_handler=SIG_IGN;
if(sigemptyset(&action.sa_mask)<0)
    exit_message(1,"sigemptyset");
if(sigaddset(&action.sa_mask,SIGALRM))
    exit_message(1,"sigaddset");
if(sigaction(SIGALRM,&action,NULL)<0)
    exit_message(1,"sigaction");
...
```

## 5.3 Blocking a signal

Sometimes you need to block signals. For example, if you set an alarm and you perform IO you may want to block the alarm during the IO. In the same fashion if you are using signal driven IO and you expect a SIGIO you may want to block the SIGIO during some other IO. The process of blocking one or more signals usually works as follows:

- Register the present signal mask
- Block all signals that need to be blocked
- Perform the operation (...) that should not be interrupted by the signal
- Check if a signal is pending and take appropriate action
- Restore the initial signal mask

```

// register present signal mask in oset
// select new signal mask in set
sigset_t set, oset;
if(sigemptyset(&set)<0)
    exit_message(1,"sigemptyset");
// add one or more signals to block
if(sigaddset(&set,SIGALRM)<0)
    exit_message(1,"sigaddset");
sigprocmask(SIG_BLOCK,&set,&oset);
// perform the operation in question
...
// check if signal is pending
sigset_t pending;
sigpending(&pending);
// check for each possible pending signal
if(sigismember(&pending,SIGALRM) {
    // deal with pending signal
}
// restore signal mask (IMPORTANT)
sigprocmask(SIG_BLOCK,&oset,NULL);

```

## 5.4 Creating a process

The simplest way to do concurrent programming is to create a child process that runs in parallel with the parent process. For example, one process does socket IO and another process does console IO at the same time. The console IO is not blocked by the blocking socket IO functions.

To create a child process:

```

pid=fork();
if(pid<0) {
    exit_message(1,"fork");
} else if(pid) {
    // parent process
    // (pid contains the pid of child)
} else {
    // child process
}

```

```
// code executed by both parent and child
```

This is okay. However, interprocess communications may be difficult, processes may become zombies and you have to use signals. Programs that fork, if written well, are more complex than programs with threads.

## 5.5 Killing zombies

When a child process dies before the parent, it sends a signal SIGCHLD to the parent and becomes a zombie. The zombie status means that the OS does not clean-up the memory used by the process and is waiting for the parent to authorize the cleanup. To prevent a child from becoming a zombie the parent should kill the child when it receives SIGCHLD. To achieve this, before forking the parent you should:

```
void zombie_handler(int s) {
    pid_t pid;
    int status;
    pid=wait(&status);
    // pid is the pid of dead child
    // status holds the return error of the dead child
}
...
// before forking register handler
struct sigaction action;
action.sa_handler=zombie_handler;
if(sigemptyset(&action.sa_mask)<0)
    exit_message(1,"sigemptyset");
if(sigaddset(&action.sa_mask,SIGCHLD))
    exit_message(1,"sigaddset");
if(sigaction(SIGCHLD,&action,NULL)<0)
    exit_message(1,"sigaction");
```

## 5.6 Creating a deamon process (fork twice)

A deamon is a process not connected to console and with no parent shell. To detach a process from shell fork twice.

```
pid=fork()
```

```

if(pid<0) exit_message(1,"fork");
if(pid>0) exit(0);
pid=fork()
if(pid<0) exit_message(1,"fork");
if(pid>0) exit(0);
// we are now detached from shell

To close the console IO close stdin, stderr and stdout.

close(fileno(stdin));
close(fileno(stderr));
close(fileno(stdout));

```

## 5.7 Running commands withing programs: system and exec

Sometimes you want to run a shell command within a program. For example, to list all running processes,

```

// do something
...
system("/usr/bin/ps aux");
// do something else
...

```

To perform the same operation and substitute the call ps for the present process:

```

// do something
...
execl("/usr/bin/ps","aux",0); // all arguments ended by 0
// it never makes it here
...

```

## 5.8 Creating a deamon process for inetd

The inetd is a system deamon that accepts connection on multiple ports to serve a set of registered services (ssh, tcp, etc.). When a connection occurs a new process is forked, the socket is renamed 0 and the proper application is execl-ed. Here is a service (built for inetd) that sends to the connected peer the peer's ip address:

```

int main(int argc, char** argv) {
    string message="You are ";
    struct sockaddr_in remote;
    int addrlen;
    getpeerbyname(0,(struct sockaddr*) remote, (socklen_t*) &addrlen);
    message=message+inet_ntop(AF_INET,remote.sin_addr);
    send(0,message.c_str(),message.length(),0);
    close(0);
}

```

The above process did not create the socket itself, nor did it call bind, listen and accept. It is started by inetd with a peer already connected to the socket with `sfd=0`. To determine the ip address, port and protocol of the peer connected to 0, the program needs to call `getpeerbyname(0,...)`.

After a valid inetd service is created and compiled (let's say in `/usr/local/bin/myservice.exe`) the service has to be registered with inetd. This is a 3-step process:

- Pick a port for the service that does no conflict with other services (let's say 1234).
- Tell inetd to listen from that port.
- Associate your compiled program to the service.
- Restart inetd.

Step 1 implies looking into `/etc/services` and finding an unused port number.

Step 2 implies editing the file `/etc/services` and adding a new line/entry:

```
\texttt{\%myservicename 1234/tcp}
```

Step 3 implies editing the file `/etc/inetd.conf` and adding a new line/entry:

```
myservicename stream tcp nowait nobody /usr/local/bin/myservice.exe
```

## 5.9 Locking a file

When one writes to a file one may want to lock the file so that no other process can write to the same file. For an integer file descriptor `fd`:

```

// lock fd for writing
fcntl(fd,F_SETLK,F_WRLCK);
// use fd
...
// unlock fd
fcntl(fd,F_SETLK,F_UNLCK);

```

To lock for reading F\_RDLCK and to lock for both reading and writing F\_RWLCK.

## 5.10 Getting file info

Use the command "stat" as follows:

```

struct stat file_info;
stat("path/myfile.dat",&file_info);

```

Now struct stat file\_info contains the following information about the file:

```

struct stat {
    dev_t      st_dev;      /* device inode resides on */
    ino_t      st_ino;      /* inode's number */
    mode_t     st_mode;     /* inode protection mode */
    nlink_t    st_nlink;    /* number of hard links to the file */
    uid_t      st_uid;      /* user-id of owner */
    gid_t      st_gid;      /* group-id of owner */
    dev_t      st_rdev;     /* device type, for special file inode */
    struct timespec st_atimespec; /* time of last access */
    struct timespec st_mtimespec; /* time of last data modification */
    struct timespec st_ctimespec; /* time of last file status change */
    off_t      st_size;     /* file size, in bytes */
    quad_t     st_blocks;   /* blocks allocated for file */
    u_long     st_blksize;  /* optimal file sys I/O ops blocksize */
    u_long     st_flags;    /* user defined flags for file */
    u_long     st_gen;      /* file generation number */
};

```

## 5.11 Creating threads

This is an example that uses two threads.

```

void* f(void* p) {
    sleep(2);
    cerr << "Hello from a thread\n";
    pthread_exit(NULL);
    return 0;
}

int main() {
    pthread_t thread_number;
    pthread_attr_t thread_attr;
    pthread_attr_init(&thread_attr);
    pthread_attr_setdetachstate(&thread_attr, PTHREAD_CREATE_JOINABLE);

    cout << "Starting thread\n";

    if(pthread_create(&thread_number, &thread_attr, f, (void*) NULL))
        exit_message(1,"pthread_create: Unable to create thread");

    cout << "Thread started. waiting ... \n";

    int status;
    if(pthread_join(thread_number,(void**) &status))
        exit_message(1,"pthread_join: Unable to join thread");

    cout << "Thread joined\n";
}

```

The advantage of two threads over two processes is that threads share the same memory and see the same variables. This makes communication between threads easier than communication between processes. Because they share the same memory one has to prevent two threads from writing to the same variable simultaneously. To avoid this, use the mutex functions. They work very much like a file locking but they lock variables, not files.

## 6 Appendix A - Working Examples

### 6.1 Gethostbyname

This program returns the IP address of a host.

```
// BEGIN FILE: gethostbyname.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc<2) {
        cout << "usage: name host_name\n";
        exit(1);
    } else {

        struct hostent* hptr=gethostbyname(argv[1]);
        if(hptr==0) cout << "erreor\n";
        if(hptr->h_length!=4) cout << "it is ipv6\n";

        char** q=hptr->h_aliases;
        cout << "Aliases:\n";
        while(*q!=0) {
            cout << *q << endl;
            q++;
        }

        char** p=hptr->h_addr_list;
        cout << "Addresses:\n";
        while(*p!=0) {
            cout << inet_ntop(family,*((struct in_addr*)*(p))) << endl;
            p++;
        }
    }
    return 0;
}
```

### 6.2 TCP server with fork

This server sends a buffer containing "Here I am" to the clients that connect.

```

// BEGIN FILE: plain_server.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc<2) {
        cout << "USAGE: server local_port\n";
        return 0;
    }
    sockaddr_in clientaddr;
    sockaddr_in servaddr;
    int port=atoi(argv[1]);
    int sfd=socket(AF_INET,SOCK_STREAM,0);
    memset(&servaddr,0,sizeof(sockaddr_in));
    servaddr.sin_family=AF_INET;
    servaddr.sin_port=htons(port);
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    if(bind(sfd, (sockaddr*) &servaddr, sizeof(sockaddr_in)))
        exit_message(1,"bind: failure to connect");
    if(listen(sfd,1024))
        exit_message(1,"listen: max connections exceeded");

    char data[]="HERE I AM\n";
    int connfd;
    int length[sizeof(clientaddr)];
    int pid;
    while(1) {
        cout << "waiting...\n";
        connfd=accept(sfd,(sockaddr*) &clientaddr, (socklen_t*) &length);

        pid=fork();
        if(pid==0) {
            close(sfd);
            cout << "serving...\n";
            send(connfd,data,strlen(data),0);
            close(connfd);
            cout << "done!\n";
            exit(0); // because of fork this will make a zombie!
        }
    }
}

```

```
}
```

### 6.3 TCP client

This client connects to a server and expects to receive a '\n' terminated string from the server.

```
// BEGIN FILE: plain_client.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc<3) {
        cout << "USAGE: client address port\n";
        return 0;
    }
    sockaddr_in remote_sockaddr;
    int sfd;
    int port=atoi(argv[2]);
    sfd=socket(AF_INET,SOCK_STREAM,0);
    memset(&remote_sockaddr,0,sizeof(sockaddr_in));
    remote_sockaddr.sin_family=AF_INET;
    remote_sockaddr.sin_port=htons(port);
    inet_pton(AF_INET,argv[1],&remote_sockaddr.sin_addr);
    int ce=connect(sfd,(sockaddr*)&remote_sockaddr, sizeof(sockaddr_in));
    if(ce!=0 && (errno==ECONNREFUSED || errno==ETIMEDOUT)) {
        close(sfd);
    } else {

        int bytes;
        string s;
        char c;
        while(1) {
            bytes=recv(sfd,&c,1,0);
            if(bytes<0) {
                cerr << "CONNECTION ERROR\n";
                break;
            } else if(bytes==0) {
                cerr << "END OF INPUT\n";
                break;
            }
        }
    }
}
```

```

    }
    else if(c=='\n') break;
    s=s+c;
}
cout << s << endl;
close(sfd);
}
return 0;
}

```

## 6.4 UDP server

This UDP server waits to receive datagrams. It displays each datagram it receives.

```

// BEGIN FILE: dgram_server.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc!=2) {
        cout << "USAGE: ./a.out serving_port\n";
        return 0;
    }

    int sockint, s, namelen, client_address_size;
    struct sockaddr_in client, server;
    char buf[32];

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if(s==-1) exit_message(1,"Socket was not created.\n");

    server.sin_family = AF_INET;
    server.sin_port = htons(atoi(argv[1]));
    server.sin_addr.s_addr = INADDR_ANY;

    if(bind(s,(const sockaddr*)&server,sizeof(server))<0)
        exit_message(3,"Error binding server");

    client_address_size = sizeof( client );

```

```

printf("Waiting for a message to arrive.\n");
if(recvfrom(s,buf,sizeof(buf),0,(struct sockaddr *)&client,
            (socklen_t*)&client_address_size)<0)
    exit_message(4,"Error in receive from\n");

cout << "I received: " << buf << endl;
close(s);
}

```

## 6.5 UDP client

This UDP client sends a datagram “Hello” to the server.

```

// BEGIN FILE: dgram_client.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc!=3) {
        cout << "Usage: ./a.out remote_address remote_port\n";
        return 0;
    }

    int s;
    struct sockaddr_in server;
    char buf[32];

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if(s == -1) exit_message(1,"Socket was not created.\n");
    server.sin_family = AF_INET;
    server.sin_port = 0;
    server.sin_addr.s_addr = INADDR_ANY;
    if(bind(s,(const sockaddr*)&server, sizeof(server))<0)
        exit_message(3,"Error binding server.\n");

    server.sin_family = AF_INET;
    server.sin_port = htons(atoi(argv[2]));
    server.sin_addr.s_addr = inet_addr(argv[1]);

    strcpy( buf, "Hello" );

```

```

    cout << "Sending data to the socket.\n";
    sendto(s,buf,(strlen(buf)+1),0,
           (const sockaddr*)&server,sizeof(server));
    cout << "Data has been sent to the socket\n";
    close(s);
}

```

## 6.6 UDP server using non-blocking IO

A UDP server that uses non-blocking IO.

```

// BEGIN FILE: dgram_server_unblock.cpp
#include "mdp_all.h"
int main(int argc, char** argv) {
    if(argc!=2) {
        cout << "USAGE: ./a.out serving_port\n";
        return 0;
    }

    int sockint, s, namelen, client_address_size;
    struct sockaddr_in client, server;
    char buf[32];

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if(s== -1) exit_message(1,"Socket was not created.\n");

    server.sin_family = AF_INET;
    server.sin_port = htons(atoi(argv[1]));
    server.sin_addr.s_addr = INADDR_ANY;

    if(bind(s,(const sockaddr*)&server,sizeof(server))<0)
        exit_message(3,"Error binding server");

    fcntl(s,F_SETFL,O_NONBLOCK);

    client_address_size = sizeof( client );
    printf("Waiting for a message to arrive.\n");
    while(1) {

```

```

    sleep(1);
    cout << "tick\n";
    int e=recvfrom(s,buf,sizeof(buf),0,
                   (struct sockaddr *)&client,
                   (socklen_t*)&client_address_size);
    if(e==EWOULDBLOCK) exit_message(1,"socket error\n");
    else if(e>0) {
        cout << "I received: " << buf << endl;
        break;
    }
}

close(s);
}

```

## 6.7 UDP server using asynchronous IO

A UDP server that uses asynchronous IO.

```

// BEGIN FILE: dgram_server_sigio.cpp
#include "mdp_all.h"

int s;
void sigio_handler(int signum) {
    char buf[32];
    struct sockaddr_in client;
    int client_address_size = sizeof( client );
    cout << "I got a signal SIGIO\n";
    int e=recvfrom(s,buf,sizeof(buf),0,
                   (struct sockaddr *)&client,
                   (socklen_t*)&client_address_size);
    cout << e << endl;
    if(e==EWOULDBLOCK) exit_message(1,"socket error\n");
    else if(e>0)
        cout << "I received: " << buf << endl;
}

int main(int argc, char** argv) {

```

```

if(argc!=2) {
    cout << "USAGE: ./a.out serving_port\n";
    return 0;
}

int sockint, namelen;
struct sockaddr_in server;

s = socket(AF_INET, SOCK_DGRAM, 0);
if(s== -1) exit_message(1,"Socket was not created.\n");

server.sin_family = AF_INET;
server.sin_port = htons(atoi(argv[1]));
server.sin_addr.s_addr = INADDR_ANY;

if(bind(s,(const sockaddr*)&server,sizeof(server))<0)
    exit_message(3,"Error binding server");

fcntl(s,F_SETFL,O_NONBLOCK);
fcntl(s,F_SETOWN,getpid());
fcntl(s,F_SETFL,FASYNC);

cout << "Waiting for a message to arrive.\n";

struct sigaction action;
action.sa_handler=sigio_handler;
if(sigemptyset(&action.sa_mask)<0)
    exit_message(1,"sigemptyset");
if(sigaddset(&action.sa_mask,SIGIO))
    exit_message(1,"sigaddset");
if(sigaction(SIGIO,&action,NULL)<0)
    exit_message(1,"sigaction");

while(1) {
    sleep(1);
    cout << "tick\n";
}

```

```
    close(s);
}
```

## 6.8 UDP server using poll

A UDP server that uses poll.

```
// BEGIN\ FILE: dgram_server_poll.cpp
#include "mdp_all.h"

int main(int argc, char** argv) {
    if(argc!=2) {
        cout << "USAGE: ./a.out serving_port\n";
        return 0;
    }

    int sockint, s, namelen, client_address_size;
    struct sockaddr_in client, server;
    char buf[32];

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if(s== -1) exit_message(1,"Socket was not created.\n");

    server.sin_family = AF_INET;
    server.sin_port = htons(atoi(argv[1]));
    server.sin_addr.s_addr = INADDR_ANY;

    if(bind(s,(const sockaddr*)&server,sizeof(server))<0)
        exit_message(3,"Error binding server");

    struct pollfd fds[2];
    fds[0].fd=fileno(stdin);
    fds[0].events=POLLRDNORM;
    fds[1].fd=s;
    fds[1].events=POLLRDNORM | POLLERR;

    string text;
    char c;
```

```

cout << "type something:\n";
while(1) {
    int n=poll(fds,2,-1);
    if(n==0) exit_message(0,"timeout");
    if(n==-1) exit_message(1,"auch!");

    if(fds[0].revents) {
        read(fileno(stdin),&c,1);
        if(c=='\n') {
            cout << "You typed: " << text << endl;
            cout << "type something:\n";
            text="";
        } else {
            text=text+c;
        }
    }

    if(fds[1].revents) {
        int e=recvfrom(s,buf,sizeof(buf),0,
                      (struct sockaddr *)&client,
                      (socklen_t*)&client_address_size);
        if(e==EWOULDBLOCK) exit_message(1,"socket error\n");
        cout << "From: " << inet_ntoa(client.sin_addr) << endl;
        cout << "I received: " << buf << endl;
    }
}
close(s);
}

```

## 6.9 UDP server using select

Same as above but using select instead of poll.

```

// BEGIN FILE: dgram_server_select.cpp
#include "mdp_all.h"

int max(int a, int b) {

```

```

    if(a>b) return a;
    return b;
}

int main(int argc, char** argv) {
    if(argc!=2) {
        cout << "USAGE: ./a.out serving_port\n";
        return 0;
    }

    int sockint, s, namelen, client_address_size;
    struct sockaddr_in client, server;
    char buf[32];

    s = socket(AF_INET, SOCK_DGRAM, 0);
    if(s== -1) exit_message(1,"Socket was not created.\n");

    server.sin_family = AF_INET;
    server.sin_port = htons(atoi(argv[1]));
    server.sin_addr.s_addr = INADDR_ANY;

    if(bind(s,(const sockaddr*)&server,sizeof(server))<0)
        exit_message(3,"Error binding server");

    fd_set rset;
    fd_set eset;

    string text;
    char c;
    int maxn=max(fileno(stdin),s)+1;
    cout << "type something:\n";
    while(1) {

        FD_ZERO(&rset);
        FD_SET(fileno(stdin),&rset);
        FD_SET(s,&rset);
        FD_ZERO(&eset);
        FD_SET(fileno(stdin),&eset);

```

```

FD_SET(s,&eset);

int n=select(maxn,&rset,0,&eset,0);
if(n==0) exit_message(0,"timeout");
if(n==-1) exit_message(1,"auch!");

if(FD_ISSET(fileno(stdin),&rset)) {
    read(fileno(stdin),&c,1);
    if(c=='\n') {
        cout << "You typed: " << text << endl;
        cout << "type something:\n";
        text="";
    } else {
        text=text+c;
    }
}

if(FD_ISSET(s,&rset) ||
   FD_ISSET(s,&eset)) {
    int e=recvfrom(s,buf,sizeof(buf),0,
                   (struct sockaddr *)&client,
                   (socklen_t*)&client_address_size);
    if(e==EWOULDBLOCK) exit_message(1,"socket error\n");
    cout << "From: " << inet_ntoa(client.sin_addr) << endl;
    cout << "I received: " << buf << endl;
}
close(s);
}

```

## 6.10 Setting an alarm

A program that asks the OS to send an alarm to the program.

```

// BEGIN FILE: test_alarm.cpp
#include "mdp_all.h"

void handler(int s) {

```

```

if(s==SIGALRM)
    cout << "I got a SIGALRM signal\n";
}

int main() {
    struct sigaction action;
    action.sa_handler=handler;
    if(sigemptyset(&action.sa_mask)<0)
        exit_message(1,"sigemptyset");
    if(sigaddset(&action.sa_mask,SIGALRM))
        exit_message(1,"sigaddset");
    if(sigaction(SIGALRM,&action,NULL)<0)
        exit_message(1,"sigaction");

    cout << "Setting the alarm\n";
    alarm(5);
    cout << "Waiting\n";
    if(sleep(10)!=0)
        cout << "I have been awakened\n";
    else
        cout << "I woke up naturally\n";
    cout << "Done\n";
    return 0;
}

```

## 6.11 Killing zombies

A program that forks and prevents its child from becoming a zombie.

```

// BEGIN FILE: test_zombie.cpp
#include "mdp_all.h"

void handler(int s) {
    if(s==SIGCHLD) {
        pid_t pid;
        int status;
        pid=wait(&status);
        cout << "Process " << pid << " terminated with status " << status << "\n";
    }
}

```

```

        }
    }

int main() {
    int pid=fork();
    if(pid==0) {
        // the child process dies!
        sleep(3);
        exit(1);
    } else {
        cout << "My child has pid=" << pid << endl;
        struct sigaction action;
        action.sa_handler=handler;
        if(sigemptyset(&action.sa_mask)<0)
            exit_message(1,"sigemptyset");
        if(sigaddset(&action.sa_mask,SIGCHLD))
            exit_message(1,"sigaddset");
        if(sigaction(SIGCHLD,&action,NULL)<0)
            exit_message(1,"sigaction");

        if(sleep(10)!=0)
            cout << "I have been woken up\n";
        else
            cout << "I woke up naturally\n";
    }
    return 0;
}

```

## 7 Appendix B - OSI Model

The OSI, or Open System Interconnection, model defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

## **7.1 Physical**

(Layer 1) This layer conveys the bit stream - electrical impulse, light or radio signal – through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects. Fast Ethernet, RS232, and ATM are protocols with physical layer components.

## **7.2 Data Link**

(Layer 2) At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and frame synchronization. The data link layer is divided into two sublayers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sublayer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.

## **7.3 Network**

(Layer 3) This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internet.

## **7.4 Transport**

(Layer 4) This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer. working, error handling, congestion control and packet sequencing.

## **7.5 Session**

(Layer 5) This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and dialogues between the applications at each end. It deals with session and connection coordination.

## 7.6 Presentation

(Layer 6) This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the syntax layer.

## 7.7 Application

(Layer 7) This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services. Telnet and FTP are applications that exist entirely in the application level. Tiered application architectures are part of this layer.

# 8 Appendix B - UNIX Commands

- ping
- arp
- netstat
- telnet
- tcpdump
- ifconfig
- ifup

## **Part VI**

# **Other Documents**



# Publishing Linked Data Using web2py

Chris Baron\* and Massimo Di Pierro†

School of Computing, DePaul University, Chicago, IL

October 28, 2010

## Abstract

In this paper we describe a software plugin to publish any database as Linked Data using the web2py Database Abstraction Layer. It works with new database tables as well as with existing ones. Both simple and complex RDF structures can be expressed with ease and the plugin automatically creates web services to expose the data. The Database Abstraction Layer writes the SQL dynamically and transparently. It supports ten different database back-ends, including Oracle and Google Big Table. This approach leverages on the capabilities of the framework, including a Role Based Access Control system with pluggable authentication methods. Our system runs on any platform supported by Python including the Google App Engine cloud platform. We believe our approach can contribute to make programming for the semantic web more accessible.

## 1 Introduction

The amount of information posted online is rapidly growing beyond the point where humans can comprehend it and we will need automated agents to be able to find, collect, and summarize information for us. Most of this information already exists in structured form, in relational databases.

It has been shown that the growth of traditional Web pages supersedes the growth of semantic representations and most web developers are unaware of the Semantic Web. As a result, a significant portion of Web data is unavailable to semantic search engines. Despite the fact that there are many tools already available to program for the Semantic Web we believe there are not enough tools that make it easy. This is the problem we try to address in this paper.

Modern web applications rarely use raw SQL to communicate with databases but, instead, they use a Object Relational Mapper (ORM) or a Database Abstraction Layer (DAL) to map programming objects into database objects. The DAL write the SQL dynamically in the specific dialect of the database back-end.

---

\*topher.baron@gmail.com

†mdipierro@cs.depaul.edu

This approach ensures portability of the application over multiple databases and prevents SQL Injection attacks.

The purpose of this paper is to show how we can take advantage of one of these Web Frameworks and its DAL (specifically we will use WEB2PY [3]) to annotate database tables, fields and relations with OWL and expose the data in RDF. The proposed system is powerful enough to translate one-to-many and many-to-many database relations into both simple and complex RDF expressions and expose them via a RESTful web service.

At the time of writing, the WEB2PY DAL supports transparently SQLite, MySQL, PostgreSQL, Oracle, DB2, Informix, FireBird, Ingres, MSSQL and Google Big Table (the non-relational database provided by Google App Engine).

Using the technology described here any of the relational databases listed above can become a node of the semantic web. Not all tables and records will be exposed in RDF, only those appropriately annotated.

WEB2PY also includes a Role Based Access Control mechanism that can be enforced on any object.

We believe this mechanism simplifies the process of exposing data in RDF.

## 2 Linked Data

Semantic Web is a group of methods and technologies to allow machines to understand the meaning - or “semantics” - of information on the World Wide Web.

The Web Ontology Language (OWL) is a collection of knowledge representation languages for describing ontologies. These languages are characterised by formal semantics often expressed in RDF (and XML-based serializations).

The World Wide Web Consortium (W3C) created a Semantic Web Education and Outreach (SWEO) interest group that expired in 2008 [4]. The group started the Linking Open Data project with the goal of building a data commons for “making various open data sources available on the Web as RDF and by setting RDF links between data items from different data sources”. According to the W3C, over 4.7 billion RDF triples have been published to date, and they are connected by about 142 million RDF links [5].

Examples of Linked Data Ontologies are:

- dc: Dublin Core. A set of text elements to catalog generic online resources [6].
- sioc: Semantically-Interlinked Online Communities ontology, which aims to describe relation between online communities [7].
- foaf: Friend-of-a-Friend ontology which describes relations between people [8].

Here is a basic example of how to describe a blog entry using SIOC:

```

1 <sioc:Post rdf:about="http://...">
2   <dcterms:title>Creating connections between ...</dcterms:title>
3   <dcterms:created>2006-09-07T09:33:30Z</dcterms:created>
4   <sioc:has_container rdf:resource="http://..."/>
5   <sioc:has_creator>
6     <sioc:User rdf:about="http://..."/>
7       <rdfs:seeAlso rdf:resource="http://..."/>
8     </sioc:User>
9   </sioc:has_creator>
10  <sioc:content>SIOC provides a .... </sioc:content>
11  <sioc:topic rdfs:label="Semantic Web" rdf:resource="http://..."/>
12 </sioc:Post>

```

Notice how every resource is identified by a URL and how the tag name and attributes specify the relations between resources.

### 3 web2py

Web Frameworks are collections of libraries that abstract many of the basic tasks of web applications such as parsing cookies, managing sessions, handling persistence, dispatching HTTP requests and mapping them into function calls, validating input for security, logging errors, pooling database connections for speed, abstracting details about the database back-end, and more. There are Web Frameworks written in almost every programming language.

WEB2PY is one of these web frameworks, one of the few born in an academic environment. WEB2PY is written in the Python programming language and it is programmable in Python. It is designed with three primary goals: be easy to use, force developers to follow good software engineering practices, provide strong security (prevents SQL Injections and XSS vulnerabilities).

WEB2PY does not require installation and includes a web server with SSL capabilities, the SQL(ite) relational database, a web based IDE (with editor, testing and debugging capabilities), a template language similar to PHP, a Model View Controller design, a Database Abstraction Layer that writes the SQL for you (it also CREATE, and ALTER tables as necessary), a web based administrative interface for the database, a Role Based Access Control system with pluggable authentication methods, a mechanism for automatic generation of forms with validation of all form fields, a plugin system, and an internationalization system. It works with multiple web servers (for example Apache) and databases. It is the only framework that allows you to deploy apps on the Google Cloud without the need to program in the native Google API.

In WEB2PY an application consists of files that are divided in the following basic groups<sup>1</sup>:

- **Models:** These are files that contain a description of the data representation using the Database Abstraction Layer. Typically a Model contains a definition of a set of tables and their relations. WEB2PY generates all

---

<sup>1</sup>An application may also contain static files, modules, plugins, private files, session files, cache files, error logs, and translation files to deal with internationalization.

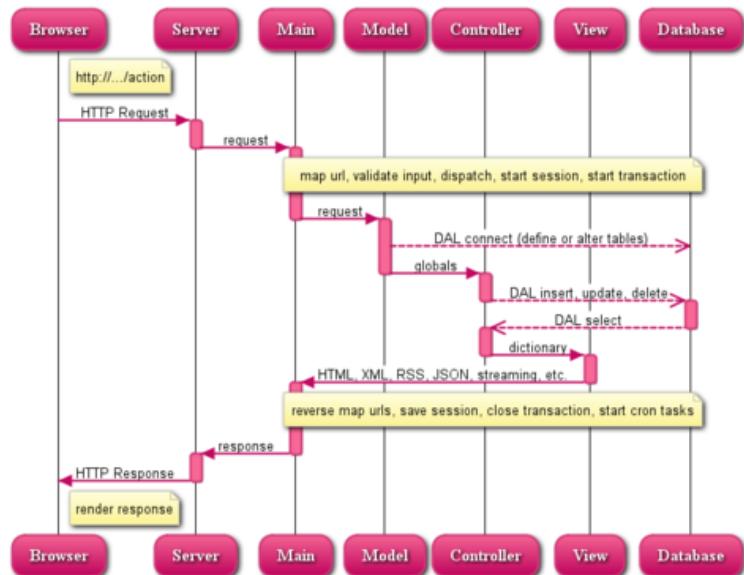


Figure 1: Information flow inside the WEB2PY framework. The proposed annotation mechanism resides in the Model and exposes the Linked Data as a RESTful web service via the provided controller `plugin_rdf.py`.

required SQL to create the tables if they do not exist, alter them if needed, and query them. It also automatically creates a web based interface to the database from models.

- **Controllers:** They contain the logic and control flow of the application. A controller may contain one or more functions called “actions”. The framework maps URLs into actions.
- **Views:** They describe how to serialize the data returned by the actions, for example in HTML, XML, JSON, CSS, RSS, and/or RDF.

Each group of files is located in its own folder under the application folder. When the web server receives a request from a client, it runs the models, then maps the URL of the HTTP request into an action, runs the action, renders the output of the action into - for example - HTML using the associated view for the specific requested format. The resulting HTML is returned to the requesting client.

The information flow within the framework is graphically depicted in fig. 1.

For the purpose of this paper we only need to understand the syntax of Model files. That is where all the RDF annotation will be done. In order to expose the RDF data as a service we created a single controller file called “plugin\_rdf.py”. This file does not need to be edited, it just needs to be dropped in the controllers folder for the application or installed as a plugin and, for practical purposes, it can be treated as a black-box that extends WEB2PY functionality.

We refer to the official WEB2PY documentation for further details [3].

From now on we will assume we have a WEB2PY sever running on localhost (127.0.0.1) on port 8000 and we are working on an application called “semantic”. The content of this application includes exclusively default files created by WEB2PY and the files described below.

Some screenshots for WEB2PY are shown in fig. 2.

## 4 Annotating the Relational Database

In this section we provide a practical example of how to create a model, annotate its tables, fields and relations with OWL, and expose it via a RESTful RDF service.

In the following we will adopt the convention that table names are singular.

The example we consider is a Blogging application that defines the following tables:

- **Post:** It stores each blog post’s text content, its author, and its title.
- **Comment:** A comment is associated with an author and the body of the comment.
- **Tag:** It is simply a collection of keywords, not to be confused with RDF tags. These tags (for example “Obama”, “Politics”, “Democrats”) are

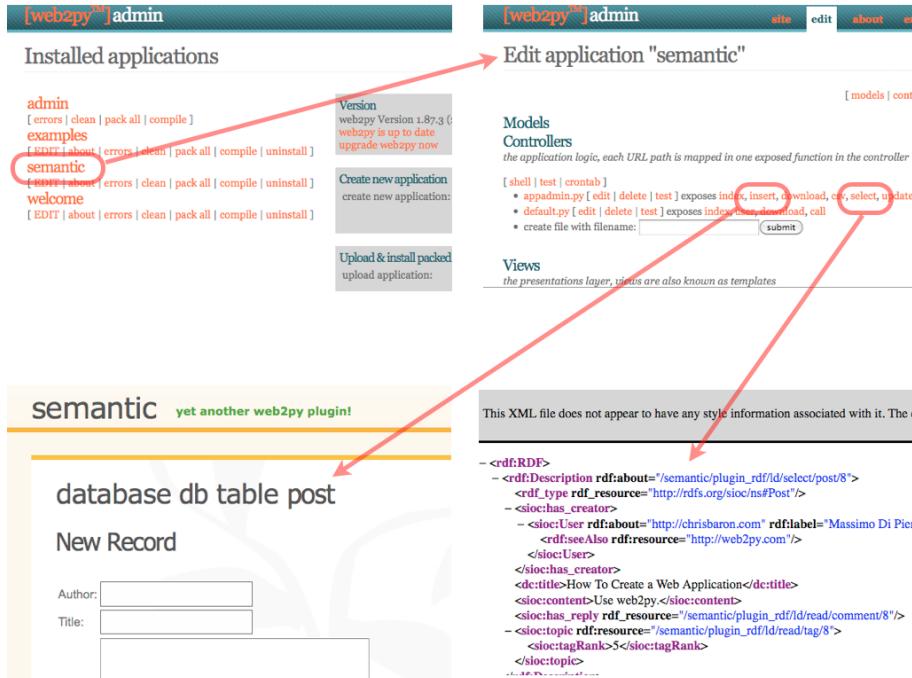


Figure 2: The top-left image shows the main page of the web based IDE, it lists installed applications. The top-right page is also part of the web based IDE and it lists the content of our example application. The bottom-left image shows the database administrative interface for our example DB. The bottom-right image shows an example Linked Data output as rendered by Firefox.

associated with blog posts as a quick categorization tool: blog readers can find posts tagged with topics they are interested in.

- **Post-Tag-Link:** A post may have multiple tags and a tag may be associated to multiple posts. This table implements such linkage. The link table also has a ranking field since some tags apply to a certain post more than other posts.

The relation between posts and Comments is one-to-many. The relation between posts and tags, as provided by the post\_tag\_link table is many-to-many.

This is how the aforementioned tables are defined in a WEB2PY model (which we will call “models/db.example.py”):

```

1 db.define_table('post',
2                 Field('author', 'string'),
3                 Field('title', 'string'),
4                 Field('body', 'text'))
5
6 db.define_table('comment',
7                 Field('post_id', db.post),
8                 Field('author', 'string'),
9                 Field('body', 'text'))
10
11 db.define_table('tag',
12                  Field('name', 'string'))
13
14 db.define_table('post_tag_link',
15                  Field('post_id', db.post),
16                  Field('tag_id', db.tag),
17                  Field('ranking', 'integer'))
```

Here `db` represents a database connection object. Each table has a name and fields. The first argument of a Field is the field name, the second argument is the field type. When the field type is another table than it is a foreign key reference field. Once a table is defined, WEB2PY creates the table if it does not exist or alters it as necessary.

Until this point we have done nothing specific about the Semantic Web yet. In order to expose the data in these tables as RDF we annotate them with OWL. This is done in 3 steps.

## 4.1 Step 1: exposing the service

The first step consists of installing the following plugin into the application:

```

1 web2py.plugin.rdf.w2p
from
1 http://web2py.com/examples/static/web2py.plugin.rdf.w2p
```

This is done from the administrative interface of web2py (at the bottom of the top-right screenshot in fig. 2). The plugin contains a controller file “plugin\_rdf.py”. This will create and expose all required services which can be accessed via a RESTful interface:

```

1 http://127.0.0.1:8000/semantic/plugin_rdf/ld/tables
2 http://127.0.0.1:8000/semantic/plugin_rdf/ld/select/<table>
3 http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/<table>/<record>

```

Here “semantic” is the application name; “plugin\_rdf” is the plugin controller; “ld” is the Linked Data action defined by the controller; “table”, “select” and “read” are the functions performed by the action; <table> and <record> refer to a table annotation as explained below.

## 4.2 Step 2: annotating fields

The second step consists of annotating the individual fields of each table. We annotate the fields of the `db.post` table with ontologies by adding the following code immediately after the table definition:

```

1 db.post.author.rdf = 'sioc:has_creator'
2 db.post.title.rdf = 'dc:title'
3 db.post.body.rdf = 'sioc:content'

```

The left-hand-side of each expression refers to the optional RDF attribute of a table column. The right-hand-side is the OWL property that the column refers to . The prefix refers to the ontology namespace: “sioc:” and “dc” respectively.

## 4.3 Step 3: annotating tables

So far we have described how records in the `db.post` table are to be represented in RDF. Here we describe what a `db.post` is in OWL, by defining the `db.post` table’s “rdf” property which is implemented as a Python dictionary of dictionaries. After the previous code, we append:

```

1 db.post.rdf = {
2     'type': 'http://rdfs.org/sioc/ns#Post',
3     'namespaces': {
4         '_xmlns:dc': 'http://purl.org/dc/elements/1.1/',
5         '_xmlns:sioc': 'http://rdfs.org/sioc/ns#'
6     },
7     'references': {
8         'comment': 'sioc:has_reply'
9     }
10 }

```

The top level dictionary keys have the following meaning:

- The “type” key of the dictionary defines what a `db.post` is in the Semantic Web. In our case, the SIOC ontology has defined a “Post” class which all of our posts are an instance of.
- “namespaces” defines the OWL namespaces to be included in order to properly serialize a `db.post` in RDF/XML. Recall in the previous step, annotating fields, we defined a post’s author to be associated with the `sioc:has_creator` property, the title with `dc:title`, and body as `sioc:content`. This means we need to include namespaces for sioc and dc.

- The “references” key defines how matching records from foreign key tables are exposed. Here we are saying that when a `db.post` is serialized, the `sioc:has_reply` property will be used to reference all related comments.

In this paper we refer to code written in steps 2 and 3 as RDF annotations.

#### 4.4 More annotations

Similarly we can annotate other tables (tag and comment):

```

1 db.comment.author.rdf = 'sioc:has_creator'
2 db.comment.body.rdf = 'sioc:content'
3
4 db.comment.rdf = {
5     'type': 'http://rdfs.org/sioc/ns#Post',
6     'namespaces': { '_xmlns:sioc': "http://rdfs.org/sioc/ns#" }
7 }
8
9 db.tag.name.rdf = 'sioc:content'
10
11 db.tag.rdf = {
12     'type': 'http://rdfs.org/sioc/ns#topic',
13     'namespaces': { '_xmlns:sioc': "http://rdfs.org/sioc/ns#" }
14 }
```

#### 4.5 Exposing the Service

After the database tables are annotated, the web services are exposed automatically by web2py.

Visit the following URL:

```
1 http://127.0.0.1:8000/semantic/plugin_rdf/ld/tables
```

to get a semantic description of our database:

```

1 <rdf:RDF>
2   <rdf:Description rdf:about="/semantic/plugin_rdf/ld/tables">
3     <rdf:type rdf:resource="http://www.dbs.cs.uni-duesseldorf.de/RDF/
4       relational.owl#Database"/>
5     <relational:hasTable
6       rdf:resource="/semantic/plugin_rdf/ld/select/post"/>
7     <relational:hasTable
8       rdf:resource="/semantic/plugin_rdf/ld/select/comment"/>
9     <relational:hasTable
10      rdf:resource="/semantic/plugin_rdf/ld/select/tag"/>
11   </rdf:Description>
12 </rdf:RDF>
```

Specifically, the url exposes a Linked Data resource listing database table resources [25]. WEB2PY looks for all tables with an `rdf` property as a filter for this resource.

Looking at the post table:

```
1 http://127.0.0.1:8000/semantic/plugin_rdf/ld/select/post
```

we receive information about the table and its records:

```

1 <rdf:RDF>
2   <rdf:Description rdf:about="/semantic/plugin_rdf/ld/select/post">
3     <rdf:type rdf:resource="http://www.dbs.cs.uni-duesseldorf.de/RDF/
4       relational.owl#Table"/>
5     <relational:has
6       rdf_resource="/semantic/plugin_rdf/ld/read/post/1"/>
7     <relational:has
8       rdf_resource="/semantic/plugin_rdf/ld/read/post/2"/>
9   </rdf:Description>
9 </rdf:RDF>

```

Our post is an instance of the relational OWL class Table and it contains two rows.

Continuing on to one of the records:

```
1 http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/post/1
```

```

1 <rdf:RDF>
2   <rdf:Description rdf:about="/semantic/plugin_rdf/ld/select/post/1">
3     <rdf_type rdf:resource="http://rdfs.org/sioc/ns#Post"/>
4     <sioc:has_creator>Chris Baron</sioc:has_creator>
5     <dc:title>How To Publish RDF Using web2py</dc:title>
6     <sioc:content>
7       How To Publish RDF Using web2py, really
8     </sioc:content>
9     <sioc:has_reply
10    rdf_resource="/semantic/plugin_rdf/ld/read/comment/1"/>
11    <sioc:topic
12    rdf:resource="/semantic/plugin_rdf/ld/read/tag/1">
13    <sioc:tagRank>10</sioc:tagRank>
14  </sioc:topic>
15 </rdf:Description>
16 </rdf:RDF>

```

This is a db.post defined semantically. Notice the `rdf:type` element, it indicates that the post is an instance of the type defined in Ref.[7], which we set via the `type` attribute of the `db.post.rdf`. Each of the columns with an `rdf` attribute are transformed into their associated ontology where the record values are literals inside these XML elements. The `sioc:has_reply` is not referenced as a column's `rdf` attribute, but was retrieved from the “references” property of the `db.post.rdf` data structure.

web2py also recognizes many-to-many relations implemented via link tables (as in post tag link). Link tables are special because their records do not just connect two or more tables. They may also contain additional information associated to the link (like for example the ranking in the case of post tag link). We have the option to propagate this information and include in the requested entries connected by the link. Here is an example:

```

1 db.post_tag_link.rdf_mapping =
2   { 'post': { 'reference': 'sioc:topic',
3             'columns': { 'ranking': 'sioc:tagRank' } } }

```

Note that if we remove the “columns” attribute from the above code, the post serialization is the same except for the `sioc:tagRank` element disappears.

Also, if the `post` attribute is removed, the `sioc:topic` object property is replaced by the default property `relational:references`.

## 4.6 Complex annotations

While the previous example covers many cases of practical interests, in some cases it may not be sufficient and further customization may be necessary. Here is a more complex example of how one may annotate the `db.post.author` field:

```

1 db.post.author.rdf = {
2     'name': 'sioc:has_creator',
3     'children': [ {
4         'name': 'sioc:User',
5         '_rdf:about': 'http://chrisbaron.com',
6         '_rdf:label': '$VALUE',
7         'children': [ { 'name': 'rdf:seeAlso',
8                         '_rdf:resource': 'http://web2py.com'
9                     }
10                ]
11            }
12        }

```

Above, `$VALUE` is replaced with the database record's value, the “`children`” attribute denotes child nodes; “`name`” corresponds to the element name, and any attribute beginning with an underscore is an element attribute.

A request to

```
1 http://127.0.0.1:8000/semantic/plugin_rdf/ld/read/post/1
```

generates the following complex `sioc:hascreator` response:

```

1 <sioc:has_creator>
2   <sioc:User rdf:about="http://chrisbaron.com" rdf:label="Chris Baron">
3     <rdf:seeAlso rdf:resource="http://web2py.com"/>
4   </sioc:User>
5 </sioc:has_creator>

```

## 5 Related Work

The idea of mapping Database Relations into RDF is not at all new. As already observed by [16], the Relational Model, which predates SQL, expresses the syntactic structure of the stored data and therefore it can be mapped into RDF. A number of authors have realized such mapping [9, 10, 11, 12, 13].

What is lacking in this approach is domain specific semantics, as expressed by OWL. In our system the domain specific semantic is injected by annotating the database tables, fields and relations in the DAL. This provides two advantages over other approaches: it is database agnostic and it is not domain specific.

Table 1 shows a comparative classification according to the W3C RDB2RDF Incubator Group as reported in ref. [24].

Approach	Auto(a)	Database(b)	Paradigm(c)	Map(d)	Domain(e)
Dartgrid [14]	Manual	Domain	SPARQL	Visual	specific
Hu et al. [15]	Auto	Both	ETL	intern	specific
Tirmizi et al. [16]	Auto	DB	ETL	FOL	general
Li et al. [17]	Semi	DB	ETL	n/a	general
DB2OWL [18]	Semi	DB	SPARQL	R2O	general
RDBToOnto [19]	Semi	DB+M	ETL	Visual	general
Sahoo et al. [20]	Manual	Domain	ETL	XSLT	specific
R2O [21]	Manual	DB+M	SPARQL	R2O	specific
D2RQ [22]	Auto	DB+M	LD,SPARQL	D2RQ	general
Virtuoso [23]	Semi	DB+M	SPARQL	own	general
Triplify [24]	Manual	Domain	LD	SQL	general
(this paper)	Semi	DB+M	LD	DAL	general

Table 1: Reference table of common mapping approaches. The classification criteria are: (a) degree of mapping creation automation: automatic, semi automatic, or manual; (b) database driven (DB), domain specific, or database driven with additional domain specific information (DB+M); (c) the resulting paradigm (LD for Linked Data); (d) language used for the mapping. (DAL refers to the web2py specific Database Abstraction Layer); (e) domain specific or general.

## 6 Conclusions

The web is a very young place, very much evolving, and it can offer us much more than we have experienced already. The next phase in its evolution is probably the semantic web. Today, web crawling is mostly achieved by parsing unstructured information from web sites (like Google does) and it has a high margin of error. Tomorrow, Linked Data users will be able to run programs (intelligent agents) that mine data in an intelligent way and generate the information by reasoning. Linked-data provide an extensible way to publish structured information.

In this paper we have shown a tool that allows someone to create a database (or connect to an existing database) using a Database Abstraction Layer and expose the data as RDF via a web service. While this idea is not new, our system allows to annotate arbitrary databases using ontological information in a portable way that is independent of the database back-end and can leverage on the capability of the WEB2PY framework for web application development.

## Acknowledgements

## References

- [1] Berners-Lee, T.: Design Issues: Linked Data (2006), <http://www.w3.org/DesignIssues/LinkedData.html>

- [2] Bizer, C., Cyganiak, R., Heath, T.: How to publish Linked Data on the Web (2007), <http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/>
- [3] <http://www.web2py.com/>
- [4] <http://www.w3.org/2001/sw/sweo>
- [5] <http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/>
- [6] <http://dublincore.org/documents/dcmi-terms>
- [7] <http://rdfs.org/sioc/ns\#Post>
- [8] Brickley, D., Miller, L.: FOAF vocabulary specification 0.91. Technical report, ILRT (2007)
- [9] de Laborda, C. P. and Conrad, S. 2005. Relational.OWL: a data and schema representation format based on OWL. In Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling
- [10] Svihla M., Jelnek I. 2004. Two Layer Mapping from Database to RDF. In Proceedings of Electronic Computers and Informatics (ECI), Slovakia, Kosice
- [11] Laclavik, M. 2006. RDB2Onto: Relational Database Data to Ontology Individual Mapping In: Tools for Acquisition, Organisation and Presenting of Information and Knowledge. P.Navrat et al. (Eds.),
- [12] Bizer, C. Seaborne, A. 2004. D2RQ Treating Non-RDF Databases as Virtual RDF Graphs. Poster at 3rd International Semantic Web Conference (ISWC2004)
- [13] Bizer, C. 2003. D2R MAP - A Database to RDF Mapping Language. The Twelfth International World Wide Web Conference (WWW2003)
- [14] Zhaohui Wu, Shuming Tang, Shuguang Deng, Jian Wu, Huajun Chen, Haijun Gao, "DartGrid II: A Semantic Grid Platform for ITS," IEEE Intelligent Systems, vol. 20, no. 3, pp. 12-15, May/June 2005, doi:10.1109/MIS.2005.44
- [15] W. Hu and Y. Qu. Discovering simple mappings between relational database schemas and ontologies. In Proceedings of ISWC/ASWC2007, Busan, South Korea, volume 4825 of LNCS, pages 225238, 2007
- [16] Sequeda, J F. Tirmizi, S.H. Corcho, O. Miranker, D.P. (2009) Direct Mapping SQL Databases to the Semantic Web. Technical Report 09-04. The University of Texas at Austin, Department of Computer Sciences.

- [17] M. Li, X. Du, and S. Wang. A semi-automatic ontology acquisition method for the semantic web. In W. Fan, Z. Wu, and J. Yang, editors, WAIM, volume 3739 of LNCS, pages 209220. Springer, 2005.
- [18] Cullot, N., Ghawi, R., Ytongnon, K.: DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. Universit de Bourgogne (2007)
- [19] F. Cerbah. Learning highly structured semantic repositories from relational databases: The RDBToOnto tool. In Proc. of ESWC 2008, Tenerife, 2008.
- [20] S. S. Sahoo, O. Bodenreider, J. L. Rutter, K. J. Skinner, and A. P. Sheth. An ontology-driven semantic mashup of gene and biological pathway information: Application to the domain of nicotine dependence. Journal of biomedical informatics, February 2008.
- [21] Barrasa, J., Corcho, O., Gmez-Prez, A.: R2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language. In: Bussler, C.J., Tannen, V., Fundulaki, I. (eds.) SWDB 2004. LNCS, vol. 3372. Springer, Heidelberg (2005)
- [22] C. Bizer and A. Seaborne, D2RQtreating non-RDF databases as virtual RDF graphs.. In: S.A. McIlraith, D. Plexousakis and F. van Harmelen, Editors, Proceedings of 3rd International Semantic Web Conference (ISWC04) Hiroshima, Japan. Springer, November (2004).
- [23] Orri Erling and Ivan Mikhailov. RDF Support in the Virtuoso DBMS. In Sören Auer, Christian Bizer, Claudia Müller, and Anna V. Zhdanova, editors, CSSW, volume 113 of LNI, pages 5968. GI, 2007.
- [24] Sören Auer *et al.*, Triplify Light-Weight Linked Data Publication from Relational Databases
- [25] <http://www.schemaweb.info/schema/SchemaDetails.aspx?id=252>

# QCDUTILS

Massimo Di Pierro

February 21, 2012

## Abstract

This manual describes a set of utilities developed for Lattice QCD computations. They are collectively called `qcdutils`. They are comprised of a set of Python programs each of them with a specific function: download gauge ensembles from the public NERSC repository, convert between formats, split files by time-slices, compile and run physics algorithms, generate visualizations in the form of VTK files, convert the visualizations into images, perform bootstrap analysis of results, fit the results of the analysis, and plot those results. These tools implement the typical workflow of most Lattice QCD computations and automate it by enforcing filename conventions: the output of one tool is understood by the next tool in the workflow. This manual is organized as a series of autonomous recipes which can be combined together.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Resources . . . . .	5
1.2	Getting the tools . . . . .	6
1.3	Dependencies . . . . .	7
1.4	License . . . . .	7
1.5	Acknowledgments . . . . .	7
<b>2</b>	<b>Accessing public data with <code>qcdutils_get.py</code></b>	<b>7</b>
2.1	Searching data on the NERSC <i>Gauge Connection</i> . . . . .	8
2.2	Downloading data from NERSC . . . . .	10
2.3	Testing download . . . . .	12
2.4	Converting to ILDG format (.ildg) . . . . .	12
2.5	Using the catalog file . . . . .	13
2.6	Converting gauge configurations to the FermiQCD format (.mdp) . . . . .	14
2.7	Splitting gauge configurations into time-slices . . . . .	14
2.8	Splitting ILDG propagators into timeslices . . . . .	15
<b>3</b>	<b>Details about file formats</b>	<b>15</b>
3.1	NERSC file format (3x3) . . . . .	16
3.2	NERSC file format (3x2) . . . . .	17
3.3	MILC file format . . . . .	19
3.4	FermiQCD file format . . . . .	20
3.5	LIME file format . . . . .	22
3.6	ILDG file format . . . . .	24
3.7	SciDAC file format . . . . .	25
<b>4</b>	<b>Running physics algorithms with <code>qcdutils_run.py</code></b>	<b>26</b>
4.1	Running in parallel . . . . .	27
4.2	General syntax . . . . .	28
4.3	Creating a cold or hot gauge configuration . . . . .	29
4.4	Loading a gauge configuration . . . . .	30
4.5	Heatbath Monte Carlo . . . . .	31
4.6	Computing a pion propagator . . . . .	32
4.7	Action and inverters . . . . .	34
4.8	Meson propagators . . . . .	36
4.9	Current insertion . . . . .	38

4.10	Four quark operators . . . . .	39
<b>5</b>	<b>Images and movies with <code>qcdutils_vis.py</code> and <code>qcdutils_vtk.py</code></b>	<b>41</b>
5.1	About VTK file format . . . . .	42
5.2	Plaquette . . . . .	43
5.3	Topological charge density . . . . .	46
5.4	Cooling . . . . .	48
5.5	Polyakov lines . . . . .	50
5.6	Quark propagator . . . . .	52
5.7	Pion propagator . . . . .	52
5.8	Meson propagators . . . . .	54
5.9	Current insertions . . . . .	55
5.10	Localized instantons . . . . .	55
<b>6</b>	<b>Analysis with <code>qcdutils_boot.py</code>, <code>qcdutils_plot.py</code>, <code>qcdutils_fit.py</code></b>	<b>57</b>
6.1	A simple example . . . . .	61
6.2	2-point and 3-point correlation functions . . . . .	63
6.3	Fitting data with <code>qcdutils_fit.py</code> . . . . .	66
6.4	Dimensional analysis and error propagation . . . . .	70
<b>A</b>	<b>Filename conventions</b>	<b>71</b>
<b>B</b>	<b>Help Pages</b>	<b>72</b>
B.1	<code>qcdutils_get.py</code> . . . . .	72
B.2	<code>qcdutils_run.py</code> . . . . .	72
B.3	<code>qcdutils_vis.py</code> . . . . .	75
B.4	<code>qcdutils_vtk.py</code> . . . . .	77
B.5	<code>qcdutils_boot.py</code> . . . . .	77
B.6	<code>qcdutils_plot.py</code> . . . . .	78
B.7	<code>qcdutils_fit.py</code> . . . . .	78

# 1 Introduction

In this manual we provide a description of the following tools:

- `qcdutils_get.py`: a program to download gauge configurations from the NERSC *Gauge Connection* archive [1] and convert them from one format to another, including to ILDG [2] and FermiQCD formats [3, 4].
- `qcdutils_run.py`: a program to download, compile and run various parallel Physics algorithms (for example compute the average plaquette, the topological charge density, two and three points correlation functions). `qcdutils_run` is a proxy for FermiQCD. Most of the FermiQCD algorithms and examples generate files that are suitable for visualization (VTK files [5])
- `qcdutils_vis.py`: a program to manipulate the VTK files generated by `qcdutils_run` which can be used to split VTK files into components, interpolate them, and generate 3D contour plots as JPEG images. This program uses metaprogramming to write a VisIt [6] script and runs it in background.
- `qcdutils_vtk.py`: a program that converts a VTK file into a web page (HTML) which displays iso-surfaces computed from the VTK file. The generated files can be visualized in any browser and allows interactive rotation of the visualization. This program is based on the “processing.js” library [7].
- `qcdutils_boot.py`: a tool for performing bootstrap analysis of the output of `qcdutils_run` and other QCD Software. It computes autocorrelations, moving averages, and distributions.
- `qcdutils_plot.py`: a tool to plot results from `qcdutils_boot`.
- `qcdutils_fit.py`: a tool to fit results from `qcdutils_boot.py`.

As the .py extension implied, these programs are written in Python [8] (2.7 version recommended).

Together these tools allow automation of the workflow of most Lattice QCD computations from downloading data to computing scientific results, plots, and visualizations.

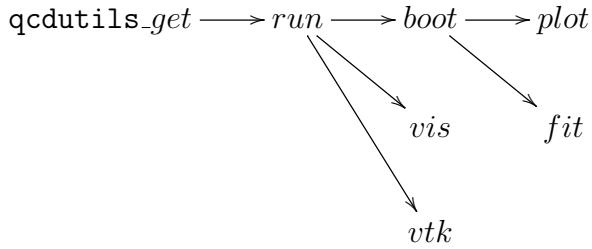
Notice that each of the utilities has its own help page which you can access using the -h command line option. The output for each is reported in the Appendix.

The data downloaded by `qcdutils_get` can be read by `qcdutils_run` which executes the physics algorithms implemented in C++. The output can be VTK files manipulated by

`qcdutils_vis` and then transformed into images and movies by VisIt, or they can be tabulated data that require bootstrap analysis. This is done by `qcdutils_boot`. The output of the latter plotted by `qcdutils_plot` and can be fitted with `qcdutils_fit`.

These files enforce a workflow by following the file naming conventions described in the Appendix but, they do not strictly depend on each other. For example `qcdutils_boot` can be used to analyze the output of any of your own physics simulations even if you do not use `qcdutils_run`.

Here is an overview of the workflow:



This manual is not designed to be complete or exhaustive because our tools are in continuous development and new features are added every day. Yet it is designed to provide enough examples to allow you to explore further. Our analysis and visualizations are created on sample data and aimed exclusively at explaining how to use the tools.

Our hope is that these tools will be useful to practitioners in the field and specifically to graduate students new to the field of Lattice QCD and looking to jumpstart their research projects.

These tools can also be used to automate the workflow of analyzing gauge configurations in real time in order to obtain and display preliminary results.

Some of the tools described here find more general application than Lattice QCD and can be utilized in other scientific areas.

## 1.1 Resources

`qcdutils` can be downloaded from:

<http://code.google.com/p/qcdutils>

More information FermiQCD code used by `qcdutils_run` is available from refs. [3, 4] and the web page:

<http://fermiqcd.net>

More examples of visualizations and links do additional code and examples can be found at:

<http://latticeqcd.org>

## 1.2 Getting the tools

There are two ways to get the tools described in here. The easiest way to get `qcdutils` is to use Mercurial:

Install mercurial from:

```
1 http://mercurial.selenic.com/
```

and download `qcdutils` from the googlecode repository

```
1 http://code.google.com/p/qcdutils/source/browse/
```

using the following commands:

```
1 hg clone https://qcdutils.googlecode.com/hg/ qcdutils
2 cd qcdutils
```

The command creates a folder called “`qcdutils`” and download the latest source files in there.

You can also download individual files using `wget` (default on Linux systems) or `curl` (default on mac systems):

```
1 wget http://qcdutils.googlecode.com/hg/qcdutils_get.py
2 wget http://qcdutils.googlecode.com/hg/qcdutils_run.py
3 wget http://qcdutils.googlecode.com/hg/qcdutils_vis.py
4 wget http://qcdutils.googlecode.com/hg/qcdutils_vtk.py
5 wget http://qcdutils.googlecode.com/hg/qcdutils_boot.py
6 wget http://qcdutils.googlecode.com/hg/qcdutils_plot.py
7 wget http://qcdutils.googlecode.com/hg/qcdutils_fit.py
```

## 1.3 Dependencies

These files do not depend on each other so you can download only those that you need. `qcdutils_run` is special because it is a Python interface to the `FermiQCD` library. As it is explained later, when executed, it downloads and compiles `FermiQCD`. It assumes you have `g++` installed.

`qcdutils_fit.py` and `qcdutils_plot.py` requires the Python `numpy` and `matplotlib` installed.

All the file require Python 2.x (possibly 2.7) and do not work with Python 3.x.

## 1.4 License

`qcdutils` are released under the GPLv2 license.

## 1.5 Acknowledgments

We thank all members of the USQCD collaborations for making most of their data and code available to the public, and for a long-lasting collaboration. We thank David Skinner, Schreyas Cholia, and Jim Hetrick for their collaboration in improving and running the NERSC gauge connection. We particularly thank Jim Hetrick for sharing his code for t'Hooft instantons. We thank Chris Maynard for useful discussions about ILDG. We thanks Simon Catterall, Yannick Meurice, Jonathan Flynn, and all those that over time have submitted patches for FermiQCD thus contributing to make it better. We also thank all of those who have used and who still use FermiQCD, thus providing the motivation for continuing this work. We thank the graduate students that over time have helped with coding, testing, and documentation: Yaoqian Zhong, Brian Schinazi, Nate Wilson, Vincent Harvey, and Chris Baron.

This work was funded by Department of Energy grant DEFC02-06ER41441 and by National Science Foundation grant 0970137.

## 2 Accessing public data with `qcdutils_get.py`

## 2.1 Searching data on the NERSC *Gauge Connection*

The *Gauge Connection* [1] is a repository of Lattice QCD data, primarily but not limited to gauge ensembles, hosted by the National Energy Research Science Center (NERSC) on their High Performance Storage System (HPSS). At the time of writing the Gauge Connection hosts 16 Terabytes of data and makes it publicly accessible to researchers worldwide.

The new Gauge Connection site consists of a set of dynamic web pages in hierarchical structures that closely mimics the folder structure in the HPSS FTP server. Each folder corresponds to a web page. The web page provide a description of the folder content, in the form of an editable wiki, comments about the content, links to sub-folder and links to files contained in the folder. Since folders may contain thousands of files, files with similar filenames are grouped together into filename patterns. For example all files with the same name but different extension or similar names differing only for a numerical value are grouped together. Pages are tagged and can be searched by tag. Users can search for files by browsing the folder structure, searching by tags and can download individual files or all files matching a pattern.

You do not need an account to login and you can use your OpenID account, for example a Google email account. You do not need to login to search but you need to login to download. From now on we assume you are logged in into the Gauge Connection.

Fig 1 (left) is a screenshot of the main Gauge Connection site. Each gauge ensemble is stored in a folder which is represented by a dynamic web page and tagged. You can search these pages by tag, as shown in Fig. 1 (right).

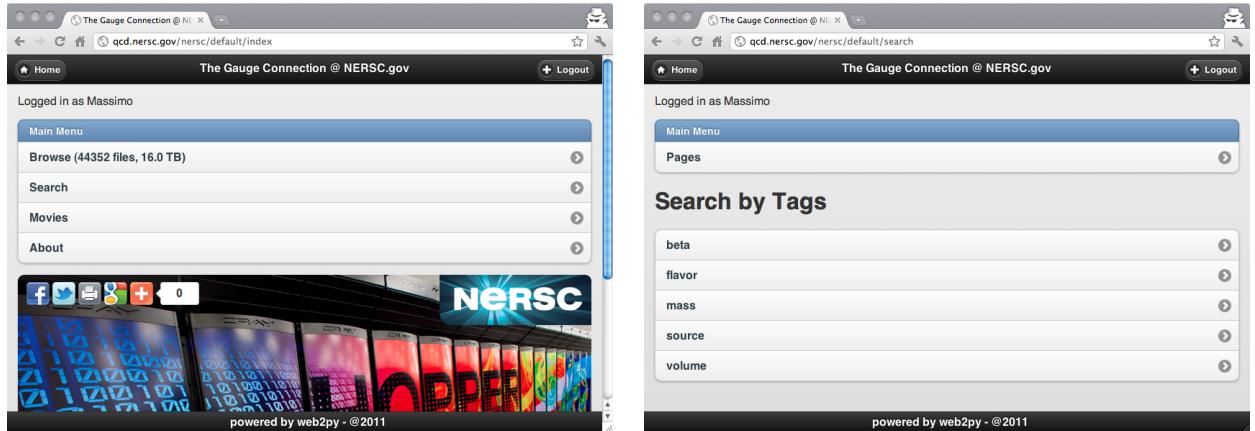


Figure 1: Main NERSC Gauge Connection web site (left) and search by tag feature (right).

Fig. 2 (left) shows statistical information about tags.

Each tag has the form “type/value” where the tag type can be:

- *source*: the name of the organization who donated the data, value can be, for example MILC [9].
- *flavor*, the flavor content of the data, value can be “0” for quenched data, “2” for two flavor unquenched data, “2+1” for three flavor unquenched where two quarks have one mass and 1 quark has another mass.
- *kappa*: the  $\kappa$  value
- *mass*: the quark mass

We have also processed many of the ensambles using some of the tools described here and generated animations of the topological charge densities. This is shown in fig. 2 (right).

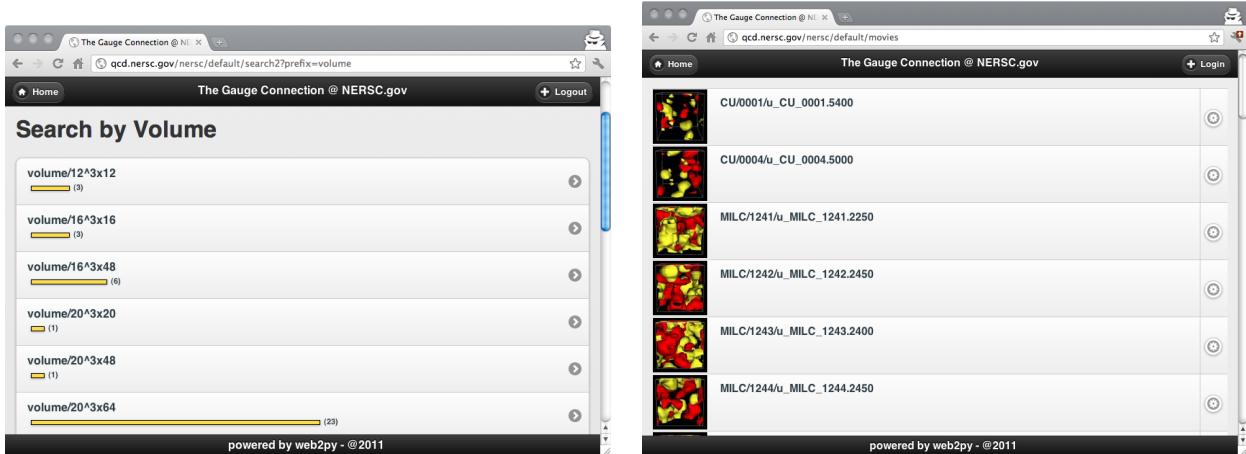


Figure 2: A page showing statitistical information (left) and list of visualzations (right).

A screenshot of a folder page is shown in fig. 3 (left)

You can see a description, a list of tags, list of file patterns in the folder, and comments. The comments are only visible to logged in users. The login link is at top left of the page.

A screenshot of a page listing the files in an ensemble is shown in fig. 3 (right)

You can download an individual files by clicking on the file.

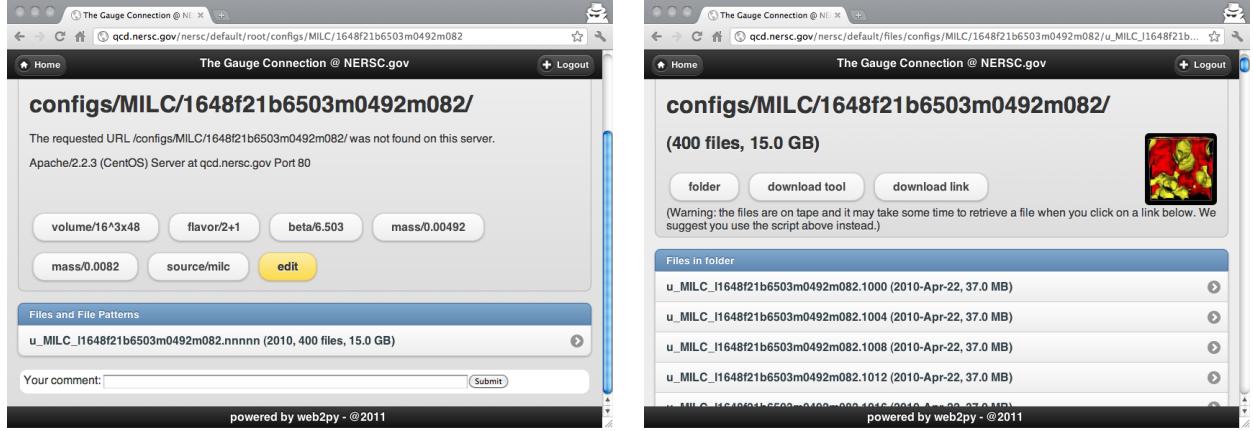


Figure 3: Folder page (left) and page listing all files in an ensamble (right).

To download files in batch you need to first download `qcdutils_get.py`. The web page above includes a link [*download tool*] that explains where to get and how to use `qcdutils_get.py`. We suggest you first read the rest of this section but also read the linked instructions which may be more updated. The page also contains a link [*download link*] which is used to reference the data for later download.

## 2.2 Downloading data from NERSC

Here we assume you want to download the 400 MILC gauge configurations of size  $16^3 \times 48$  and  $\beta = 6.503$  computed using 2+1 quarks of mass respectively 0.00492 (light) and 0.082 (heavy). These files can be found at

```
1 http://qcd.nerc.gov/nersc/default/root/configs/MILC/1648f21b6503m0492m082
```

where you should notice the folder name

```
1 1648f21b6503m0492m082
```

It follows the MILC filename convention

```
1 [time] [space] b [beta] m [mass] m [mass]
```

and the values of [beta] and [mass] omit the decimal point.

The above page links the pattern page:

```
1 http://qcd.nersc.gov/nersc/default/files/configs/MILC/1648  
f21b6503m0492m082/u_MILC_11648f21b6503m0492m082.nnnnn
```

where

```
1 u_MILC_11648f21b6503m0492m082.nnnnn
```

is the filename pattern and `nnnnn` is just a wildcard for the gauge configuration numbers in the ensemble.

The page contains a [*download link*] to a document in JSON format listing all files in the folder matching the pattern and additional meta-data about each file. You do not need to open this document. All you need to do is copy the link address and pass it to `qcdutils_get.py` as a command line argument. The program opens the URL, download the list, loop over the files in the ensemble, and download them one by one.

Copy the *download link* to clipboard and it looks something like this:

```
1 http://qcd.nersc.gov/nersc/api/files.json/.../  
u_MILC_11648f21b6503m0492m082.nnnnn
```

We have shortened the full actual path using .... This URL is a personal token and different users get different URLs for the same data. This allows the server to monitor usage and expire an URL in case of indiscriminate downloads from one user without affecting other users.

To download all data referenced by this link you simply paste the download link after a call to `qcdutils_get`:

```
1 python qcdutils_get.py [download link]
```

`qcdutils_get.py` performs the following operations:

Before downlaod, `qcdutils` creates a folder with the same name as the ensemble:

```
1 u_MILC_11648f21b6503m0492m082.nnnnn/
```

and then download all files in there. The files retain the original file name.

`qcdutils` also creates a file called “`qcdutils.catalog`” where it keeps track of successful downloads. This allows automatic resume on restart: if your download is interrupted, for any reason (for example network problem or server crash), you can re-issue the download command and it resumes where it stopped. `qcdutils` does not download again files that were already downloaded and are currently present on your system.

`qcdutils` can check if a file is complete by checking its size. Data integrity during transmission is guaranteed by the TCP protocol. It is still possible that data is corrupted at the source or locally after download (for example due to a bad disk sector). If a file is found to be corrupted simply delete it, run `qcdutils_get` again, and it downloads it again.

Notice that most of the files stored and served by the Gauge Connection are in either the NERSC 3x3 or the NERSC 3x2 file format, described later. If your program can read them, you do not need any conversion. Yet it is likely you need to convert them and this is the subject of the rest of the section.

## 2.3 Testing download

If you encounter any problem downloading real data you can try download a single small demo gauge configuration:

```
1 python qcdutils_get.py http://qcd.nersc.org/nersc/api/files/demo
```

It creates a folder called `demo` and download a single file

```
1 demo/demo.nersc
```

## 2.4 Converting to ILDG format (.ildg)

The `qcdutils_get.py` can also auto-detect and convert file formats. It can input NERSC3x2, NERSC3x3, MILC, UKQCD, ILDG, SciDAC, FermiQCD and it can output ILDG and FermiQCD formats. Other output formats may be supported in the future if this becomes necessary. `qcdutils` converts files using the following syntax

```
1 python qcdutils_get.py -c [target-format] [source]
```

Here [source] can be a *download link*, a *glob* pattern such as “`demo/*`”, or an individual file. [target-format] is one of the following:

- *ildg* converts a gauge configuration to ILDG
- *mdp* converts a gauge configuration to the FermiQCD format
- *slice.mdp* converts a gauge configuration (for example  $12^3 \times 48$ ) into multiple configuration files, one for each time slice (for example  $1 \times 12^3$ ), in the FermiQCD file format.

- *prop* like *mdp* but converts file propagators from *Scidac-ILDG format* into the FermiQCD format.
- *slice.prop.mdp* like *prop.mdp* but converts a *Scidac-ILDG* propagator into FermiQCD time-slice files.

Most gauge configuration files are very large and require physics algorithms to run in parallel. Yet some algorithms, specifically some visualization ones, can work on individual time-slices. *slice.mdp* and *slice.prop.mdp* allow you to break large files into time-slices for this purpose.

Here is an example to convert to ILDG format:

```
1 python qcdutils_get.py http://qcd.nersc.gov/nersc/api/files/demo
2 python qcdutils_get.py -c ildg demo/*
```

Or in one single line:

```
1 python qcdutils_get.py -c ildg http://qcd.nersc.gov/nersc/api/files/demo
```

If the source file is “demo/demo.nersc”, the converted file has the “.ildg” postfix appended and be called “demo/demo.nersc.ildg”. The original file is not deleted. These are the output folder/files:

```
1 demo/
2 demo/demo.nerc
3 demo/demo.nersc.ildg
4 demo/qcdutils.catalog
```

By default `qcdutils` preserves the precision of the input data, but can specify the precision of the target gauge configuration using the `-4` flag for single precision and the `-8` flag for double precision. The input precision is automatically detected. For example:

```
1 python qcdutils_get.py -c ildg -4 demo/*
```

## 2.5 Using the catalog file

The file “`qcdutils.catalog`” is only used internally by `qcdutils` and should not be deleted or else it loses track of completed downloads and may perform them again unnecessarily.

If can pass a `qcdutils.catalog` to `qcdutils_get` you get a report about the downloaded files.

```
1 python qcdutils_get.py demo/qcdutils.catalog
```

Notice that output files are never overwritten so make sure you delete the old one if you want to create new ones.

## 2.6 Converting gauge configurations to the FermiQCD format (.mdp)

This option works similarly to the previous section:

```
1 python qcdutils_get.py http://qcd.nersc.gov/nersc/api/files/demo
2 python qcdutils_get.py -c mdp demo/*
```

or in one line

```
1 python qcdutils_get.py -c mdp http://qcd.nersc.gov/nersc/api/files/demo
```

It creates

```
1 demo
2 demo/demo.nersc
3 demo/demo.nersc.mdp
4 demo/qcdutils.catalog.db
```

As in the previous case you can specify the precision of the converted file using `-4` or `-8`.

Notice you cannot specify the endianness. The FermiQCD format (.mdp) uses LITTLE endianness by convention because that is the format used internally by x386 compatible architectures.

## 2.7 Splitting gauge configurations into time-slices

Often we need to break a single gauge configuration with  $T$  time-slices into  $T$  gauge configurations with 1 time-slice each. You can do it using the `slice.mdp` output file format:

```
1 python qcdutils_get.py http://qcd.nersc.gov/nersc/api/files/demo
2 python qcdutils_get.py -c slice.mdp demo/demo.mdp
```

The first line creates the following files:

```
1 demo/
2 demo/demo.nersc
3 demo/qcdutils.catalog.db
```

while the second line creates:

```
1 demo/demo.nersc.t0001.mdp
2 demo/demo.nersc.t0002.mdp
3 demo/demo.nersc.t0003.mdp
4 demo/demo.nersc.t0004.mdp
```

Four files because “demo.nersc” contains 4 timeslices.

## 2.8 Splitting ILDG propagators into timeslices

We can play the same trick with propagators. While for gauge configurations `qcdutils` can read multiple file formats, for input propagators qcd can only read FermiQCD and SciDAC propagators.

Given a file “propagator.scidac” we can convert it into FermiQCD format:

```
1 python qcdutils_get.py -c prop.mdp propagator.scidac
```

which creates

```
1 propagator.scidac.prop.mdp
```

or split into time-slices

```
1 python qcdutils_get.py -c slices.prop.mdp propagator.scidac
```

This creates

```
1 propagator.scidac.t0000.prop.mdp
2 propagator.scidac.t0001.prop.mdp
3 propagator.scidac.t0002.prop.mdp
4 ...
```

In this case you can specify the target precision.

## 3 Details about file formats

In this section we show simplified code snippets that should help you understand the different file formats used in Lattice QCD. They are very similar to the actual code implemented in `qcdutils` but simplified for readability.

### 3.1 NERSC file format (3x3)

To better illustrate each data format we present a minimalist program to store data in the corresponding format.

We assume the input is available through an instance of the following class called `data`:

```
1 class GenericGauge(object):
2     def u(x,y,z,t,mu):
3         # u_ij below are complex numbers
4         return [[u_00,u_01,u_02],
5                 [u_10,u_11,u_12],
6                 [u_20,u_21,u_22]]
```

In this section (and only in this section) we follow the convention that  $\mu = 0$  is  $X$ , 1 is  $Y$ , 2 is  $Z$  and 3 is  $T$ . Everywhere else, in particular in the input parameters of `qcdutils.run`  $\mu = 0$  is  $T$ , 1 is  $X$ , 2 is  $Y$  and 3 is  $Z$ , which is the FermiQCD convention.

The following code shows how to read `data` and write it in the NERSC3x3 format:

```
1 NERSC_3x3_HEADER = """BEGIN_HEADER
2 HDR_VERSION = 1.0
3 DATATYPE = 4D_SU3_GAUGE_3x3
4 DIMENSION_1 = %(NX)i
5 DIMENSION_2 = %(NY)i
6 DIMENSION_3 = %(NZ)i
7 DIMENSION_4 = %(NT)i
8 CHECKSUM = %(checksum)s
9 LINK_TRACE = %(linktrace)f
10 PLAQUETTE = %(plaquette)f
11 CREATOR = %(creator)s
12 ARCHIVE_DATE = %(archive_date)s
13 ENSEMBLE_LABEL = %(label)s
14 FLOATING_POINT = %(precision)s
15 ENSEMBLE_ID = %(ensemble_id)s
16 SEQUENCE_NUMBER = %(sequence_number)i
17 BETA = %(beta)f
18 MASS = %(mass)f
19 END_HEADER
"""
21 def save_3x3_nersc(filename,metadata,data):
22     f = open(filename,'wb')
23     f.write(NERSC_3x3_HEADER % metadata)
24     nt = metadata['NT']
25     nx = metadata['NX']
```

```

26 ny = metadata['NY']
27 nz = metadata['NX']
28 if metadata['FLOATING_POINT']=='IEEE32':
29     couple = '>2f'
30 elif metadata['FLOATING_POINT']=='IEEE64':
31     couple = '>2d'
32 else:
33     raise RuntimeError, "Unknown precision"
34 for t in range(nt):
35     for z in range(nz):
36         for y in range(ny):
37             for x in range(nz):
38                 for mu in range(0,1,2,3):
39                     u = data.u(x,y,z,t,mu)
40                     for i in range(3):
41                         for j in range(3)
42                             c = u[i][j]
43                             re,im = real(c),imag(c)
44                             f.write(struct.pack(couple,re,im))

```

The variable `couple` determines how to pack in binary the 2 variables `re,im` using big endianness (“>”) in single (“f”) or double precision (“d”). For more info read the documentation on the Python “`struct`” package.

All common file formats used by the community to store QCD gauge configuration require two loops: one loop over the lattice sites and one loop over the link directions at each lattice site.

In the NERSC, ILDG and MILC case, the first loop is:

```

1 for t ...
2   for z ...
3     for y ...
4       for x ...

```

and the second loop is:

```

1 for mu in (X,Y,Z,T) # (0,1,2,3)

```

## 3.2 NERSC file format (3x2)

The NERSC 3x2 format is more common than NERSC 3x3 and here is how to write it:

```

1 NERSC_3x2_HEADER = """BEGIN_HEADER
2 HDR_VERSION = 1.0
3 DATATYPE = 4D_SU3_GAUGE
4 DIMENSION_1 = %(NX)i
5 DIMENSION_2 = %(NY)i
6 DIMENSION_3 = %(NZ)i
7 DIMENSION_4 = %(NT)i
8 CHECKSUM = %(checksum)s
9 LINK_TRACE = %(linktrace)f
10 PLAQUETTE = %(plaquette)f
11 CREATOR = %(creator)s
12 ARCHIVE_DATE = %(archive_date)s
13 ENSEMBLE_LABEL = %(label)s
14 FLOATING_POINT = %(precision)s
15 ENSEMBLE_ID = %(ensemble_id)s
16 SEQUENCE_NUMBER = %(sequence_number)i
17 BETA = %(beta)f
18 MASS = %(mass)f
19 END_HEADER
"""
20
21 def save_3x2_nersc(filename, metadata, data):
22     f = open(filename, 'wb')
23     f.write(NERSC_3x2_HEADER % metadata)
24     nt = metadata['NT']
25     nx = metadata['NX']
26     ny = metadata['NY']
27     nz = metadata['NZ']
28     if metadata['FLOATING_POINT'] == 'IEEE32':
29         couple = '>2f'
30     elif metadata['FLOATING_POINT'] == 'IEEE64':
31         couple = '>2d'
32     else:
33         raise RuntimeError, "Unknown precision"
34     for t in range(nt):
35         for z in range(nz):
36             for y in range(ny):
37                 for x in range(nx):
38                     for mu in range(0,1,2,3):
39                         u = data.u(x,y,z,t,mu)
40                         for i in range(3):
41                             for j in range(2) # here
42                             c = u[i][j]
43                             re,im = real(c),imag(c)
44                             f.write(struct.pack(couple,re,im))

```

Notice it differs from NERSC 3x3 by only two lines. One line is in the header:

```
1 DATATYPE = 4D_SU3_GAUGE
```

instead of

```
1 DATATYPE = 4D_SU3_GAUGE_3x3
```

and in the line marked by a `here`.

This file format does not store all the 3x3 matrices but only the first two rows. The third row can be reconstructed when reading the file using the condition that the third row is the (complex) vector product of the first two.

`qcdutils` can reads and rebuilds the missing rows using this code:

```
1 def reunitarize(s):
2     (aire, a1im, a2re, a2im, a3re, a3im,
3      b1re, b1im, b2re, b2im, b3re, b3im) = s
4     c1re = a2re*b3re - a2im*b3im - a3re*b2re + a3im*b2im
5     c1im = -(a2re*b3im + a2im*b3re - a3re*b2im - a3im*b2re)
6     c2re = a3re*b1re - a3im*b1im - a1re*b3re + a1im*b3im
7     c2im = -(a3re*b1im + a3im*b1re - a1re*b3im - a1im*b3re)
8     c3re = a1re*b2re - a1im*b2im - a2re*b1re + a2im*b1im
9     c3im = -(a1re*b2im + a1im*b2re - a2re*b1im - a2im*b1re)
10    return (aire, a1im, a2re, a2im, a3re, a3im,
11            b1re, b1im, b2re, b2im, b3re, b3im,
12            c1re, c1im, c2re, c2im, c3re, c3im)
```

### 3.3 MILC file format

The MILC file format is the same as the NERSC 3x3 but the header contains different information, it uses a binary format, and the endianness is not specified (although generally large endianness is used). The binary data after the header is the same as NERSC3x3.

Here is an example of code to write a MILC gauge configuration in Python:

```
1 def save_milc(filename,metadata,data):
2     f = open(filename,'wb')
3     milc_header = '>i4i64siii'
4     milc_magic = 20103
5     f.write(struct.pack(milc_header,
6                         milc_magic,
7                         metadata['NX'],
```

```

8         metadata[ 'NY' ] ,
9         metadata[ 'NZ' ] ,
10        metadata[ 'NT' ] ,
11        metadata[ 'ARCHIVE_DATE' ][ :64 ] ,
12        metadata[ 'ORDER' ] ,
13        metadata[ 'CHECKSUM1' ] ,
14        metadata[ 'CHECKSUM2' ] )
15 nt = metadata[ 'nt' ]
16 nx = metadata[ 'nx' ]
17 ny = metadata[ 'ny' ]
18 nz = metadata[ 'nz' ]
19 if metadata[ 'FLOATING_POINT' ] == 'IEEE32':
20     couple = '>2f'
21 elif metadata[ 'FLOATING_POINT' ] == 'IEEE64':
22     couple = '>2d'
23 else:
24     raise RuntimeError, "Unknown precision"
25 for t in range(nt):
26     for z in range(nz):
27         for y in range(ny):
28             for x in range(nx):
29                 for mu in range(0,1,2,3):
30                     u = data.u(x,y,z,t,mu)
31                     for i in range(3):
32                         for j in range(3)
33                             c = u[i][j]
34                             re,im = real(c),imag(c)
35                             f.write(struct.pack(couple,re,im))

```

When reading a MILC gauge configuration, `qcdutils.get` checks for the magic number and determines the endianness from the first 4bytes of the header. `qcdutils` also determines the precision from the total file size.

### 3.4 FermiQCD file format

The file format used by FermiQCD is called MDP and files have a ".mdp" extensions. They are very similar to the MILC format with these differences:

- the header has a different format and stores slightly different information
- the endianness is always little-endian.
- the inner loop over mu has the same order as the outer loop

- it is designed to work for an arbitrary number of dimensions (from 1D lattices to 10D lattices and have an arbitrary site structure) and the FermiQCD code deal with this aspect in an automated way that is explored later.

For regular QCD ( $SU(3)$  matrices per link and 4D lattice) a FermiQCD gauge configuration can be generated using the following code:

```

1 def save_fermiqcd_4d_su3(filename,metadata,data):
2     f = open(filename, 'wb')
3     nt = metadata['nt']
4     nx = metadata['nx']
5     ny = metadata['ny']
6     nz = metadata['nz']
7     header_format = '<60s60s60sLi10iii'
8     maginc_number = 1325884739
9     ndim,
10    if metadata['FLOATING_POINT']=='IEEE32':
11        couple = '>2f'
12        metadata['SITE_SIZE'] = 4*9*2*4
13    elif metadata['FLOATING_POINT']=='IEEE64':
14        couple = '>2d'
15        metadata['SITE_SIZE'] = 4*9*2*4
16    else:
17        raise RuntimeError, "Unknown precision"
18    f.write(struct.pack(header_format,
19                        'File Type: MDP FIELD',
20                        metadata['FILENAME'],
21                        metadata['ARCHIVE_DATE'][::60],
22                        magic_number,ndim,nt,nx,ny,nz,0,0,0,0,0,0,0,
23                        metadata['SITE_SIZE'],nt*nx*ny*nz))
24    for t in range(nt):
25        for z in range(nz):
26            for y in range(ny):
27                for x in range(nx):
28                    for mu in range(3,2,1,0):
29                        u = data.u(x,y,z,t,mu)
30                        for i in range(3):
31                            for j in range(3)
32                                c = u[i][j]
33                                re,im = real(c),imag(c)
34                                f.write(struct.pack(couple,re,im))

```

Notice the following:

- The header is binary but uses a string “File Type: MDP FIELD” to identify the file

format and version. This allows you to identify the file using an ordinary editor, like in the NERSC format.

- It still uses an integer magic number to allow the reader to check the endianness.
- It requires an `ndim` variable which is set to 4 because this manual mostly deals with 4D fields.
- It stores in `metadata['SITE_SIZE']` the number of bytes for each lattice site. For single precision this is 4 directions times 9 SU(3) matrix elements times 2 (real+complex) times 4 (4 bytes for IEEE32, single precision, float) = 288 bytes. It is 576 bytes for double precision.

## 3.5 LIME file format

The file formats described so far store metadata in a header which precedes the binary data.

The LIME data format is different. It is similar to TAR or MIME as scope. It is designed to package multiple files into one file.

A LIME file is divided into segments (sometime called records in the literature although it does not strictly conform to the definition of a record because LIME has nothing to do with databases). A segment is comprised of five parts: a magic number, a version number, an integer storing the size of the binary data, a segment name, and the binary data.

The magic number identifies the file as a LIME file and the version number identifies the LIME version. This information is repeated for each segment.

Notice that LIME records do not declare the type of the segments and this has to be inferred from the name of the segments. One important caveat of LIME is that some segments contain binary data, while some contain ASCII strings such as XML. Segments that contain ASCII strings are null-terminated and the terminating zero is counted in the size. Binary segments are not null-terminated. This is an important detail when reading the data.

`qcdutils` contains a class LIME that can be used to open LIME files and read/write segments in or out of order.

A minimalist implementation of the LIME file format is the following:

```
1 class Lime(object):
2     def __init__(self, filename, mode, version = 1):
3         self.magic = 1164413355
```

```

4     self.version = version
5     self.filename = filename
6     self.mode = mode
7     self.file = open(filename, mode)
8     self.records = [] # [(name, position, size)]
9     if mode == 'r' or mode == 'rb':
10        while True:
11            header = self.file.read(144)
12            if not header: break
13            magic, null, size, name = struct.unpack('!iiq128s',header)
14            if magic != 1164413355:
15                raise IOError, "not in LIME format"
16            name = name[:name.find('\0')]
17            position = self.file.tell()
18            self.records.append((name,position,size)) # in bytes
19            padding = (8 - (size % 8)) % 8
20            self.file.seek(size+padding,1)
21    def read(self,record):
22        (name,position,size) = self.records[record]
23        self.file.seek(position)
24        return (name, self.file, size)
25    def __iter__(self):
26        for record in range(len(self)):
27            yield self.read(record)
28    def write(self,name,data,size = None,chunk = MAXBYTES):
29        position = self.file.tell()
30        header = struct.pack('!iiq128s',self.magic,self.version,size,name)
31        self.file.write(header)
32        self.file.write(data)
33        self.file.write('\0'*(8 - (size % 8)) % 8)
34        self.records.append((name,size,position))
35    def close(self):
36        self.file.close()

```

The actual implementation in `qcdutils` is more complex because it performs more checks and because it can read and write segments even if they do not fit in RAM, which is not the case in the example above.

Here is an example of usage from Python:

Open a LIME file for writing

```

1 >>> from qcdutils_get import Lime
2 >>> lime = Lime('test.lime','w')

```

Write two records in it

```
1 >>> lime.write('record1', '01234567')
2 >>> lime.write('record2', 'other binary data')
```

Close it

```
1 >>> lime.close()
```

Open the file again for reading:

```
1 >>> lime = Lime('test.lime', 'r')
```

Loop over the segments and print, name size, content:

```
1 >>> for name,reader,size in lime:
2 ...     print (name, size, reader.read(size))
```

## 3.6 ILDG file format

The ILDG file format uses LIME to package two segments:

- One segment contains the metadata marked up in XML.
- One segment contains the binary data, in the same format as in MILC and NERSC 3x3.

The XML markup is specified by ILDG for 4D gauge files. Notice that because the first segment refers to the second, many programs that read ILDG expect the metadata segment to precede the data segment.

Here is an example of code to write an ILDG file:

```
1 def save_ildg(filename,metadata,data,lfn):
2     lime = Lime(filename, 'wb')
3     lime.write('ildg-format', """
4         <?xml version = "1.0" encoding = "UTF-8"?>
5         <ildgFormat>
6             <version>%(VERSION)s</version>
7             <field>su3gauge</field>
8             <precision>%(PRECISION)s</precision>
9             <lx>%({NX})s</lx>
10            <ly>%({NY})s</ly>
11            <lz>%({NZ})s</lz>
12            <lt>%({NT})s</lt>
13        </ildgFormat>
```

```

14 """ .strip() % metadata)
15     nt = metadata[ 'NT' ]
16     nx = metadata[ 'NX' ]
17     ny = metadata[ 'NY' ]
18     nz = metadata[ 'NZ' ]
19     def writer():
20         for t in range(nt):
21             for z in range(nz):
22                 for y in range(ny):
23                     for x in range(nz):
24                         for mu in range(0,1,2,3):
25                             u = data.u(x,y,z,t,mu)
26                             for i in range(3):
27                                 for j in range(3):
28                                     c = u[i][j]
29                                     re,im = real(c),imag(c)
30                                     yield struct.pack(couple,re,im)
31     self.lime.write('ildg-binary-data',writer)
32     self.lime.write('ildg-data-LFN',lfn)

```

Notice that this file takes the same arguments as `save_3x3_nersc` plus an addition one called `lfn`. `lfn` stands for *lattice file name*.

```
1 lfn://myCollab/myFilename
```

The `lfn` is intended to be a Unique Resource Identifier (URI) but it not a Universal Resource Locator (URL). The prefix `lfn` is not a protocol like `http` or `ftp`.

## 3.7 SciDAC file format

The SciDAC format is used primarily for storing propagators. It uses LIME and it packages the following segments:

- `scidac-binary-data`: the actual binary data
- `scidac-private-file-xml`
- `scidac-private-record-xml`

We do not describe it here because this file type is not used by the tools which are described in this manual. Yet we observe that `qcdutils_get` can convert this files into FermiQCD propagators.

## 4 Running physics algorithms with `qcdutils_run.py`

`qcdutils_run.py` is a program for downloading, compiling, and running FermiQCD [3, 4]. FermiQCD is a library for parallel Lattice QCD algorithms. The library has been improved over time and it now includes algorithms for visualization of Lattice QCD data. You can learn more about LatticeQCD from refs. [10, 11, 12]. You can learn more about FermiQCD from:

```
http://fermiqcd.net
```

After you download `qcdutils`, run the following command:

```
1 python qcdutils_run.py -download
```

This creates a local folder called “`fermiqcd`”, download the latest FermiQCD source from the google code repository:

```
1 http://code.google.com/p/fermiqcd
```

The source include a file “`fermiqcd.cpp`” file, which can parse command line arguments and run various physics algorithms, some described in this section. `qcdutils_run.py` compiles this source file and stores the compiled one in:

```
1 fermiqcd/fermiqcd.exe
```

Notice the `.exe` extension is used on all supported platforms.

`qcdutils_run` requires `g++` and you need to install it separately.

Now you can run physics algorithms with:

```
1 python qcdutils_run.py [options]
```

`qcdutils_run.py` internally calls `fermiqcd.exe` and pass its [options] along.

You can learn more about the FermiQCD options with

```
1 python qcdutils_run.py -h
```

The output is reported in the appendix but you are encouraged to run it yourself with the latest code.

`qcdutils_run.py` simply passes its command line arguments to “`fermiqcd.exe`” which parses and calls the corresponding algorithms. Some arguments are special (`-download`, `-compile`, `-mpi`, `-options`, `-h`) because they are handled by `qcdutils_run` directly. In particular a call

to `-options` introspects the source of “fermiqcd.cpp” and figures out which arguments are supported.

Notice that FermiQCD can do more of what `qcdutils_run` can access. For example it supports staggered fermions (including asqtad), staggered mesons, and domain wall fermions. It can do visualizations using those fields too, but that is not discussed here.

You can fork “fermiqcd.cpp” and force “`qcdutils_run`” to use your own source code:

```
1 python qcdutils_run.py -compile -source myownfermiqcd.cpp
```

## 4.1 Running in parallel

There are two ways to run FermiQCD in parallel with `qcdutils_run.py`. On an SMP machine you can simply run with the option `-PSIM_NPROCS=<number>`. Here is an example that loads a gauge configuration and computes the plaquette in parallel using 4 processes:

```
1 python qcdutils_run.py -PSIM_NPROCS=4 \
2   -gauge:start=load:load=demo/demo.nersc.mdp -plaquette
```

When running in parallel with `-PSIM_NPROCS`, FermiQCD uses fork to create the parallel processes and uses named pipes for the message passing. Most PCs and workstations do not allow dynamic memory allocation of more than 2GB of contiguous space and this creates problems when processing large lattices, even if there is enough total RAM available. `-PSIM_NPROCS` is designed to overcome this limitation.

FermiQCD with `-PSIM_PROCS` enables you to run parallel processes on one machine even if there is only enough RAM to run one of them at time but not all of them concurrently. This is because only one of the parallel processes needs to be loaded in RAM at once and the OS can automatically switch between processes by swapping to disk. Communications between the parallel processes are also buffered to disk and therefore they work as expected. For example:

- `qcdutils_run.py -PSIM_NPROCS=2` forks two processes (0 and 1)
- p1 is put to sleep and p0 is executed
- If p0 sends data to p1 the data is stored in a named pipe
- When p0 is completed or attempts to receive data it is put to sleep
- When p0 is put to sleep, p1 is loaded in RAM and continues execution.

- p1 can receive the data sent from p0 by reading from the named pipe.

While this is not very efficient, it does allow to run most algorithms even when there is not enough RAM available. The communication patterns are implemented in ways that avoid deadlocks.

A better option is to use MPI and this is the preferred option for production runs. If you want to use MPI, it must be pre-installed on your system separately. On Debian/Ubuntu Linux machines this is done with:

```

1 sudo apt-get install mpich2
2 cd ~
3 touch .mpd.conf
4 chmod 600 .mpd.conf
5 mpd &
```

In order to use it from `qcdutils_run` you need to recompile FermiQCD with MPI:

```

1 python qcdutils_run.py -compile -mpi
```

This makes an mpi-based executable for FermiQCD:

```

1 fermiqcd/fermiqcd-mpi.exe
```

You can run it with

```

1 python qcdutils_run.py -mpi=4 \
2     -gauge:start=load:load=demo/demo.nersc.mdp -plaquette
```

Internally it calls `mpirun`.

## 4.2 General syntax

The main options of “`qcdutils_run.py`” are:

- `-gauge`: creates, loads, and saves gauge configurations
- `-plaquette`: computes the average plaquette
- `-plaquette_vtk`: generates images of the plaquette density
- `-polyakov`: computes Polyakov lines
- `-polyakov_vtk`: computes images from polyakov lines

- `-topcharge`: computes the total topological charge
- `-topcharge_vtk`: generates images of the topological charge density
- `-cool`: cools the gauge configurations
- `-cool_vtk`: cools the gauge configurations and save images of the topological charge at every step
- `-quark`: computes a quark propagator (different sources are possible)
- `-pion`: computes a pion propagator (and optionally saves images of the pion propagator)
- `-meson`: computes a meson propagator (and optionally saves images of the meson propagator)
- `-current_static`: computes a three points correlation function by inserting a light-light between two heavy-light meson operators (and optionally saves images of the current density)
- `-4quark`: computes all possible contractions of a 4-quark operator between two light mesons.

Each option takes optional attributes in the form `:name=value`. All attributes have default values. The `-pion`, `-meson` and `current_static` operators take an optional `:vtk=true` argument needed to save the VTK files for visualization.

Multple options can be listed and executed together in one run. Although we recommend separating the following operations in different runs:

- Generate gauge configurations,
- Compute propagators on each gauge configuration.
- Measure opeartors by reading previously computed gauge configurations and propagators.

The code described here should be considered and example and other cases can be dealt with by modifying the provided examples.

### 4.3 Creating a cold or hot gauge configuration

You can create a cold gauge configuration with the following command

```
1 python qcdutils_run.py -gauge:start=cold:nt=16:nx=4:ny=4:nz=4:nc=3
```

The `-gauge` option sets the gauge parameters of FermiQCD. The option is followed by parameters separated by a colon. All parameters have default values.

`qcdutils_run.py` creates a cold gauge configuration with volume `nt=16:nx=4:ny=4:nz=4`,  $SU(N_c)$  with `nc=3`, and saves it with the name “cold.mdp”.

The order of the parameters is not important. All parameters have default values. The output lists all parameters which are used.

You can also run

```
1 python qcdutils_run.py -gauge:start=hot:nt=16:nx=4
```

to generate a “hot.mdp” gauge configuration. Notice `nc=3` is the default.

## 4.4 Loading a gauge configuration

The `start` attribute of the `-gauge` option takes four possible values:

- `cold`: makes a cold gauge configuration
- `hot`: makes a hot gauge configuration
- `instantons`: makes a cold configuration containing one instanton and an, optionally, one anti-instanton at given positions.
- `load`: loads one or more gauge configurations (if more than one, it loops over them)

When not set, `start` defaults to `load`, and FermiQCD expects to load input gauge configurations.

In this case, the `load` attribute of the `-gauge` option specifies the pattern of the filenames to read.

You can specify one single gauge configuration by filename or multiple configurations using a glob pattern (for example “\*.mdp”).

Here is an example that loads all gauge configurations in the “demo” folder and computes their average plaquette (`-plaquette`):

```
1 python qcdutils_run.py -gauge:start=load:load=demo/*.mdp -plaquette
```

Similarly if you want to download a stream of NERSC gauge configurations and compute the average plaquette on each of them you can do:

```
1 python qcdutils_get.py -c mdp -4 http://qcd.nersc.org/nersc/api/files/demo
2 python qcdutils_run.py -gauge:load=demo/*.mdp -plaquette > run.log
3 grep plaquette run.log
```

When loading gauge configurations there is no need to specify the volume since FermiQCD reads that information from the input files.

If you peek into “fermiqcd/fermiqcd.cpp” you can find code like this:

```
1 if(arguments.have("-plaquette")) {
2     mdp << "plaquette = " << average_plaquette(U) << endl;
3 }
```

Here `arguments.have("-plaquette")` checks that the option is present and `average_plaquette(U)` performs the computation for the input gauge configuration `U`. `mdp` is the parallel output stream and it double as wrapper object for the MPI communicator.

## 4.5 Heatbath Monte Carlo

Whether you start form a cold, hot or loaded gauge configuration you can generate more by using the `n` attribute. In this example:

```
1 python qcdutils_run.py -gauge:start=cold:beta=4:n=10:therm=100:steps=5
```

FermiQCD starts from a cold configuration, and using the Wilson gauge action [13] (default) generates `n=10` gauge configurations. It perform 100 thermalization steps (*therm*) starting from the cold one and then 5 *steps* separating the one configuration from the next.

It saves the gauge configuration files with progressive names:

```
1 cold.mdp
2 cold.0000.mdp
3 cold.0001.mdp
4 ...
5 cold.0099.mdp
```

If you want to change “cold” prefix of numbered filename you can specify the `prefix` attribute of the `-gauge` option. When this attribute is missing, `prefix` defaults to the name of the starting gauge configuration, i.e. “cold”.

When you start from hot or cold, FermiQCD generates output files in the current working directory. If you start from a loaded file, it generates output files (gauge configurations, propagators, vtk files) in the same folder as the input files.

You can use the optional `alg` attribute to use an improved action or a SSE2 optimized action.

Here is the relevant code in “fermiqcd.cpp”:

```

1 int nconfigs = arguments.get("-gauge", "n", 0);
2 ...
3 for(int n=-1; n<nconfigs; n++) {
4     if(n>=0) {
5         int niter =(n==0)?ntherm:nsteps;
6         if (gauge_action=="wilson")
7             WilsonGaugeAction::heatbath(U,gauge,niter);
8         else if (gauge_action=="wilson_improved")
9             ImprovedGaugeAction::heatbath(U,gauge,niter);

```

Use `-options` to see which algorithms are available. For example you can declare an improved gauge action:

```
1 -gauge:action=wilson_improved:beta=...:zeta=...:u_s=...:u_t=...
```

where  $\zeta$ ,  $u_t$ , and  $u_s$  are the parameters of the improved un-isotropic action defined in ref. [14].

## 4.6 Computing a pion propagator

We define a pion propagator as

$$C_2[t_1] = \sum_{\mathbf{x}} \langle \pi(0, \mathbf{0}) | \pi(+t, \mathbf{x}) \rangle \quad (1)$$

$$= \sum_{\mathbf{x}} \sum_{ij,\alpha\beta\delta\rho} \langle 0 | \bar{q}_a^{i\alpha}(0) \gamma_{\alpha\beta}^5 q_b^{j\beta}(0) \bar{q}_b^{j\delta}(t, \mathbf{x}) \gamma_{\delta\rho}^5 q_a^{i\rho}(t, \mathbf{x}) | 0 \rangle \quad (2)$$

$$= \sum_{\mathbf{x}} \sum_{i,\alpha} |S^{ii,\alpha\alpha}(t, \mathbf{x})|^2 \quad (3)$$

where

$$S^{ij,\alpha\beta}(t, \mathbf{x}) \equiv \langle 0 | \{q^{i\alpha}(0), \bar{q}^{j\beta}(t, \mathbf{x})\} | 0 \rangle \quad (4)$$

is a quark propagator with source at  $\mathbf{0}$ . Here  $a$  and  $b$  label quark flavours,  $i$  and  $j$  label color indexes,  $\alpha$ ,  $\beta$ ,  $\delta$ ,  $\rho$  label spin indexes. Notice we used the known identity

$$\langle 0 | \{ q^{i\alpha}(t, \mathbf{x}), \bar{q}^{j\beta}(0) \} | 0 \rangle = \sum_{\rho\delta} \gamma_{\alpha\rho}^5 S^{*,ji,\delta\rho}(t, \mathbf{x}) \gamma_{\delta\beta}^5 \quad (5)$$

You can compute C2 using the following syntax:

```
1 python qcdutils_run.py \
2     -gauge:start=cold:beta=4:n=10:steps=5:therm=100 \
3     -quark:kappa=0.11:c_sw=0.4:save=false -pion > run.log
```

`qcdutils_run` calls “fermiqcd/fermiqcd.exe” which generates 10 gauge configurations and, for each, computes a quark propagators with the given values of  $\kappa$  and  $c_{SW}$  using a fast implementation of the clover action (another attribute that can be set) and compute the pion propagator.

The `-quark` option loops over the  $j, \beta$  indexes and computes the  $S^{ij,\alpha\beta}(t, \mathbf{x})$ . The `-pion` options loops over the  $i, \alpha$  indexes and for every  $t$  computes the zero momentum Fourier transform in  $\mathbf{x}$  of eq. 2.

Notice that by default `qcdutils` saves all the  $S$  components. We can avoid it with `save=false`.

The pion propagator for each gauge configuration can be found in the output log file.

```
1 grep C2 run.log
```

The output of `qcdutils.run` in this case is looks like the following.

```
1 C2[0] = 14.4746
2 ...
3 C2[15] = 0.794981
4 C2[0] = 14.4746
5 ...
6 C2[15] = 0.794981
7 ...
```

For each  $t$ , `C[t]` takes a different value on each gauge configuration.

In some of the following example we rely on the output pattern:

```
1 C2[...] = ...
```

Later we show how to use `vtk=true` option to save the propagator as function of  $x$  and visualize it. We also show tools to automate the analysis of logfiles like “run.log”.

If you peek into “fermiqcd.cpp” you find the following code that computes the pion propagator:

```

1 for(int a=0; a<4; a++) {
2     for(int i=0; i<nc; i++) {
3         psi = 0;
4         if (on_which_process(U.lattice(),0,0,0,0)==ME) x.set(0,0,0,0);
5         psi(x,a,i)=1;
6         psi.update();
7         [...]
8         mul_invQ(phi,psi,U,quark,abs_precision,rel_precision);
9         [...]
10        if (arguments.have("-pion")) {
11            [...]
12            forallsitesandcopies(x) {
13                for(int b=0; b<4; b++)
14                    for(int j=0; j<nc; j++) {
15                        tmp = real(phi(x,b,j)*conj(phi(x,b,j)));
16                        pion[(x(TIME)-t0+NT)%NT] += tmp;
17                        Q(x) += tmp;
18                    }
19            }
20        }
21    }

```

Notice the field  $Q$  which is used in the next section. It is used for 3D visualizations of the propagator.

## 4.7 Action and inverters

You can change the action by setting the `action` attribute of the `-quark` option to one of the following: `clover_fast`, `clover_slow`, `clover_sse2`. The first of them is the fastest portable implementation. The second is a slower but more readable one. The first two support arbitrary  $SU(N_c)$  gauge groups while the latter is optimized in assembler for  $N_c = 3$ . All of them support clover, and un-isotropic actions. The attributes are

<code>kappa</code>	$\kappa$
<code>kappa_s</code>	$\kappa_s$
<code>kappa_t</code>	$\kappa_t$
<code>c_SW</code>	$c_{SW}$
<code>c_E</code>	$c_E$
<code>c_B</code>	$c_B$

If separate values for  $\kappa_{s,t}$  are not specified,  $\kappa$  is used for both.  $c_E$  is the coefficient that multiplies the electric part of the SW term,  $c_B$  multiplies the magnetic part.  $c_{SW}$  defaults to 0.

The inverter can be specified using the `alg` attribute of the `-quark` option and it can be one of the following: `bicgstab`, `minres`, `bicgstabvtk`, `minresvtk`. The meaning of the first two is obvious. The second two perform the extra task of saving the field components and the residue at every step of the inversion as a VTK file.

The `-quark` option also takes an optional `source_type` attribute which can be `point` or `wall` and, if point, a `source_point` attribute to position the source at `zero` or the `center` of the lattice. It also takes the optional `smear_steps` and `smear_alpha` which are used to smear the sink.

The relevant code in “fermiqcd.cpp” is:

```

1   for(int a=0; a<4; a++) {
2       for(int i=0; i<nc; i++) {
3           if(source_type==''point '') {
4               psi = 0;
5               if (on_which_process(U.lattice(),t0,x0,y0,z0)==ME) {
6                   x.set(t0,x0,y0,z0);
7                   psi(x,a,i)=1;
8               }
9           }
10          [...]
11          psi.update();
12          [...]
13          if (arguments.get(``-quark '', ''load '', ''false|true '')== ''true '') {
14              phi.load(quarkfilename);
15          } else {
16              mul_invQ(phi,psi,U,quark,abs_precision,rel_precision);
17              phi.save(quarkfilename);
18          }
19          [...]
20          if(use_propagator) {
21              forallsites(x) {
22                  forspincolor(b,j,nc) {
23                      S(x,a,b,i,j) = phi(x,b,j);
24                  }
25              }
26          }
27      }

```

Notice that in FermiQCD inverters are action agnostic. A call to `mul_Q(phi,psi,U,...)` computes  $\phi = Q[U]\psi$  where  $Q$  is the selected action for the type of fermion  $\psi$  (in this document we deal only with wilson type fermions but it works with staggered and domain wall too). A call to `mul_invQ(phi,psi,U,...)` computes  $\phi = Q^{-1}[U]\psi$  using the same  $Q$  and the selected inverter. There is no code in the inverter which is action specific.

## 4.8 Meson propagators

Given a meson created by  $\bar{q}\Gamma q |0\rangle$ , a meson propagator can be defined as follows:

$$C_2[t_1] = \sum_{\mathbf{x}} \langle \Gamma^{source}(0, \mathbf{0}) | \Gamma^{sink}(+t, \mathbf{x}) \rangle \quad (6)$$

$$= \sum_{\mathbf{x}} \sum_{ij,\alpha\beta\delta\rho} \langle 0 | \bar{q}_a^{i\alpha}(0) \Gamma_{\alpha\beta}^{source} q_b^{j\beta}(0) \bar{q}_b^{j\delta}(t, \mathbf{x}) \Gamma_{\delta\rho}^{sink} q_a^{i\rho}(t, \mathbf{x}) | 0 \rangle \quad (7)$$

$$= \sum_{\mathbf{x}} \dots S^{ij,\beta\delta}(t, \mathbf{x}) (\Gamma^{sink} \gamma^5)_{\delta\rho} S^{*ij,\alpha\rho}(t, \mathbf{x}) (\gamma^5 \Gamma^{source})_{\alpha\beta} \quad (8)$$

The command to compute an arbitrary meson propagator and reuse the previously computed propagators (the code assumes different flavours of degenerate quarks, i.e. same mass):

```
1 python qcutils_run.py \
2     -gauge:start=cold:beta=4:n=10:steps=5:therm=100 \
3     -quark:kappa=0.11:c_sw=0.4:save=false \
4     -meson:source_gamma=1:sink_gamma=1 > run.log
```

The `source_gamma` and `sink_gamma` attributes can be specified according to the following table:

source_gamma/sink_gamma	$\Gamma^{source}/\Gamma^{sink}$
I	1
5	$\gamma^5$
0	$\gamma^0$
1	$\gamma^1$
2	$\gamma^2$
3	$\gamma^3$
05	$\gamma^0\gamma^5$
15	$\gamma^1\gamma^5$
25	$\gamma^2\gamma^5$
35	$\gamma^3\gamma^5$
01	$\gamma^0\gamma^1$
02	$\gamma^0\gamma^2$
03	$\gamma^0\gamma^3$
12	$\gamma^1\gamma^2$
13	$\gamma^1\gamma^3$
23	$\gamma^2\gamma^3$

The relevant code in “fermiqed.cpp” is described here:

```

1  if(arguments.have("-meson")) {
2      [...]
3      G1 = Gamma5*parse_gamma(arguments.get("-meson", "source_gamma", ...))
4      G2 = parse_gamma(arguments.get("-meson", "sink_gamma", ...))*Gamma5
5      forspincolor(a,i,U.nc) {
6          forspincolor(b,j,U.nc) {
7              forallsites(x) {
8                  s1=s2=0;
9                  for(int c=0;c<4;c++) {
10                      s1 += S(x,a,c,i,j)*G2(c,b);
11                      s2 += conj(S(x,c,b,i,j))*G1(c,a);
12                  }
13                  tmp = abs(s1*s2);
14                  meson[(x(TIME)-t0+NT)%NT] += tmp;
15                  Q(x) += tmp;
16              }
17          }
18      }
}

```

As before we use a scalar field  $Q$  for data visualization.

In this and the other examples the two quarks are degenerate but it is possible to change one of the quark propagators by simply replacing it in the code for a different one. We leave it

to the reader as an exercise. A next version of “fermiqcd.cpp” will have an option `-quark2` for doing this automatically.

## 4.9 Current insertion

We define it as follows (for two light quarks  $a, b$  and one static quark  $h$ ):

$$\begin{aligned}
 C_{current}[t] &= \sum_{\mathbf{x}} \langle \Gamma_{ha}^{source}(-t, \mathbf{x}) | \bar{q}_a \Gamma^{current} q_b(0) | \Gamma_{bh}^{sink}(+t, \mathbf{x}) \rangle \\
 &= \sum_{\mathbf{x}} \sum_{\dots} \langle 0 | \bar{h}^{i\alpha}(-t, \mathbf{x}) \Gamma_{\alpha\beta} q_a^{i\beta}(-t, \mathbf{x}) \bar{q}_a^{r,\zeta} \Gamma_{\zeta\theta}^{current} q_b^{s\theta} \bar{q}_b^{j\delta}(t, \mathbf{x}) \Gamma_{\delta\rho} h^{i\rho}(t, \mathbf{x}) | 0 \rangle \\
 &= \sum_{\mathbf{x}} \text{tr}(\Gamma^{source} \gamma^5 S^\dagger(-t, \mathbf{x}) \gamma^5 \Gamma^{current} S(t, \mathbf{x}) \Gamma^{sink} H^\dagger(-t, t, \mathbf{x})) \tag{9}
 \end{aligned}$$

Here  $H$  is the heavy quark propagator according to Heavy Quark Effective Theory [15] (from  $(-t, \mathbf{x})$  to  $(t, \mathbf{x})$ ):

$$H(-t, t, x) = \frac{1}{2}(1 + \gamma^0)U_0(-t, x)U_0(-t + 1, x)\dots U_0(t - 1, x) \tag{10}$$

You can compute it with

```

1 python qcutils_run.py \
2   -gauge:start=cold:beta=4:n=10:steps=5:therm=100 \
3   -quark:kappa=0.11:c_sw=0.4:save=false \
4   -current_static:source_gamma=1:sink_gamma=1:current_gamma=I > run.log

```

The relevant code in “fermiqcd.cpp” is:

```

1 G1 = parse_gamma(arguments.get("-current_static", "source_gamma", ...)) *
      Gamma5;
2 G2 = parse_gamma(arguments.get("-current_static", "sink_gamma", ...));
3 G3 = Gamma5*parse_gamma(arguments.get("-current_static", "current_gamma",
      ...));
4 G4 = G2*(1-Gamma[0])/2*G1;
5 forallsites(x)
6   if(x(TIME)>=0) {
7     z.set((NT+2*t0-x(TIME))%NT,x(1),x(2),x(3));
8     forspincolor(a,i,U.nc) {

```

```

9      for spin color(b,j,U.nc) {
10         s1 = s2 = 0;
11         for(int c=0; c<4; c++) {
12             s1 += conj(S(z,c,a,j,i))*G3(c,b);
13             for(int k=0; k<U.nc; k++)
14                 s2 += S(x,b,c,j,k)*G4(c,a)*conj(Sh(x,i,k));
15         }
16         tmp = abs(s1*s2);
17         current[(x(TIME)-t0+NT)%NT] += tmp;
18         Q(x) += tmp;
19     }
20 }
21 }
```

Here  $Sh$  is the product of links from  $-t$  to  $t$  along the time direction.

## 4.10 Four quark operators

Instead of inserting a current we can insert a 4-quark operator between two meson operators (light-light):

$$\begin{aligned}
C_3[t_1][t_2] &= \sum_{\mathbf{x}_1} \sum_{\mathbf{x}_2} \langle \Gamma^{source}(-t_1, \mathbf{x}_1) | \bar{q}_a \Gamma_A q_b \otimes \bar{q}_c \Gamma_B q_d | \Gamma^{sink}(+t_2, \mathbf{x}_2) \rangle \quad (11) \\
&= \text{tr}(\Gamma^{source} \gamma^5 S^\dagger(-t_1, \mathbf{x}_1) \gamma^5 \Gamma_A S(-t_1, \mathbf{x}_1)) \text{tr}(\Gamma^{sink} \gamma^5 S^\dagger(t_2, \mathbf{x}_2) \gamma^5 \Gamma_B S(t_2, \mathbf{x}_1)) \\
&\text{or } \text{tr}(\Gamma^{source} \gamma^5 S^\dagger(-t_1, \mathbf{x}_1) \gamma^5 \Gamma_A S(t_2, \mathbf{x}_2) \Gamma^{sink} \gamma^5 S^\dagger(t_2, \mathbf{x}_2) \gamma^5 \Gamma_B S(-t_1, \mathbf{x}_1))
\end{aligned}$$

The *or* indicates that there are two possible contractions. FermiQCD computes both of them and writes them separately in the output.

Here  $\Gamma_A \otimes \Gamma_B$  is the spin/color structure of the 4-quark operator. We are also ignoring the contractions that corresponds to disconnected diagrams.

We can compute  $\Gamma_A \otimes \Gamma_B$  for  $\gamma_5 \otimes \gamma_5$  in spin and  $\mathbf{1} \otimes \mathbf{1}$  in color (`5Ix5I`) with:

```

1 python qcutils_run.py \
2   -gauge:start=cold:beta=4:n=10:steps=5:therm=100 \
3   -quark:kappa=0.11 -4quark:source=1:operator=5Ix5I > run.log
```

In this example, `source=1` indicates that  $\Gamma^{source} = \Gamma^{sink} = \gamma^1$ .

This generates the following output, repeated for each of the 10 gauge configurations:

```

1 C3 [0] [0] = 9.12242
2 C3 [0] [1] = 0.485189
3 ...
4 C3 [15] [15] = 9.12242

```

Notice the program computes the two contractions of the operator and writes one in `C3` and one in `C3x`.

Instead of `source=1` you can use any of the operators defined for mesons.

Instead of  $5I \times 5I$  4-quark operator you can use any the following other operators:  $5Ix5I$ ,  $0Ix0I$ ,  $1Ix1I$ ,  $2Ix2I$ ,  $3Ix3I$ ,  $05Ix05I$ ,  $15Ix15I$ ,  $25Ix25I$ ,  $35Ix35I$ ,  $01Ix01I$ ,  $02Ix02I$ ,  $03Ix03I$ ,  $12Ix12I$ ,  $13Ix13I$ ,  $23Ix23I$ ,  $5Tx5T$ ,  $0Tx0T$ ,  $1Tx1T$ ,  $2Tx2T$ ,  $3Tx3T$ ,  $05Tx05T$ ,  $15Tx15T$ ,  $25Tx25T$ ,  $35Tx35T$ ,  $01Tx01T$ ,  $02Tx02T$ ,  $03Tx03T$ ,  $12Tx12T$ ,  $13Tx13T$ ,  $23Tx23T$ . Here the numerical part represents the  $\Gamma \otimes \Gamma$  stucture of the 4-quark operator, the  $I$  or  $T$  represents its color structure.  $TxT$  stands for  $\sum_a T^a \otimes T^a$  with  $T^a = \lambda^a/2$ , and  $\lambda^a$  is the  $SU(3)$  generator.

Here is the relevant source code in “fermiqcd.cpp”:

```

1 mdp_matrix G = parse_gamma(arguments.get("-4quark", "source", ...));
2 forspincolor(a,i,U.nc) {
3     for(int c=0; c<4; c++)
4         for(int d=0; d<4; d++)
5             if(G(c,d)!=0)
6                 forallsites(x)
7                     for(int k=0; k<U.nc; k++)
8                         open_prop[a][b][i][j][(x(TIME)-t0+NT)%NT] +=
9                             S(x,a,c,i,k)*conj(S(x,b,d,j,k))*G(c,d);
10    string op4q = arguments.get("-4quark", "operator", ...);
11    if(arguments.have("-4quark")) {
12        for(int a=0; a<4; a++)
13            for(int b=0; b<4; b++)
14                for(int c=0; c<4; c++)
15                    for(int d=0; d<4; d++) {
16                        mdp_complex g1 = G1(b,a);
17                        mdp_complex g2 = G2(d,c);
18                        if(g1!=0 && g2!=0)
19                            for(int i=0; i<U.nc; i++)
20                                for(int j=0; j<U.nc; j++)
21                                    if(!rotate) {
22                                        c3a+=abs(open_prop[a][b][i][j][t1s]*g1*
23                                                open_prop[c][d][j][j][t2s]*g2);
24                                        c3b+=abs(open_prop[c][b][j][i][t1s]*g1*
25                                                open_prop[a][d][i][j][t2s]*g2);
26                                    } else

```

```

27             [...]
28         }
29     }
30 }
31 }
```

Notice the two contractions are computed separately. The case `rotate==true` corresponds to the TxT color structure.

## 5 Images and movies with `qcdutils_vis.py` and `qcdutils_vtk.py`

In this section we describe how to make 3D visualizations using VisIt [6] and how to embed visualizations into web pages using “processing.js” [7].

VisIt is a visualization software developed at Lawrence Livermore National Lab based on the VTK toolkit. It provides a GUI which can be used to open the VTK files created by FermiQCD (or other scientific program) in interactive mode, but it can also be scripted using the Python language.

“processing.js” is a lightweight javascript library that allows drawing on an HTML canvas using the *processing* language or the javascript language.

`qcdutils` uses meta-programming to generate VisIt scripts (`qcdutils_vis`) or processing.js scripts (`qcdutils_vtk`). The former is more flexible and is more appropriate for making high resolution images. The latter makes it easy to embed 3D visualizations into web pages.

Using VisIt is intuitive but there are certain tasks which can be repetitive. For example if you have multiple VTK files containing topological change density (or any other scalar field), you have to determine the optimal threshold values for the contour plots. If you have many files you may want to interpolate between them for a smoother visualization. `qcdutils_vis` helps with these tasks. In particular it can:

- Split VTK files containing multiple time-slices into separate VTK files, one for each slice.
- Interpolate each couple of consecutive VTK files and make new ones in between. This is necessary for smoother visualizations.
- Compute automatic thresholds values for contour plots.
- Resample the points by interpolating between the.

- Generate VisIt scripts which converts VTK files to JPEG format (these script can be saved, edited, and reused).
- Pipe the above operations and run them for multiple files.

Images generate in this way can be assembled into mpeg4 (or quicktime or avi) movies using ffmpeg (an open source tool that is distributed with VisIt) but there are other and better tools available. We strongly recommend “MPEG Streamclip”. It is much faster, robust, and much easier to properly configure than ffmpeg.

## 5.1 About VTK file format

There are many VTK file formats. `qcdutils` uses the binary VTK file format described below to store scalar fields, usually by timeslices.

A typical file has the following content:

```

1 # vtk DataFile Version 2.0
2 filename.vtl
3 BINARY
4 DATASET STRUCTURED_POINTS
5 DIMENSIONS 4 4 4
6 ORIGIN      0 0 0
7 SPACING     1 1 1
8 POINT_DATA 64
9 SCALARS scalars_t0 float
10 LOOKUP_TABLE default
11 [binary data]
12 SCALARS scalars_t1 float
13 LOOKUP_TABLE default
14 [binary data]
15 ...

```

It consists of an ASCII header declaring the 3D dimensions (4 4 4) and the total number of points ( $4 \times 4 \times 4 = 64$ ). This is followed by blocks representing the time-slices. Each block has its own ASCII header:

```

1 SCALARS scalars_t0 float
2 LOOKUP_TABLE default

```

followed by binary data (64 floating point numbers).

`scalars_t0`, `scalars_t1`, etc. are the names of the fields as stored by FermiQCD. When time-slices are extracted by `qcdutils_vis` the slices are renamed as `slice`.

Given any VTK file, for example `demo.vtk` we can visualize it using `qcdutils_vis.py` using the following syntax:

```
1 python qcdutils_vis.py -r 'scalars_t0' -p default demo.vtk
```

`qcdutils_vis.py` generates images in JPEG format.

Similarly we can visualize by creating an interactive 3D web page:

```
1 python qcdutils_vtk.py -u 0.10 -l 0.90 demo.vtk
```

If the filename is a glob pattern (`*.vtk`), both tools loop and process all files matching the pattern.

`qcdutils_vtk` computes the range of values in the scalar field from the maximum to the minimum. `-u 0.10` indicates we want an isosurface at 10% from the max and `-l 0.90` indicates we want another isosurface at 90% from the max (10% from the min). It is also possible to specify the colors of the iso-surfaces.

`qcdutils_vtk` generates HTML files with the same as the input VTK files followed by the `.html` postfix. The isosurfaces are computed by the Python program itself but the representation of the isosurfaces is embedded in the html file, together with the “processing.js” library, and with custom JS code. These files are not static images. You can rotate them in the browser using the mouse.

## 5.2 Plaquette

As an example, we want to make a movie of the plaquette as function of the time-slice. We follow this workflow:

- Load a gauge configuration.
- Compute the plaquette at each lattice site.
- Save the plaquette as a VTK file.
- Split the VTK file into one file per time-slice.
- Interpolate the timeslices to generate more frames.
- Generate contour plots for each frame and save them as JPEG files.

This can be done in two steps. Step one:

```
1 python qcdutils_run.py \
2     -gauge:load=demo/demo.nersc.mdp \
3     -plaquette_vtk
```

This command uses FermiQCD to load the gauge configurations. For each of them it computes the trace of the average plaquette at each lattice site, and generates one VTK file contain the 4D scalar for the plaquette. This file is saved in a new file with the same prefix as the input but ending in “.plaquette.vtk”.

Step two:

```
1 python qcdutils_vis.py -s '*' -i 9 -p default 'demo/*.plaquette.vtk'
```

It reads all files matching the pattern “demo/\*.plaquette.vtk”, extracts all time-slices with names matching “\*” (all time slices), and interpolates each couple of VTK files by adding 9 more frames (-i 9), then generates a VTK script that reads each VTK file, resamples it, and stores contour plots in JPEG files with consecutive file filenames..

The generated script has a unique name which looks like this:

```
1 qcdutils_vis_2fac1b86-5b86-42ee-8552.py
```

qcdutils\_vis writes and runs the script. It saves it for you in case you want to read and modify it.

When it runs, it loops over all the frames, resamples them, computes the contour plots and saves each frame into one JPEG image:

```
1 qcdutils_vis_2fac1b86-5b86-42ee-8552_0000.00.jpeg
2 qcdutils_vis_2fac1b86-5b86-42ee-8552_0001.01.jpeg
3 ...
4 qcdutils_vis_2fac1b86-5b86-42ee-8552_0003.00.jpeg
```

Here 0000, 0001, 0002, 0003 are the original frames (timeslices) and the. .01, .02, ..., .09 are the interpolated ones.

Notice that the -i 9 option is very important to obtain smooth sequences of images to be assembled into movies.

The option

```
1 -p default
```

is equivalent to

```
1 -p 'AnnotationAttributes [] ; ResampleAttributes [] ; ContourAttributes [] '
```

Here `Annotation`, `Resample` and `Contour` are VisIt functions. Using `-p` you can set the attributes for each functions.

For example, to remove the bounding box you would replace

```
1 AnnotationAttributes []
```

with

```
1 AnnotationAttributes [axes3D.bboxFlag=0]
```

To increase the re-sampling points from 100 to 160 you would replace:

```
1 ResampleAttributes []
```

with

```
1 ResampleAttributes [samplesX=160; samplesY=160; samplesZ=160]
```

To change the color of the 9th contour to Orange, you would replace:

```
1 ContourAttributes []
```

with

```
1 ContourAttributes [SetMultiColor(9,orange)]
```

The argument of the `<function>Attributes[...]` are VisIt attributes and they are described in the VisIt documentation.

The relevant page of code in “fermiqcd.cpp” that computes the VTK plaquette is here:

```
1 void plaquette_vtk(gauge_field& U, string filename) {
2   mdp_field<mdp_real> Q(U.lattice());
3   mdp_site x(U.lattice());
4   forallsites(x) if(x(0)==0) {
5     Q(x)=0;
6     for(int mu=0; mu<4; mu++)
7       for(int nu=mu+1; nu<4; nu++)
8         Q(x)+=real(trace(plaquette(U,x,mu,nu)));
9   }
10   Q.save_vtk(filename,-1);
11 }
12 [...]
13 if (arguments.have("-plaquette_vtk")) {
```

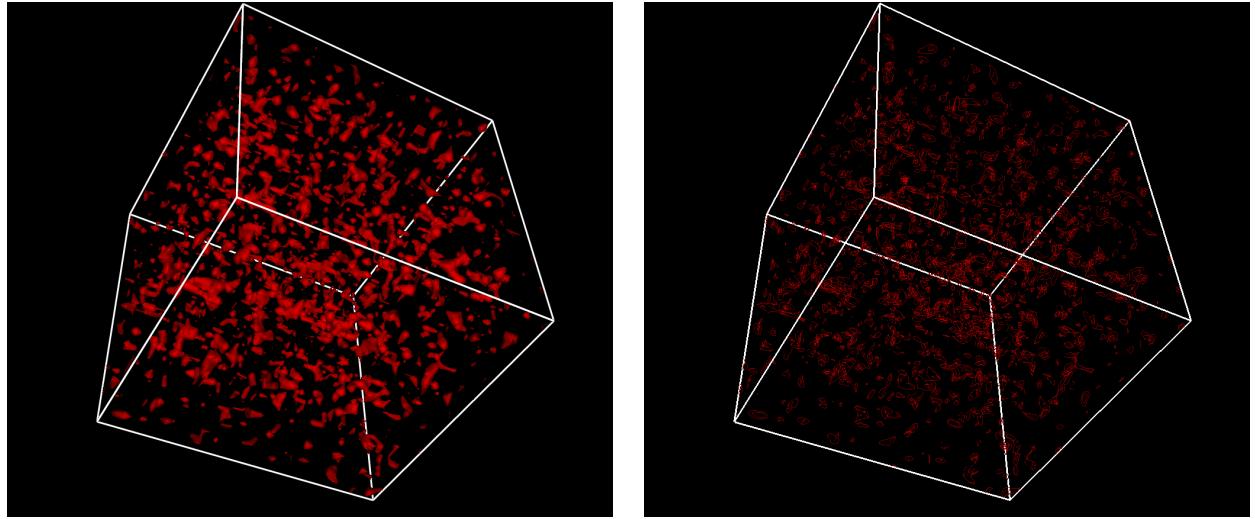


Figure 4: Visualization of a contour plot for the average plaquette (left) and the intersection of the contours with the bounding box (right)

```
14     plaquette_vtk(U,newfilename+" .plaqette.vtk");
15 }
```

Notice how the plaquette is computed for each  $x$ , summed over  $\mu, \nu$ , stored in a scalar field  $Q(x)$ , and then saved in a file. This strategy can be used to visualize any FermiQCD scalar field with minor modifications of the source.

### 5.3 Topological charge density

Similarly to the average plaquette we can make images corresponding to the topological charge density.

To generate the topological charge density we need to cool the gauge configurations (`-cool`) and then compute the topological charge (`-topcharge_vtk`):

```
1 python qcutils_run.py \
2     -gauge:load=demo/demo.nersc.mdp \
3     -cool:steps=20 -topcharge_vtk
1 python qcutils_vis.py -s '*' -i 9 -p default 'demo/*.topcharge.vtk'
```

The relevant code in “fermiqcd.cpp” is below:

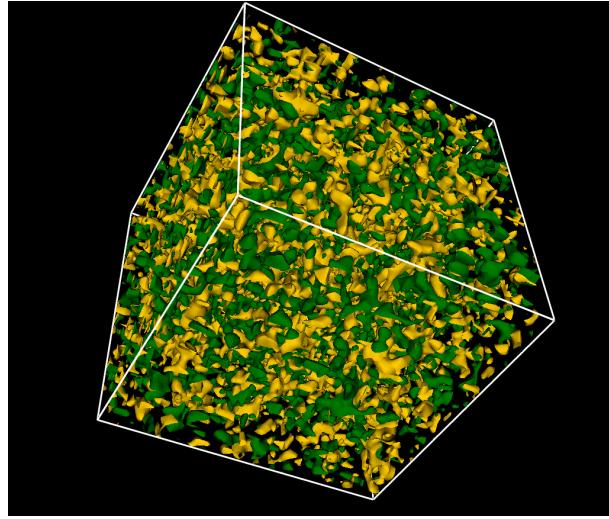


Figure 5: Visualization of the topological charge density.

```

1 if (arguments.have("-topcharge_vtk")) {
2     float tc = topological_charge_vtk(U,newfilename+".topcharge.vtk",-1);
3     mdp << "topcharge = " << tc << endl;
4 }
```

`topological_charge_vtk` is defined in “`fermiqcd.topological_charge.h`”. The `-1` arguments indicates we want to save all time slices. The actual code to compute the topological charge density is:

```

1 void topological_charge(mdp_field<float> &Q, gauge_field &U) {
2     compute_em_notrace_field(U);
3     mdp_site x(U.lattice());
4     forallsitesandcopies(x) {
5         Q(x)=0;
6         for(int i=0; i<U.nc; i++)
7             for(int j=0; j<U.nc; j++)
8                 Q(x)+=real(U.em(x,0,1,i,j)*U.em(x,2,3,j,i)-
9                             U.em(x,0,2,i,j)*U.em(x,1,3,j,i)+
10                            U.em(x,0,3,i,j)*U.em(x,1,2,j,i));
11     }
12     Q.update();
13 }
```

Here `U.em` is the electro-magnetic field computed from `U`.

## 5.4 Cooling

Sometimes we may want to see the changes in the topological charge density as the configuration is cooled down. This requires computing the topological charge density at every cooling step. This can be done with the `-cool_vtk` option:

```
1 python qcdutils_run.py \
2     -gauge:start=load:load=demo/demo.nersc.mdp \
3     -cool_vtk:cooling=10 > run.log

1 python qcdutils_vis.py -r 'scalars_t0' -i 9 -p default 'demo/*.cool???.vtk'
```

The `-cool_vtk` option creates VTK files ending in “cool00.vtk”, “cool01.vtk”,..., “cool49.vtk”. To make a smooth movie we do not break files into time-slices (no `-s` option) but instead we extract the same slice for every file (`-r 'scalars_t0'`). Then we interpolate the frames (`-i 9`).

The above code generates JPEG images showing different stages of cooling of the data. You can see some of the images in fig. 6

The relevant code in “fermiqcd.cpp” is here:

```
1 void cool_vtk(gauge_field& U, mdp_args& arguments, string filename) {
2     if (arguments.get("-cool","alg","ape")=="ape")
3         for(int k=0; k<arguments.get("-cool_vtk","n",20); k++) {
4             ApeSmearing::smear(U,
5                 arguments.get("-cool_vtk","alpha",0.7),
6                 arguments.get("-cool_vtk","steps",1),
7                 arguments.get("-cool_vtk","cooling",10));
8             topological_charge_vtk(U,filename+".cool"+tostring(k,2)+".vtk",0);
9         }
10    else
11        mdp.error_message("cooling algorithm not supported");
12 }
```

The smearing algorithm is in the “topological\_charge\_vtk” file:

```
1 class ApeSmearing {
2     public: static void smear(gauge_field &U,
3                             mdp_real alpha=0.7,
4                             int iterations=20,
5                             int cooling_steps=10) {
6         gauge_field V(U.lattice(),U.nc);
7         mdp_site x(U.lattice());
8         for(int iter=0; iter<iterations; iter++) {
```

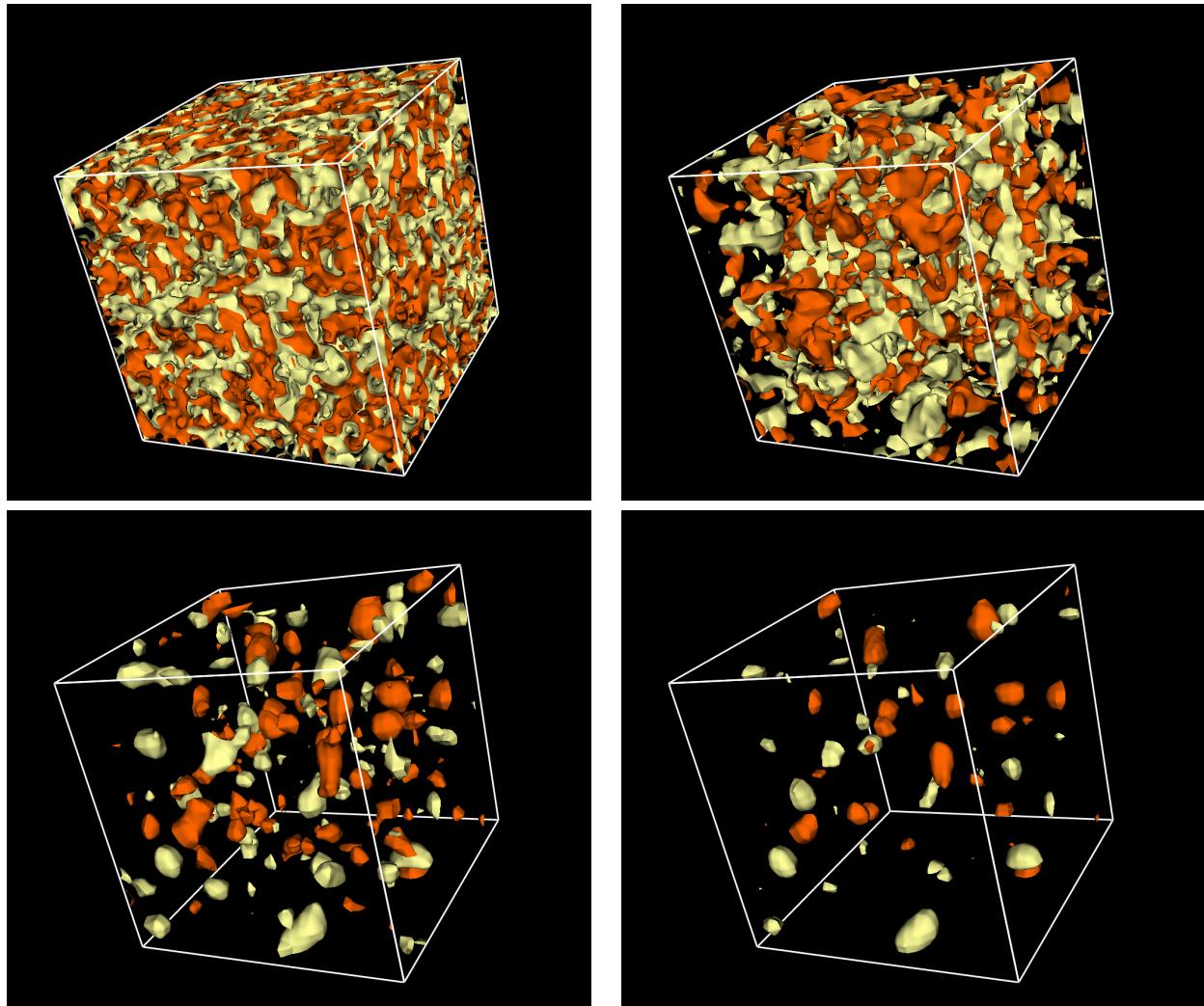


Figure 6: Visualization of the topological charge density at different cooling stages.

```

9   cout << "smearing step " << iter << "/" << iterations << endl;
10  V=U;
11  for(int mu=0; mu<4; mu++) {
12      forallsites(x) {
13          U(x,mu)=(1.0-alpha)*V(x,mu);
14          for(int nu=0; nu<U.ndim; nu++)
15              if(nu!=mu)
16                  U(x,mu)+=(1.0-alpha)/6*
17                      (V(x,nu)*V(x+nu,mu)*hermitian(V(x+mu,nu))+
```

```

18         hermitian(V(x-nu,nu))*V(x-nu,mu)*V((x-nu)+mu,nu));
19     U(x,mu)=project_SU(U(x,mu),cooling_steps);
20 }
21 }
22 U.update();
23 }
24 }
25 };

```

## 5.5 Polyakov lines

A Polyakov line is the trace of the product of the gauge links along the time direction, therefore it is a 3D complex field. Here we are interested in the real part only (the image part is qualitatively the same).

We can visualize Polyakov lines using the `-polyakov_vtk` option:

```

1 python qcutils_run.py \
2     -gauge:load=demo/demo.nersc.mdp \
3     -polyakov_vtk

```

which we can convert to images with:

```

1 python qcutils_vis.py \
2     -r 'scalars_t0' -i 9 -p default 'demo/*.polyakov.vtk'

```

The output is show in fig. 7.

Here is the relevant code in “fermqcd.cpp”:

```

1 void polyakov_vtk(gauge_field& U, string filename) {
2     int L[3];
3     L[0]=U.lattice().size(1);
4     L[1]=U.lattice().size(2);
5     L[2]=U.lattice().size(3);
6     mdp_lattice_space(3,L,
7         default_partitioning<1>,
8         torus_topology,
9         0, 1, false);
10    mdp_matrix_field V(space,U.nc,U.nc);
11    mdp_field<mdp_real> Q(space,2);
12    mdp_site x(U.lattice());
13    mdp_site y(space);
14

```

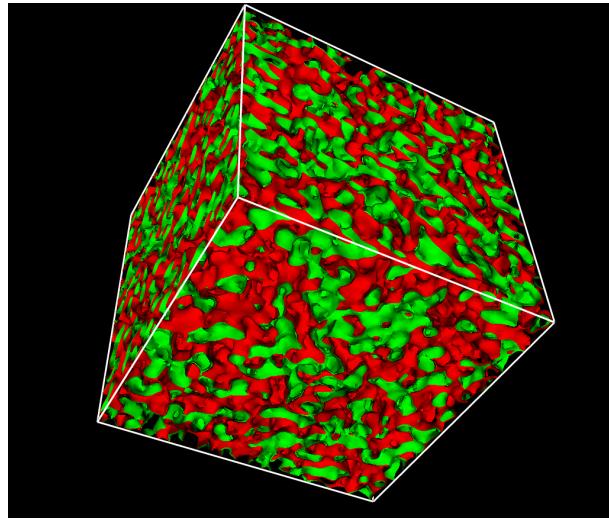


Figure 7: Visualizaition of contour plots for the Polyakov lines. Different colors represent positive and negative values of the real part of the Polyakov lines.

```

15 int k,mu=0,nu=1;
16 mdp_complex s=0;
17
18 forallsites(y) V(y)=1;
19 for(int t=0; t<L[0]; t++) {
20     forallsites(y) {
21         x.set(t,y(0),y(1),y(2));
22         V(y)=V(y)*U(x,0);
23     }
24 }
25
26 forallsites(y) {
27     mdp_complex z=trace(V(y));
28     Q(y,0)=real(z);
29     Q(y,1)=imag(z);
30 }
31 Q.save_vtk(filename,-1,0,0,false);
32 }
33 [...]
34 if (arguments.have("-polyakov_vtk")) {
35     polyakov_vtk(U,newfilename+".polyakov.vtk");
36 }
```

This code is a little different than the previous one. It creates a 3D lattice called `space` which

is a time projection of the 4D space. While  $x$  lives on the full lattice,  $y$  leaves only on the 3D space.  $q$  is a scalar field with two components (real and imaginary part of the Polyakov lines) which lives in 3D space.

## 5.6 Quark propagator

Given any gauge configuration we can visualize quark propagators in two ways. We can use the normal inverter and save the propagator at the end of the inversion for each source/sink spin/color component:

```
1 python qcutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:alg=bicgstab:vtk=true > run.log
```

(here using the `bicgstab`, the Stabilized Bi-Conjugate Gradient). Alternatively we can use a modified inverter which saves the components but also VTK visualization for the field components and the residue at each step of the inversion.

```
1 python qcutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:alg=bicgstab_vtk > run.log
```

Fig. 8 shows different components of a quark propagator on a hot and a cold configuration. From now on we assume the propagator has been computed and we reuse it.

## 5.7 Pion propagator

In a previous section, computed the zero momentum Fourier transform of the pion propagator. Now we want to visualize it for every point in space:

$$Q(t, \mathbf{x}) = \langle \pi_{ab}(0, \mathbf{0}) | \pi_{ab}(+t, \mathbf{x}) \rangle = \sum_{i,\alpha} |S^{ii,\alpha\alpha}(t, \mathbf{x})|^2 \quad (12)$$

This can be done using the `vtk=true` attribute of the `-pion` option:

```
1 python qcutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:load=true \
4   -pion:vtk=true > run.log
```

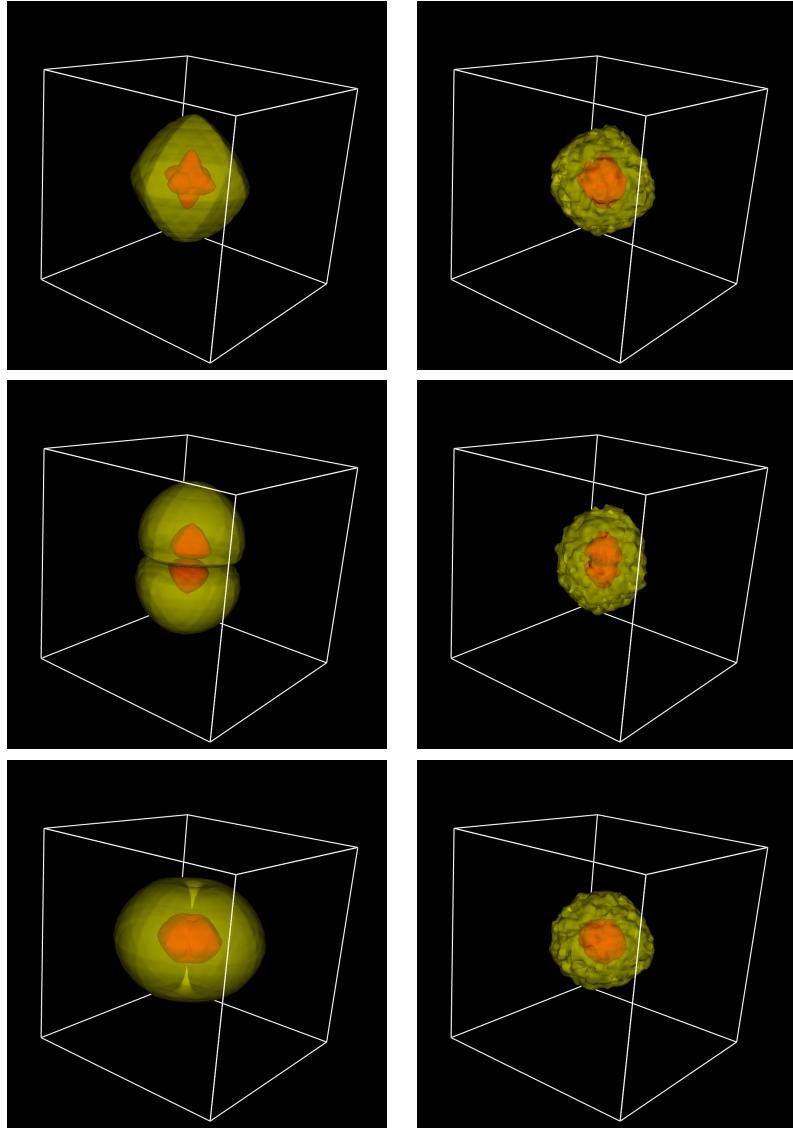


Figure 8: Different components of a quark propagator on a cold gauge configuration (left) and on a thermalized gauge configuration (right). From top to bottom, they show the magnitude of  $S^{\alpha\beta ij}(t, \mathbf{x}) = S^{0000}(0, \mathbf{x})$ ,  $S^{0200}(0, \mathbf{x})$ , and  $S^{0300}(0, \mathbf{x})$ .

Notice the `-quark...:load=true` which reloads the previous propagator. We can now convert the pion VTK visualization into images using `qcdutils_vis`:

```
1 python qcdutils_vtk.py -u 0.01 -l 0.00001 'demo/*.pion.vtk'
```

```
2 python qcduutils_vis.py -s '*' -i 9 -p default 'demo/*.pion.vtk'
```

In this case the `-i 9` option is used to interpolate between time-slices in case the images are to be assembled into a movie.

Examples of images are shown in fig.9

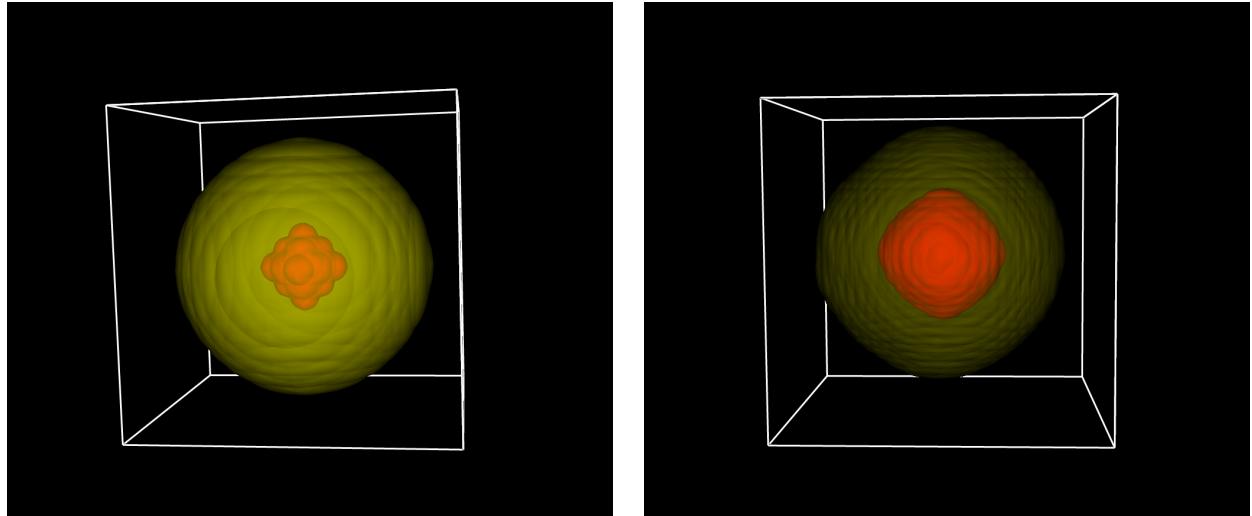


Figure 9: Contour plot for a pion propagator.

## 5.8 Meson propagators

A meson propagator is defined similarly to a pion propagator but it has a different gamma structure:

$$Q(t, \mathbf{x}) = \langle \Gamma_{ab}^{source}(0, \mathbf{0}) | \Gamma_{ab}^{sink}(+t, \mathbf{x}) \rangle \quad (13)$$

$$= \sum S^{ij,\beta\delta}(t, \mathbf{x}) (\Gamma^{sink} \gamma^5)_{\delta\rho} S^{*ij,\alpha\rho}(t, \mathbf{x}) (\gamma^5 \Gamma^{source})_{\alpha\beta} \quad (14)$$

...

We can visualize a Meson propagator using the following code:

```
1 python qcduutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:load=true \
4   -meson:source_gamma=1:sink_gamma=1:vtk=true > run.log
```

and then process the VTK file as in the pion example. In this case  $\Gamma^{source} = \Gamma^{sink} = \Gamma^1$  indicates a vector meson polarized along the  $X$  axis.

## 5.9 Current insertions

We can also visualize the mass density and the charge density of a heavy-light meson by inserting an operator ( $\bar{q}q$  and  $\bar{q}\gamma^0 q$  respectively) in between meson bra-kets.

$$\begin{aligned} Q(t, \mathbf{x}) &= \langle \Gamma_{ha}^{source}(-t, \mathbf{x}) | \bar{q}_a \Gamma^{current} q_b | \Gamma_{bh}^{sink}(+t, \mathbf{x}) \rangle \\ &= \text{tr}(\Gamma^{source} \gamma^5 S^\dagger(-t, \mathbf{x}) \gamma^5 \Gamma^{current} S(t, \mathbf{x}) \Gamma^{sink} H^\dagger(-t, t, \mathbf{x})) \end{aligned} \quad (15)$$

Here we measure the mass distribution for a static vector meson:

```
1 python qcutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:load=true \
4   -current_static:source_gamma=1:sink_gamma=1:current_gamma=I:vtk=true \
5   > run.log
```

Using the same diagram we can compute the spatial distribution of  $B^* \rightarrow B\pi$  by inserting the axial current ( $\bar{q}\gamma^5 q$ ) in between a static  $B$  ( $\bar{q}\gamma^5 h$ ) and a static  $B^*$  ( $\bar{q}\gamma^1 h$ ):

```
1 python qcutils_run.py \
2   -gauge:start=load:load=demo/demo.nersc.mdp \
3   -quark:kappa=0.135:source_point=center:load=true \
4   -current_static:source_gamma=1:sink_gamma=5:current_gamma=5:vtk=true \
5   > run.log
```

A sample image is shown in fig. 10.

## 5.10 Localized instantons

FermiQCD allows the creation of custom gauge configurations with localized topological charge. Here we consider the case of a pion propagator on a single gauge configuration in presence of one t'Hooft instanton (localized lump of topological charge). Here is the code:

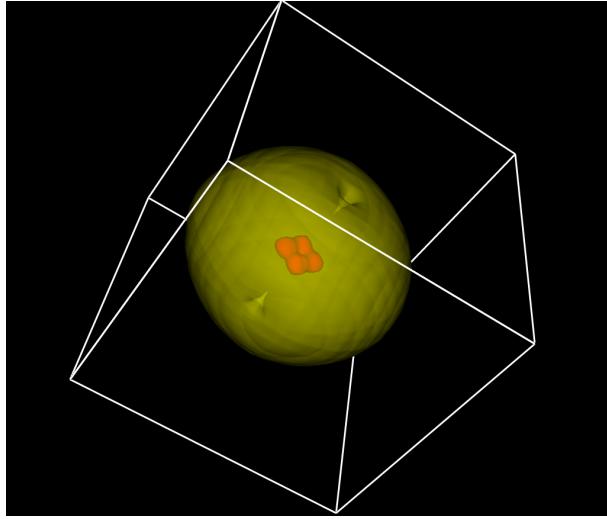


Figure 10: Contour plot for a three points correlation function.

```

1 python qcdutils_run.py \
2     -gauge:start=instantons:nt=20:nx=20:t0=0:x0=4.5:y0=10:z0=10 \
3     -topcharge_vtk \
4     -quark:kappa=0.120:source_point=center \
5     -pion:vtk=true > run.log

```

This code places the center of the instanton at point  $(t_0, x_0, y_0, z_0) = (0, 4.5, 10, 10)$  and then computes a pion propagator with source on time slice 0 but spatial coordinates  $(x, y, z) = (10, 10, 10)$  (center).

Fig. 11 show the pion propagator in presence of the instanton as the instanton nears the center of the propagator. Each image has been generated using the above command by placing the instanton at different locations. The last image shows a superposition of the pion propagator with and without the instanton in order to emphasize the difference. The difference is small but visible. The propagator retracts as the instanton nears. One may say that the quark interacts with the instanton and acquires mass thus making the propagator decrease faster when going through the instanton. Fig. 12 shows the effect of the instanton on individual components of the quark propagator.

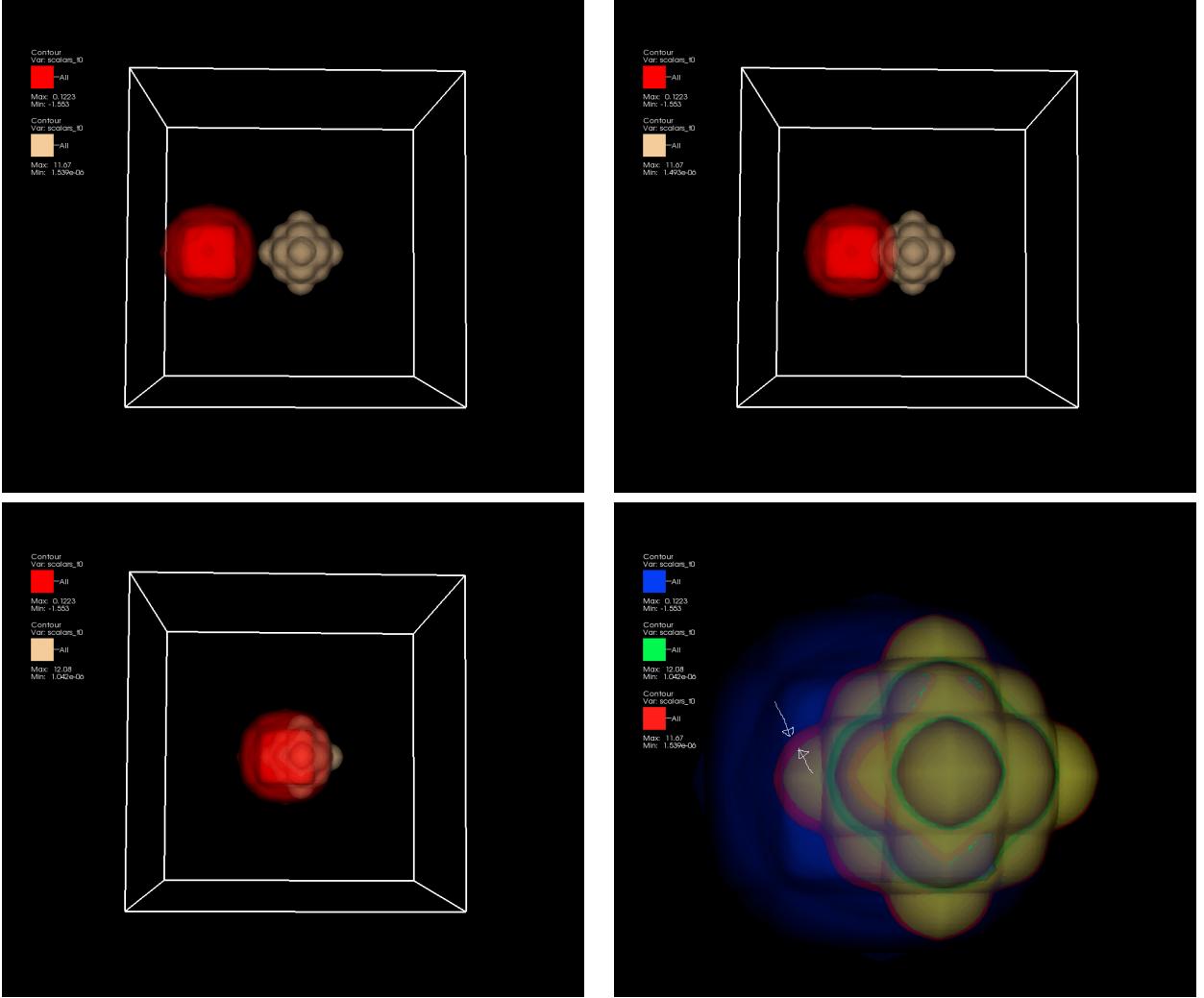


Figure 11: Pion propagator on a semi-cold configuration in presence of one localized instanton. The bottom right image shows the instanton (in blue) and an overlay of the pion propagator with and without the instanton. The difference shows that propagator shrinks as the instanton nears.

## 6 Analysis with `qcdutils_boot.py`, `qcdutils_plot.py`, `qcdutils_fit.py`

The console output of the `qcdutils_run` program consists of human readable text with comments and results of measurements performed on each gauge configuration. Here are some

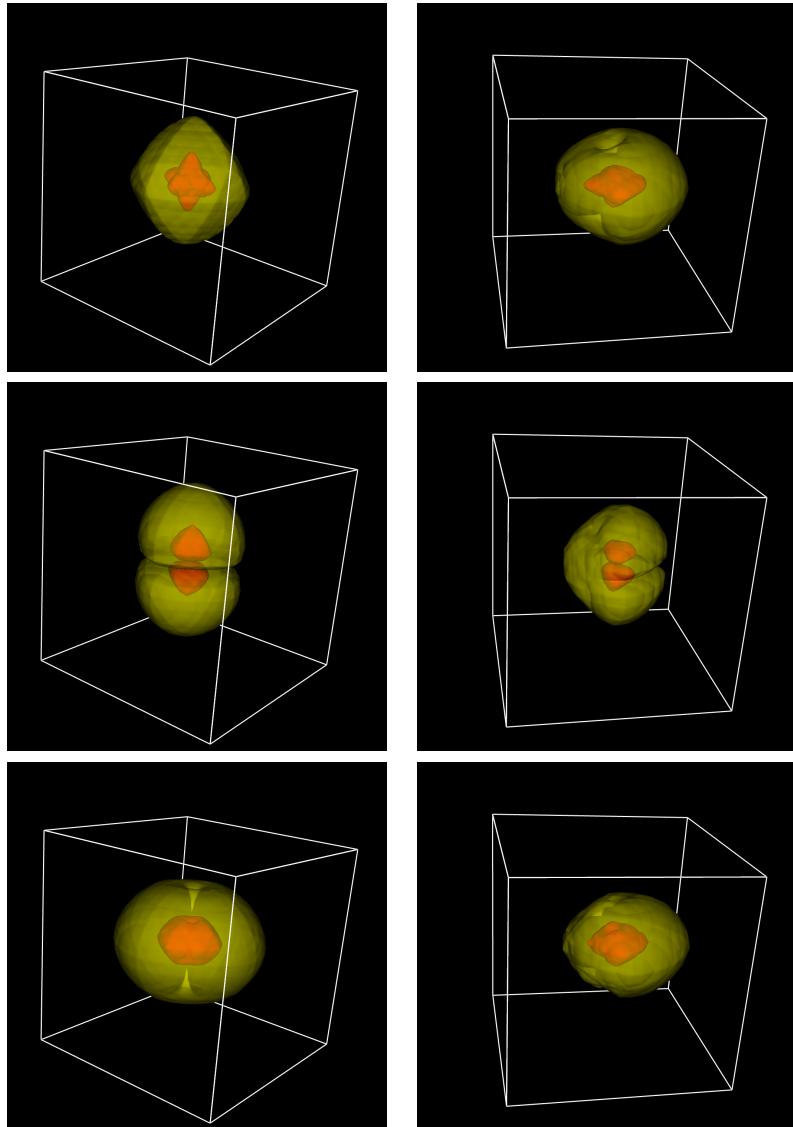


Figure 12: Comparison of quark propagator components on a cold configuration (left) and in presence of a localized instanton (right). The instanton is located as in fig. 11 (bottom-right).

examples of measurements logged in the output:

```

1 ...
2 plaquette = 0.654346
3 C2[0] = 14.5234

```

```

4 C3 [0] [0] = 1.214321
5 C3 [0] [0] = 1.123425
6 ...

```

`qcdutils_boot.py` is a tool that can extract the values for these measurements, aggregate them and analyze them in various ways. For example it computes the average and bootstrap errors [16] of any function of the measurements. `qcdutils_plot.py` is a tool to visualize the results of the analysis. It uses the Python `matplotlib` package, one of the most powerful and versatile plotting libraries available, although `qcdutils_plot.py` uses only a small subset of the available functionality.

Now, let us consider a typical Lattice QCD computation where one or more observables are measured on each Markov Chain Monte Carlo step (on each gauge configuration). We label the observables with  $Y_j$  (gauge configuration, 2-point correlation function for a given value of  $t$ , etc.)

We also refer to each measurements with  $y_{ij}$  where  $i$  labels the gauge configuration and  $j$  labels the observable (same index as  $Y_j$ ).  $y_{i0}$  could be, for example, the plaquette on the  $i$  th gauge configuration.

The expectation value of each one observable is computed by averaging its measurements over the MCMC steps:

$$\bar{Y}_j = \langle 0 | Y_j | 0 \rangle = \frac{1}{N} \sum_i y_{ij} \quad (16)$$

Here  $N$  is the number of the measurements. The statistical error on the average for this simple case can be estimated using the following formula:

$$\delta Y_j \simeq \sqrt{\frac{1}{N(N-1)} \sum_i (y_{ij} - \bar{Y}_j)^2} \quad (17)$$

Usually we are interested in the expectation value of non-trivial functions of the observables:

$$\bar{f} = \langle 0 | f(Y_1, Y_2, \dots, T_M) | 0 \rangle = f(\bar{Y}_1, \bar{Y}_2, \dots, \bar{Y}_M) \quad (18)$$

Often the  $y_{ij}$  are not normal distributed and may depend on each other therefore standard error analysis does not apply.

The proper technique for estimating the error on  $\bar{f}$  is the bootstrap algorithm. It consists of the following steps:

- We build  $K$  vectors  $b^k$  of size  $N$ . The elements of these vectors  $b_i^k$  are chosen at random, uniformly between  $\{1, 2, \dots, N\}$ .
- For every  $k$  we compute:

$$\bar{Y}_j^k = \frac{1}{N} \sum_i y_{b_i^k j} \quad (19)$$

- Again for each  $k$  we compute:

$$\bar{f}^k = f(\bar{Y}_1^k, \bar{Y}_2^k, \dots, \bar{Y}_M^k) \quad (20)$$

- We then sort the resulting values for  $\bar{f}^k$ .
- We define the  $\alpha$  percent confidence interval as  $[\bar{f}^{k'}, \bar{f}^{k''}]$  where  $k' = \lfloor (1 - \alpha)K/2 \rfloor$  and  $k'' = \lfloor 1 + \alpha)K/2 \rfloor$ .

`qcdutils_boot` is a program that takes as input  $f(Y_0, Y_1, \dots)$  in the form of a mathematical expression where the  $Y_j$  are represented by their string pattern. It locates and extracts the corresponding  $y_{ij}$  values from the log files and stores them in a file called “`qcdutils_raw.csv`”. It computes the autocorrelation for each of the  $y_{ij}$  and stores them in “`qcdutils_autocorrelation.csv`”. It computes the moving averages for each of the  $\bar{Y}_j$  and stores them in “`qcdutils_trails.csv`”. It generates the  $K$  bootstrap samples  $\bar{f}^k$  and saves them in “`qcdutils_samples.csv`”. Finally it computes the mean and the 68% confidence level intervals  $[\bar{f}^{k'}, \bar{f}^{k''}]$  and stores it “`qcdutils_results.csv`”.

Mind that these files are created in the current working directory and they are overwritten every time the `qcdutils_boot` is run. Move them somewhere else to preserve them.

Moreover, if the input expression for  $f$  depends on wildcards, the program repeats the analysis for all matching expressions.

`qcdutils_boot` performs this analysis without need to write any code. It only needs the input  $f$  in the syntax explained below and the list of log-files to analyze for data.

## 6.1 A simple example

Consider the output of one of the previous `qcdutils_run`:

```
1 python qcdutils_run.py -gauge:load=*.mdp -plaquette > run.log
```

In this case the observable is  $Y_0$ =“plaquette”. We can analyze it with

```
1 python qcdutils_boot.py 'run.log' '"plaquette"'
```

This produces the following output:

```
1 < plaquette > = min: 0.26, mean: 0.32, max: 0.38
2 average trails saved in qcdutils_trails.csv
3 bootstrap samples saved in qcdutils_samples.csv
4 results saved in qcdutils_results.csv
```

Notice that `qcdutils_run` takes three arguments:

- A file name or file pattern (for example “run.log”)
- An expression (for example “plaquette”).
- A condition (optional)

Each of the argument must be enclosed in single quotes.

The represents  $f(Y_0, Y_1, \dots)$  and the  $Y_j$  are the names of observables in double quotes.

In ’"plaquette"’ the outer single quote delimits the expression and the term plaquette between double quotes, determines the string we want to parse from the in file.

`qcdutils` uses the observable name to find all the occurrences of

```
1 plaquette = ...
```

or

```
1 plaquette: ...
```

in the input files and maps them into  $y_{i0}$  where  $i$  labels the occurrence. In this case we have a single observable (plaquette) so we use 0 to label it.

The program opens the file or the files matching the file patterns and parses them for the values of the “plaquette” thus filling an internal table of  $y$  s. It gives the output as the result:

```
1 < plaquette > = min: 0.26, mean: 0.32, max: 0.38
```

Here “mean” is the mean of the expression “plaquette”. min and max are the 65% confidence intervals computed using the bootstrap.

Here is example of the content of the “qcdutils\_results.csv” file for the average plaquette case:

```
1 "plaquette", "[min]", "[mean]", "[max]"
2 "plaquette", 0.26, 0.32, 0.38
```

In general it contains one row for each matching expression.

You can plot the content of the files generated by `qcdutils_boot` using `qcdutils_plot`:

```
1 python qcdutils_plot.py -r -a -t -b
```

Here `-r` indicates that we want to plot the raw data, `-a` indicates we want a plot of autocorrelations, `-t` is for partial averages, and `-b` means we want a plot of bootstrap samples. `qcdutils_plot` loops over all the files reads the data in them and for each  $Y_j$  it makes one plot with raw data ( $y_{ij}$ ), one with autocorrelations, one with partial averages. Then for each  $f$  it makes one plot with the bootstrap samples, and one plot with the final results found in “`qcdutils_results.csv`”.

The plots are in PNG files which have a name prefix equal to the name of the data source file, followed by a serialization of the expression for  $Y_j$  or  $f$ , depending on the case.

For example in the case of the plaquette, the autocorrelations and the partial averages are in the files:

```
1 qcdutils_autocorrelations_plaquette.png
2 qcdutils_trails_plaquette.png
```

and they are shown in fig. 13.

Similarly, if you want to bootstrap  $f(Y_0) = \exp(Y_0/3)$  where  $Y_0$  is the plaquette you would run:

```
1 python qcdutils_boot.py run.log 'exp("plaquette"/3)'
```

It produces output like this:

```
1 < exp(plaquette/3) > = min: 1.092, mean: 1.114, max: 1.145
```

Notice that again the observable  $Y_0$  is identified for convenience by “`plaquette`”. The double quotes are necessary to avoid naming conflicts between patterns and functions.

Also notice that running `qcdutils_boot` twice does not guarantee generating the same exact results twice. That is because the bootstrap samples are random.

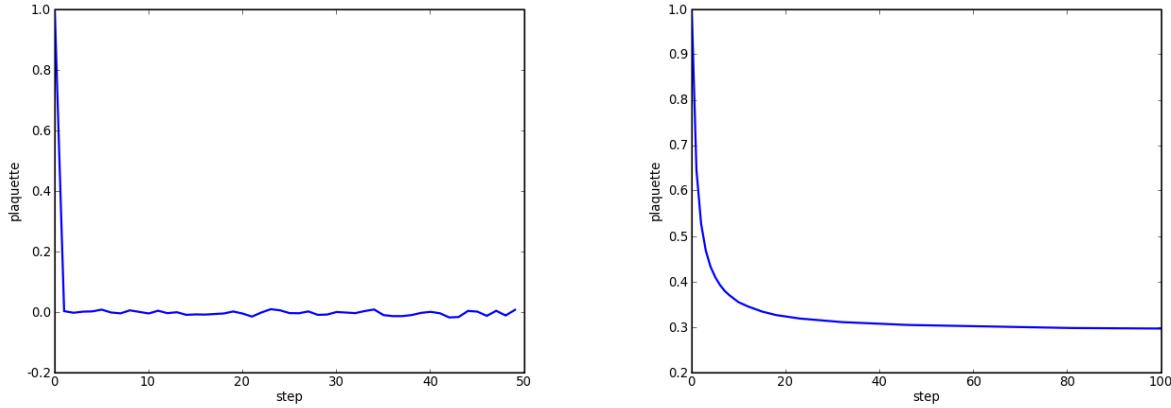


Figure 13: Example plots of autocorrelation (left) and partial averages (right).

## 6.2 2-point and 3-point correlation functions

In order to explain more complex cases we could generate 2- and 3- points correlation functions using something like:

```
1 python qcutils_run.py \
2     -gauge:start=cold:beta=4:n=10:steps=5:therm=100 \
3     -quark:kappa=0.11:c_sw=0.4:save=false -pion
4     -4quark:operator=5Ix5I > run.log
```

For testing purposes can also run:

```
1 python qcutils_boot.py -t
```

Where `-t` stands for test. This creates and analyzes a file called “`test_samples.log`” which contains *random* measurements for C2 and C3. Once this file is being created we can filter and study, for example, only the 2-point correlation function C2:

```
1 python qcutils_boot.py run.log '"C2[<t>]"'
```

Notice that `<t>` means we wish to define a variable `t` to be used internally for the analysis and whose values are to be determined by pattern-matching the data. The `t` correspond to the  $j$  of the previous abstract discussion. “`C2[<t>]`” matches `C2[0]` with  $t = 0$ , `C2[1]` matches with  $t = 1$ , etc.

The command above produces something like:

```

1 reading file test_samples.log
2 C2[00] occurs 100 times
3 ...
4 C2[15] occurs 100 times
5 raw data saved in qcdutils_raw_data.csv
6 autocorrelation for C2[02] and d=1 is -0.180453
7 ...
8 autocorrelation for C2[06] and d=1 is -0.0436378
9 autocorrelations saved in qcdutils_autocorrelations.csv
10 < C2[00] > = min: 1.988, mean: 1.999, max: 2.008
11 < C2[01] > = min: 1.617, mean: 1.629, max: 1.64
12 < C2[02] > = min: 1.328, mean: 1.345, max: 1.359
13 < C2[03] > = min: 1.064, mean: 1.079, max: 1.094
14 < C2[04] > = min: 0.878, mean: 0.894, max: 0.908
15 < C2[05] > = min: 0.722, mean: 0.733, max: 0.744
16 < C2[06] > = min: 0.574, mean: 0.584, max: 0.597
17 < C2[07] > = min: 0.478, mean: 0.49, max: 0.5
18 < C2[08] > = min: 0.395, mean: 0.407, max: 0.419
19 < C2[09] > = min: 0.322, mean: 0.331, max: 0.339
20 < C2[10] > = min: 0.268, mean: 0.277, max: 0.286
21 < C2[11] > = min: 0.225, mean: 0.231, max: 0.237
22 < C2[12] > = min: 0.18, mean: 0.186, max: 0.192
23 < C2[13] > = min: 0.138, mean: 0.144, max: 0.151
24 < C2[14] > = min: 0.107, mean: 0.112, max: 0.118
25 < C2[15] > = min: 0.0883, mean: 0.0933, max: 0.0982
26 average trails saved in qcdutils_trails.csv
27 bootstrap samples saved in qcdutils_samples.csv
28 results saved in qcdutils_results.csv

```

which we can plot as usual with

```
1 python qcdutils_plot.py -r -a -b -t
```

This produces about 60 plots. Some of them are shown in fig.14.

We can as easily compute the log of C2 (for every t):

```
1 python qcdutils_boot.py test_samples.log 'log("C2[<t>]" )'
```

or the log of the ratio between C2 at two consecutive time-slices:

```
1 python qcdutils_boot.py run2.log \
2   'log("C2[<t1>]"/"C2[<t2>]" )' \
3   't2==t1+1 if t1<8 else t2==t1-1'
```

In this case we used two implicit variables `t1` and `t2` but we used the third argument of `qcdutils_boot` to set a condition to link the two. This produces the following output:

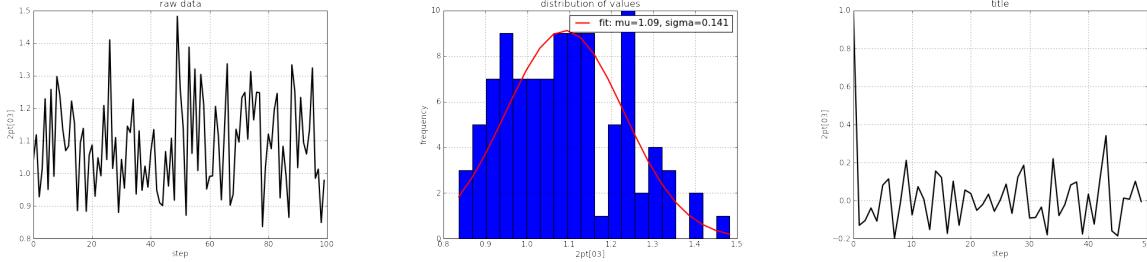


Figure 14: Example plots for the raw data (left), the distribution of raw data (center) and autocorrelations (right) for C2.

```

1 reading file test_samples.log
2 C2[00] occurs 200 times
3 ...
4 C2[15] occurs 200 times
5 raw data saved in qcutils_raw_data.csv
6 autocorrelation for C2[02] and d=1 is -0.176706
7 ...
8 autocorrelation for C2[06] and d=1 is -0.0476376
9 autocorrelations saved in qcutils_autocorrelations.csv
10 < log(C2[00]/C2[01]) > = min: 0.196, mean: 0.204, max: 0.212
11 < log(C2[01]/C2[02]) > = min: 0.18, mean: 0.191, max: 0.202
12 [...]
13 < log(C2[13]/C2[14]) > = min: 0.208, mean: 0.249, max: 0.291
14 < log(C2[14]/C2[15]) > = min: 0.147, mean: 0.189, max: 0.227
15 average trails saved in qcutils_trails.csv
16 bootstrap samples saved in qcutils_samples.csv
17 results saved in qcutils_results.csv

```

In the same fashion we can compute a matrix element as the ratio between a 3-point correlation function (C3) and a 2-point correlation function (C2):

```

1 python qcutils_boot.py test_samples.log \
2   ' "C3[<t>][<t1>]"/"C2[<t2>]"/"C2[<t3>]"' \
3   't3==t and t2==t and t1==t' > run.log
4 python qcutils_plot.py -a -t -b -r

```

Some of the generated plots can be seen in fig.19-16.

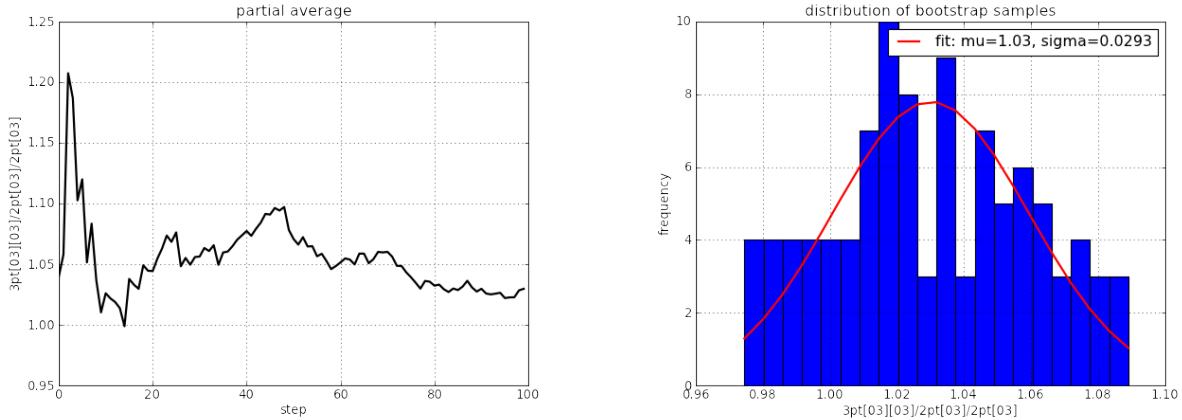


Figure 15: Example plots of moving averages (left) and distribution of bootstrap samples (right) for the ratio  $C_3/C_2^2$ .

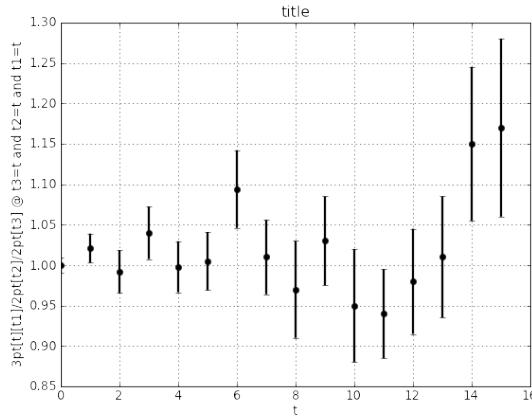


Figure 16: Example plot showing results of the bootstrap analysis.

### 6.3 Fitting data with `qcdutils_fit.py`

`qcdutils_fit.py` is a fitting and extrapolation utility. It can read and understand the output of `qcdutils_boot.py`. Internally it uses a “stabilized” multidimensional Newton method to minimize  $\chi^2$ . It is stabilized by reverting to the steepest descent in case the Newton step fails to reduce the  $\chi^2$ . The length of the steepest descent step is adjusted dynamically to guarantee that each step of the algorithm reduces the  $\chi^2$ . The program accepts for input any

function and any number of the parameters. It also accepts, optionally, Bayesian priors for those parameters and they can be used to further stabilize the fit [17]. A more sophisticated approach is described in ref. [18].

In Euclidean space C2 can be modeled by an exponential  $a \exp(-bt)$  and  $b$  is the mass of the lowest energy state which propagates between the source and the sink. Here is an example in which we fit C2 using a single exponential:

```

1 python qcdutils_boot.py -t
2 python qcdutils_boot.py test_samples.log '"C2[<t>]"' > run.log
3 python qcdutils_fit.py 'a*exp(-b*t)@a=2, b=0.3'
```

The input data is read from the output of `qcdutils_boot`. The expression in quotes is the fitting formula. You can name the fitting parameters as you wish (in this case `a` and `b`) but the other parameters (in this case `t`) must match the parameters defined in the argument of `qcdutils_boot` (`<t>`). The `@` symbol separates the fitting function (left) from the initial estimates for the fitting parameters (on the right, separated by commas). Every parameter to be determined by the fit must have an initial value.

The output looks something like this:

```

1 a = 1.99864
2 b = 0.200645
3 chi2= 12.8048378376
4 chi2/dof= 0.984987525973
```

`qcdutils_fit.py` also generates the plot of fig. 17 (left).

If C2 is a meson propagator,  $b$  here represents the mass of the meson (of the lowest energy state with the same quantum numbers as the operator used to create the meson).

Similary we can analyze and fit the log of C2:

```

1 python qcdutils_boot.py test_samples.log 'log("C2[<t>]"' > run.log
2 python qcdutils_fit.py 'a-b*t@a=1, b=0.3'
```

which produces something like:

```

1 a = 0.69169
2 b = 0.200627
3 chi2= 12.1641201448
4 chi2/dof= 0.935701549598
```

and the plot of fig. 17 (right)

If our goal is obtaining  $b$  we can also cancel the  $a$  dependency in the analysis:

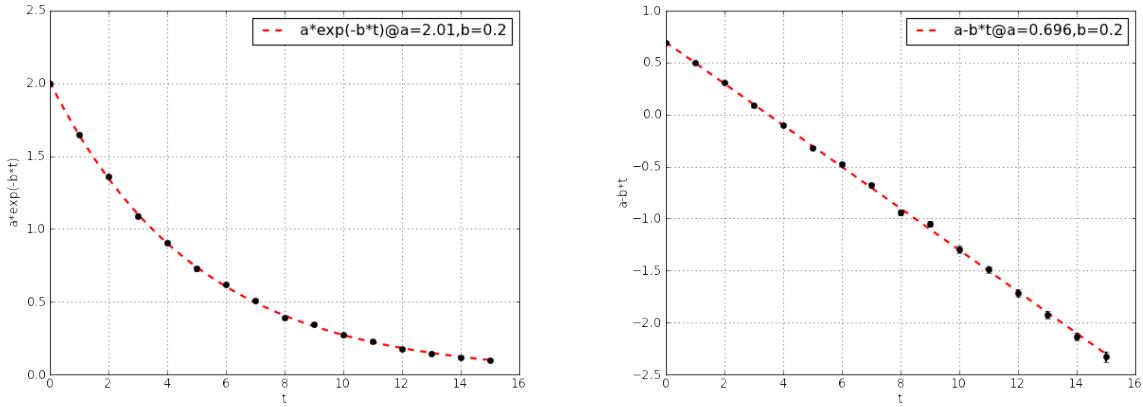


Figure 17: Example fits for a two points correlation function (left) and its log (right).

```

1 python qcdutils_boot.py test_samples.log \
2   'log("C2[<t>]"/"C2[<t1>]"' 't1==t+1' > run.log
3 python qcdutils_fit.py 'b@b=0'
```

and obtain:

```

1 b = 0.201755
2 chi2= 12.4502446913
3 chi2/dof= 0.9577111301
```

The generated plot is shown in fig. 18.

Notice that the variable names **a** and **b** are arbitrary and you can choose any name.

Similarly we can fit 3-point correlation functions:

```

1 python qcdutils_boot.py test_samples.log '"C3[<t1>][<t2>]"' > run.log
2 python qcdutils_fit.py 'a*exp(-b*(t1+t2))@a=3, b=0.3, _b=0.2'
```

In this case we have stabilized the plot with a Bayesian prior, indicated by **\_b**. A variable starting with underscore indicates the uncertainty associated with our a priori knowledge about the corresponding variable without underscore. In other words **b=0.3, \_b=0.2** is equivalent to **b=0.3 ± 0.2**. The result of this fit yields something like:

```

1 a = 3.78387
2 b = 0.195542
3 chi2= 2070.73759118
4 chi2/dof= 8.18473356199
```

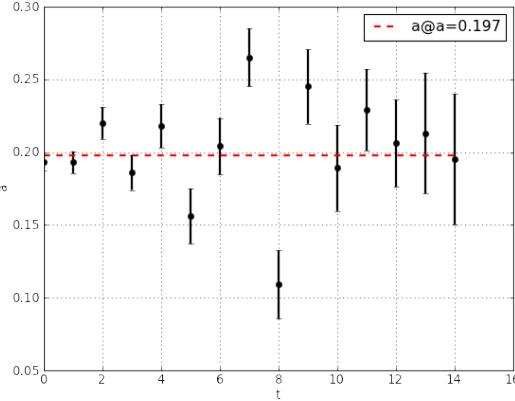


Figure 18: Example plot of fit of  $\log(C2(t)/C2(t - 1))$ .

A call to `qcdutils_plot.py` generates the plot of fig.19 (left)

In order to extract a matrix element (for example a 4-quark operator) we fit the ratio between C3 and C2:

```

1 python qcdutils_boot.py test_samples.log \
2   '"C3[<t>][<t1>]"// "C2[<t2>]"/"C2[<t3>]"' \
3   't3==t and t2==t and t1==t' > run.log
4 python qcdutils_fit.py 'a@a=0'

```

It produces output like:

```

1 a = 1.00658
2 chi2= 13.2075581022
3 chi2/dof= 0.943397007297

```

It produces the plot in fig. 19 (right).

You can use `qcdutils_fit.py` to perform extrapolations by using the `-extrapolate` command line option:

```

1 python qcdutils_fit.py -extrapolate x=100 'ax+b@a=1, b=0'

```

The extrapolated point will be added to the generated plot and represented by a square.

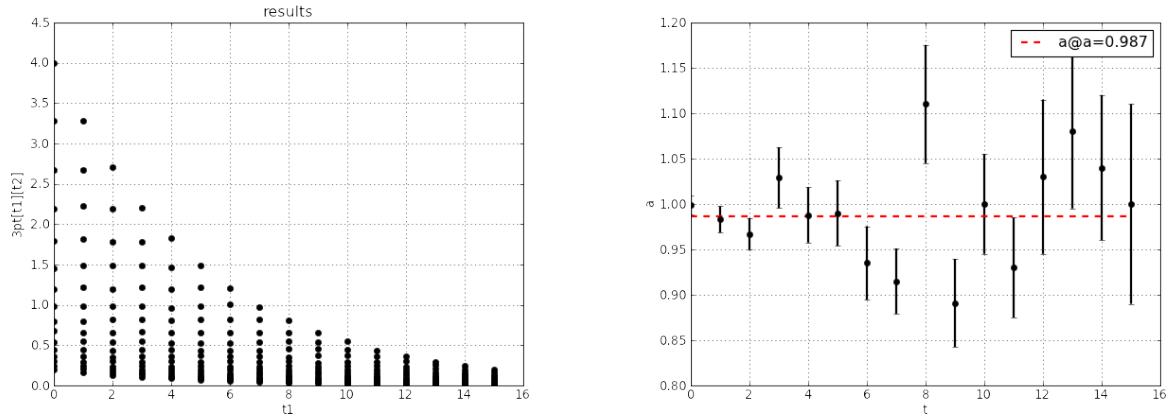


Figure 19: Example plot showing  $C3[t][t1]$  (left) and the fit of  $C3[t]/C2[t]c2[t1]$  (right).

## 6.4 Dimensional analysis and error propagation

In this section we did not discuss error propagation but we have developed a utility called *Buckingham* which is available from:

<http://code.google.com/p/buckingham/>

It provides dimensional analysis, unit conversion, and arithmetic operation with error propagation. We plan to discuss it in a separate manual but we here provide one example of usage (from inside a Python shell):

```

1 >>> from buckingham import *
2 >>> a = Number(2.0, error=0.3, dims="fermi")
3 >>> b = Number(1.0, error=0.2, dims="second^2")
4 >>> c = a/b
5 >>> print c, c.units()
6 (2.000 pm 0.500)/10^15 meter*second^-2
7 >>> print c.convert('fermi*second^-2')
8 2.000 pm 0.500
9 >>> print c.convert('lightyear*day^-2')
10 (1.578 pm 0.395)/10^21

```

(here  $\text{pm}$  stands for  $\pm$ ) Buckingham supports 944 unit types (including eV) and their combinations.

## A Filename conventions

```
1 Gauge configuration in NERSC format (3x3 or 3x2)
2   *.nersc
3 Gauge configuration in Fermiqcd format
4   *.mdp
5 Gauge configuration in MILC format
6   *.milc
7 Generic LIME file
8   *.lime
9 Gauge configuration in ILDG format
10  *.ildg
11 SciDAC quark propagator
12  *.scidac
13 Quark propagator in FermiQCD format
14  *.prop.mdp
15 Time slice for gauge configuration in FermiQCD format:
16  *.t[NNNN].mdp
17 Time slice for propagator in FermiQCD format:
18  *.t[NNNN].prop.mdp
19 Quark field for a given SPIN, COLOR source:
20  *.s[SPIN].c[COLOR].quark
21 Generic log file
22  *.log
23 VTK file containing real trace of plaquettes
24  *.plaqette.vtk
25 VTK file containing real part of Polyakov lines
26  *.polyakov.vtk
27 VTK file containing topological charge density
28  *.topcharge.vtk
29 VTK file containing topological charge density for a cooled config
30  *.topcharge.cool[STEP].vtk
31 VTK file contain a the norm squared of a pion propagator
32  *.pion.vtk
33 HTML file generated by qcdutils_vtk, represents a VTK file.
34  *.vtk.html
35 VisIt visualization script generated by qcdutils_vis.py
36  qcdutils_vis_[UUID].py
37 VisIt image generates by the previous script
38  qcdutils_vis_[UUID]_[FRAME].jpeg
39 Raw data extract from a log file by qcdutils_boot
40  qcdutils_raw_data.csv
41 Autocorrelations computed by qcdutils_boot
42  qcdutils_autocorrelations.csv
43 Partial averages computed by qcdutils_boot
```

```

44     qcdutils_trails.csv
45 Bootstrap samples generated by qcdutils_boot
46     qcdutils_samples.csv
47 Means and bootstrap errors computed by qcdutils_boot
48     qcdutils_results.csv

```

## B Help Pages

### B.1 qcdutils\_get.py

```

1 $ qcdutils_get.py -h
2 Usage:
3
4     qcdutils_get.py [options] sources
5
6 Examples:
7
8     qcdutils_get.py --test
9     qcdutils_get.py --convert ildg gauge.cold.12x8x8x8
10    qcdutils_get.py --convert mdp --float *.ildg
11    qcdutils_get.py --convert split.mdp *.mdp
12
13 Options:
14     -h, --help                  show this help message and exit
15     -q, --quiet                 no progress bars
16     -d DESTINATION, --destination=DESTINATION
17                           destination folder
18     -c CONVERT, --convert=CONVERT
19                           converts a field to format
20                           (ildg,split.prop.mdp,prop.ildg,prop.mdp,split.mdp,
21                           mdp)
22     -4, --float                 converts to float precision
23     -8, --double                converts to double precision
24     -t, --tests                 runs some tests
25     -n, --noprogressbar        disable progress bar

```

### B.2 qcdutils\_run.py

```

1 $ qcdutils_run.py -h
2 qcdutils_run.py is a tool to help you download and use fermiqcd from
3
4     http://code.google.com/p/fermiqcd
5

```

```

6 When you run:
7
8     python qcdutils_run.py [args]
9
10 It will:
11 - create a folder called fermiqcd/ in the current working directory
12 - connect to google code and download fermiqcd.cpp + required libraries
13 - if -mpi in [args] compile fermiqcd with mpiCC else with g++
14 - if -mpi in [args] run fermiqcd.exe with mpiCC else run it normally
15 - pass the [args] to the compiled fermiqcd.exe
16
17 Some [args] are handled by qcdutils_run.py:
18 -download force downloading of the libraries
19 -compile force recompiling of code
20 -source runs and compiles a different source file
21 -mpi for use with mpi (mpiCC and mpirun but be installed)
22
23 Other [args] are handled by fermiqcd.cpp for example
24 -cold make a cold gauge configuration
25 -load load a gauge configuration
26 -quark make a quark
27 -pion make a pion
28 (run it with no options for a longer list of options)
29
30 You can find the source code in fermiqcd/fermiqcd.cpp
31
32 More examples:
33     qcdutils_run.py -gauge:start=cold:nt=16:nx=4
34     qcdutils_run.py -gauge:start=hot:nt=16:nx=4
35     qcdutils_run.py -gauge:load=cold.mdp
36     qcdutils_run.py -gauge:load=cold.mdp:steps=10:beta=5.7
37     qcdutils_run.py -gauge:load=*.mdp -plaquette
38     qcdutils_run.py -gauge:load=*.mdp -plaquette_vtk
39     qcdutils_run.py -gauge:load=*.mdp -polyakov_vtk
40     qcdutils_run.py -gauge:load=*.mdp -cool:steps=20 -topcharge_vtk
41     qcdutils_run.py -gauge:load=*.mdp -quark:kappa=0.12:alg=minres_vtk
42     qcdutils_run.py -gauge:load=*.mdp -quark:kappa=0.12 -pion
43     qcdutils_run.py -gauge:load=*.mdp -quark:kappa=0.12 -pion_vtk
44
45 Options:
46     -cool
47         alg = ape
48         alpha = 0.7
49         steps = 20
50         cooling = 10

```

```

51      -cool_vtk
52          n = 20
53          alpha = 0.7
54          steps = 1
55          cooling = 10
56
57      -quark
58          action = clover_fast (default) or clover_slow or clover_sse2
59          alg = bicgstab (default) or minres or bicgstab_vtk or minres_vtk
60          abs_precision = 1e-12
61          rel_precision = 1e-12
62          source_t = 0
63          source_x = 0
64          source_y = 0
65          source_z = 0
66          source_point = zero (default) or center
67          load = false (default) or true
68          save = true (default) or false
69          matrices = FERMILAB (default) or MILC or
70                  UKQCD or Minkowsy-Dirac or Minkowsy-Chiral
71          kappa = 0.12
72          kappa_t = quark[ "kappa" ]
73          kappa_s = quark[ "kappa" ]
74          r_t = 1.0
75          r_s = 1.0
76          c_sw = 0.0
77          c_E = 0.0
78          c_B = 0.0
79
80      -meson
81          source = 5
82          sink = 5
83          current = I
84
85      -4quark
86          source = 5 (default) or I or 0 or 1 or 2 or 3 or 05 or
87                  15 or 25 or 35 or 01 or 02 or 03 or 12 or 13 or 23
88          operator = 5Ix5I (default) or 0Ix0I or 1Ix1I or 2Ix2I or 3Ix3I or
89                  05Ix05I or 15Ix15I or 25Ix25I or 35Ix35I or 01Ix01I or
90                  02Ix02I or 03Ix03I or 12Ix12I or 13Ix13I or 23Ix23I or
91                  5Tx5T or 0Tx0T or 1Tx1T or 2Tx2T or 3Tx3T or 05Tx05T or
92                  15Tx15T or 25Tx25T or 35Tx35T or 01Tx01T or 02Tx02T or
93                  03Tx03T or 12Tx12T or 13Tx13T or 23Tx23T
94
95      -gauge
96          nt = 16
97          nx = 4
98          ny = nx
99          nz = ny

```

```

96     start = load (default) or cold or hot or instantons
97     load = demo.mdp
98     n = 0
99     steps = 1
100    therm = 10
101    beta = 0
102    zeta = 1.0
103    u_t = 1.0
104    u_s = 1.0
105    prefix =
106    action = wilson (default) or wilson_improved or wilson_sse2
107    save = true
108    t0 = 0
109    x0 = 0
110    y0 = 0
111    z0 = 0
112    r0 = 1.0
113    t1 = 1
114    x1 = 1
115    y1 = 1
116    z1 = 1
117    r1 = 0.0
118 -baryon
119 -pion
120 -pion_vtk
121 -meson_vtk
122 -current_static
123 -current_static_vtk
124 -plaquette
125 -plaquette_vtk
126 -polyakov_vtk
127 -topcharge_vtk

```

### B.3 qcutils\_vis.py

```

1 $ qcutils_vis.py -h
2 Usage:
3 This is a utility script to manipulate vtk files containing scalar files.
4 Files can be split, interpolated, and converted to jpeg images.
5 The conversion to jpeg is done by dynamically generating a visit script
6 that reads the files, and computes optimal contour plots.
7
8 Examples:
9
10 1) make a dummy vtk file

```

```

11      qcdutils_vis.py -m 10 folder/test.vtk
12
13 2) reads fields from multiple vtk files
14      qcdutils_vis.py -r field folder/*.vtk
15
16 3) extract fields as multiple files
17      qcdutils_vis.py -s field folder/*.vtk
18
19 (fields in files will be renamed as "slice")
20 4) interpolate vtk files
21
22      qcdutils_vis.py -i 9 folder/*.vtk
23
24 tricubic Resample/Interpolate individual vtk files
25
26 visit -v 10x10x10 folder/*.vtk
27
28 6) render a vtk file as a jpeg image
29
30      qcdutils_vis.py -p 'AnnotationAttributes[axes3D.bboxFlag=0];
31          ResampleAttributes[samplesX=160;samplesY=160;samplesZ=160];
32          ContourAttributes[SetMultiColor(9,$orange)]' 'folder/*.vtk'
33
34 or simply
35
36      qcdutils_vis.py -p default 'folder/*.vtk'
37
38 Options:
39
40      -h, --help           show this help message and exit
41      -r READ, --read=READ name of the field to read from the vtk file
42      -s SPLIT, --split=SPLIT
43                  name of the field to split from the vtk file
44      -i INTERPOLATE, --interpolate=INTERPOLATE
45                  name of the vtk files to add/interpolate
46      -c CUBIC, --cubic-interpolate=CUBIC
47                  new size for the lattice 10x10x10
48      -m MAKE, --make=MAKE make a dummy vtk file with size^3 whete size if
49      arg of
50                  make
51      -p PIPELINE, --pipeline=PIPELINE
52                  visualizaiton pipeline instructions

```

## B.4 qcdutils\_vtk.py

```
1 $ qcdutils_vtk.py -h
2 Usage: qcdutils_vtk.py filename.vtk
3
4 Options:
5   -h, --help           show this help message and exit
6   -u UPPER, --upper-threshold=UPPER
7               threshold for isosurface
8   -l LOWER, --lower-threshold=LOWER
9               threshold for isosurface
10  -R UPPER_RED, --upper-red=UPPER_RED
11      color component for upper isosurface
12  -G UPPER_GREEN, --upper-green=UPPER_GREEN
13      color component for upper isosurface
14  -B UPPER_BLUE, --upper-blue=UPPER_BLUE
15      color component for upper isosurface
16  -r LOWER_RED, --lower-red=LOWER_RED
17      color component for lower isosurface
18  -g LOWER_GREEN, --lower-green=LOWER_GREEN
19      color component for lower isosurface
20  -b LOWER_BLUE, --lower-blue=LOWER_BLUE
21      color component for lower isosurface
```

## B.5 qcdutils\_boot.py

```
1 $ qcdutils_boot.py -h
2 Usage: qcdutils_boot.py *.log 'x[<a>]/y[<b>]' 'abs(a-b)==1'
3   scans all files *.log for expressions of the form
4     x[<a>]=<value> and y[<b>]=<value>
5   and computes the average and bootstrap errors of x[<a>]/y[<b>]
6   where <a> and <b> satisfy the condition abs(a-b)==1.
7
8 This is program to scan the log files of a Markov Chain Monte Carlo
9 Algorithm,
10 parse for expressions and compute the average and bootstrap errors of any
11 function of those expressions. It also compute the convergence trails of
12 the
13 averages.
14
15 Options:
16   --version           show program's version number and exit
17   -h, --help           show this help message and exit
18   -b MIN, --minimum_index=MIN
```

```

17                         the first occurrence of expression to be
18                         considered
19 -e MAX, --maxmimum_index=MAX
20                         the last occurrence +1 of expression to be
21                         considered
22 -n NSAMPLES, --number_of_samples=NSAMPLES
23                         number of required bootstrap samples
24 -p PERCENT, --percentage=PERCENT
25                         percentage in the lower and upper tails
26 -t, --test
27                         make a test!
28 -r, --raw
29                         Load raw data instead of parsing input
30 -a, --advanced
31                         In advanced mode use regular expressions for
                           variable
                           patterns
-i IMPORT_MODULE, --import_module=IMPORT_MODULE
                           import a python module for expression evaluation
-o OUTPUT_PREFIX, --output_prefix=OUTPUT_PREFIX
                           path+prefix used to build output files

```

## B.6 qcdutils\_plot.py

```

1 $ qcdutils_plot.py -h
2 Usage: python qcdutils_plot.py
3
4 plot the output of qcdutils.py
5
6 Options:
7   --version
8   -h, --help
9   -i INPUT_PREFIX, --input_prefix=INPUT_PREFIX
10                          the prefix used to build input filenames
11   -r, --raw
12   -a, --autocorrelations
13                          make autocorrelation plots
14   -t, --trails
15   -b, --bootstrap-samples
16                          make bootstrap samples plots
17   -v PLOT_VARIABLES, --plot_variables=PLOT_VARIABLES
18                          plotting variables
19   -R RANGE, --range=RANGE
20                          range as in 0:1000

```

## B.7 qcdutils\_fit.py

```
1 $ qcdutils_fit.py -h
```

```

2 Usage: qcdutils_fit.py [OPTIONS] 'expression@values'
3   Example: qcdutils-fit.py 'a*x+b@a=3, b=0'
4   default filename is qcdutils_results.csv
5   ...., 'x', 'min', 'mean', 'max'
6   ...., 23, 10, 11, 12
7   ...., etc etc etc
8
9 Options:
10  --version           show program's version number and exit
11  -h, --help            show this help message and exit
12  -i INPUT, --input=INPUT
13                      input file (default qcdutils_results.csv)
14  -c CONDITION, --condition=CONDITION
15                      sets a filter on the points to be fitted
16  -p PLOT, --plot=PLOT plots the hessian (not implemented yet)
17  -t, --test             test a fit
18  -e EXTRAPOLATIONS, --extrapolate=EXTRAPOLATIONS
19                      extrapolation point
20  -a AP, --absolute_precision=AP
21                      absolute precision
22  -r RP, --relative_precision=RP
23                      relative precision
24  -n NS, --number_steps=NS
25                      number of steps

```

## References

- [1] M. Di Pierro, J. Hetrick, S. Cholia, D. Skinner, PoS LAT2011, 2011
- [2] <http://www.usqcd.org/ildg/>
- [3] M. DiPierro, Comput.Phys.Commun. 141, 2001, (pp 98-148) [hep-lat/0004007]
- [4] M. Di Pierro, Nucl.Phys.Proc.Suppl.129:832-834, 2004 [<http://fermiqcd.net>]
- [5] <http://www.vtk.org/>
- [6] <https://wci.llnl.gov/codes/visit/home.html>
- [7] <http://processingjs.org/>
- [8] <http://python.org>
- [9] <http://www.physics.utah.edu/~detar/milc>

- [10] M. Creutz, *Quarks, gluons and lattices*, Cambridge University Press, 1985
- [11] I. Montvay and G. Münster, *Quantum fields on a lattice* Cambridge University Press, 1997
- [12] T. DeGrand and C. DeTar, Lattice Methods for Quantum Chromodynamics, World Scientific 2006
- [13] K. Wilson, K, *Confinement of quark*, Physical Review D 10 (8) 1974
- [14] C. Morningstar and M. Peardon, Phys.Rev.D56:4043-4061,1997
- [15] E. Eichten and B. Hill, Phys. Lett. B234 (1990) 51
- [16] B. Efron, The Annals of Statistics 7 (1), 1979
- [17] G. Lepage et al., Nucl.Phys.Proc.Suppl.106:12-20,2002 [arXiv:hep-lat/0110175v1]
- [18] M. Di Pierro, <http://arxiv.org/abs/1202.0988v2>

# Y2013 MS in Computational Finance Advising Guide

The objective of this program is to offer students the opportunity to acquire both the ability to understand existing financial models in a quantitative and mathematical way, and the ability to implement these models in the form of computer programs. This program differs from a regular MS in Finance because of a stronger mathematical component and the addition of an intensive computational component. The program aims to produce graduates with the required qualifications to become "quantitative financial analysts". The Computational Finance graduates will be able to apply these quantitative tools to solve complex problems in the areas of portfolio management, risk management, and financial engineering.

The Master of Science in Computational Finance is a joint degree between the College of Computing and Digital Media (CDM) and the Kellstadt Graduate School of Business (KGSB). Student may be admitted to the program through either CDM or KGSB.

## **IMPORTANT ADMISSION ISSUES**

Students to be admitted need a GRE or GMAT score and need a semester calculus.

Normally we admit students with Calculus background and a GRE quantitative score in the top 20% OR students with a science and engineering background and an A grade in their math courses (calculus, linear algebra, differential equations).

Students without a GRE or GMAT test cannot be admitted (Commerce requirement)

Before the 2013 degree there was a pre-requisite phase. It included calculus  
Since the 2013 the degree has a variable length and include an Introductory phase which does not include Calculus. Knowledge of calculus is still a PREREQUISITE FOR ADMISSION in the degree. The new CSC412 includes a review of Calculus but it is not a replacement for proper calculus courses.

The CDM degree and the Commerce degree are the same but Commerce requires the GMAT not the GRE or GMAT. Students with a previous MS in Finance or an MBA may be discounted 3 Commerce courses if they enroll via Commerce. Students will get an advisor in the College where they enroll and can only apply for assistantships in that College.

## Courses (variable length 13-17 courses)

<b>Introductory Phase</b>		<b>(courses can be waived)</b>
CSC401	Introduction to Programming	
CSC404	Advanced C++	
CSC412	Tools and Techniques for Computational Analysis	
IT403	Statistics and Data Analysis	
<b>CDM Foundation Phase</b>		
CSC 423	Data Analysis and Regression	(usually offered every quarter)
CSC 425	Time Series Analysis and Forecasting	(usually offered in Winter only)
CSC 431	Scientific Computing	(usually offered in Winter only)
CSC 521	Monte Carlo Algorithms	(usually offered in Spring only)
<b>Kellstadt Foundation Phase</b>		
ACC 500	Financial Accounting	
ECO 555	Economics for Decision-Making	
FIN 555	Financial Management	
FIN 523	Investment Analysis	
FIN 525	Portfolio Management	
FIN 562	Risk Management	
FIN 662	Derivatives Valuation	
<b>Advanced Courses</b>		
CSC 695	Master's Independent Study	
or CSC 697	Graduate Internship	
or CSC 559	Software Engineering for Financial Markets	(usually offered in Spring only)
<b>Major Elective Courses</b>		
Students must take 1 500-level course at CDM, Kellstadt, or the Department of Math.		
<b>Recommended Electives</b>		
CSC 452	Database Programming	
CSC 453	Database Technologies	
CSC 457	Expert Systems	
CSC 458	Symbolic Programming	
CSC 478	Programming Data Mining Applications	
CSC 583	Artificial Intelligence II	
CSC 480	Artificial Intelligence I	
CSC 503	Parallel Algorithms	
CSC 529	Advanced Data Mining	
CSC 555	Mining Big Data	
CSC 582	Machine Learning	
CSC 598	Topics in Data Analysis	
CSC 672	Data Analysis Workshop	

## Old Prerequisite Mapping

2005	2006-2009	2012	2013
MAT150/151 or MAT 160/161 or MAT 170/171	MAT150/151 or MAT 160/161 or MAT 170/171	MAT150/151 or MAT 160/161 or MAT 170/171	Students MUST know calculus prior to being admitted!
n/a	n/a	n/a	CSC412
CSC309	CSC309 or CSC261 and CSC262	CSC243 and CSC309	CSC401 and CSC404
CSC393	CSC202 and CSC321	CSC202	n/a
n/a	n/a	n/a	IT403

## Waving rules

CSC412 can be waived to student with a semester of calculus (covering limits, derivatives, integrals, Taylor series, study of functions) and one quarter of linear algebra. ATTENTION: Students who do not have a semester calculus should NOT be admitted in the program!

CSC401 can be waived to students with previous programming experience in any language. But they should be recommended to take the course since it uses Python and the language will be used throughout the program.

CSC404 can be waived to students with a semester of C++ (B-)

IT403 can be waived to students who have completed a course on basics statistics (B-)

## Other Substitution Rules

There is very little flexibility for substitutions in this program but...

Students with a Commerce degree in Finance or Accounting can substitute:

- FIN524 for ACC500
- FIN557 or IB530 for ECO555.

If there are scheduling reasons the following substitutions are also possible:

- FIN617 for CSC431
- MAT 459 for CSC521
- CSC453 in place of a 500 level elective

**DePaul**  
**ENGAGE**

Thursday, May 29<sup>th</sup>, 2014

Massimo DiPierro  
School of Computing  
College of Computing and Digital Media

Dear Professor DiPierro,

I am writing to congratulate you on being one of 97 DePaul professors who were nominated this academic year for having made a significant, positive impact on the lives of DePaul students in their development as socially responsible leaders. We are pleased to be able to recognize your contribution to the DePaul community at the 2014 DePaul ENGAGE Faculty Recognition Breakfast, to be held on Thursday, May 29<sup>th</sup>, 2014 from 9:00-10:30 am in Cortelyou Commons on the Lincoln Park Campus.

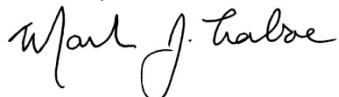
DePaul ENGAGE is a coalition of university departments, faculty, staff and student leaders interested in further strengthening and integrating DePaul's efforts to "develop socially responsible future leaders" as an essential part of our Catholic, Vincentian, and urban mission. The work of this group dates back to the early days of the *Vision Twenty12* strategic plan in 2008, which placed an emphasis on "preparing students to be socially responsible leaders and engaged alumni." Since that time, DePaul ENGAGE has worked collaboratively to coordinate among and across departments, and to reach, inform, and engage a greater number of students through our various curricular and co-curricular programs and opportunities which encourage and promote socially responsible leadership.

Each year since 2011, through an online process, DePaul ENGAGE has invited DePaul students to submit a nomination for a professor whose teaching and mentorship made a significant contribution to their learning and development as socially responsible and engaged leaders in the Vincentian tradition. The goal is to take the time to recognize the good work of faculty and to acknowledge their important role in shaping the lives of students through their teaching and mentorship. Thank you for your outstanding work!

We hope to continue to keep you "engaged" in the coming year. Perhaps you can assist us by sharing with us (via email: [ekraus@depaul.edu](mailto:ekraus@depaul.edu), via Facebook: DePaulEngage, or via Twitter: @DePaul\_Engage ) any news of events and opportunities through your department or in the city in which students may be interested in or benefit from knowing, and which may help them to learn and grow as socially responsible leaders? Or, perhaps you could get in the habit of sharing the news about events or opportunities in other areas of campus with students in your classes and department? We also look forward to being able to spread the news with students about your good work as a professor who supports their growth and learning in an outstanding way!

Thank you again for your service to DePaul and its mission through your teaching and mentorship of DePaul students. You are making a positive impact!

Sincerely,



Mark J. Laboe  
Associate VP, Student Affairs/University Ministry  
Convener, DePaul ENGAGE

***DePaul departments currently represented in DePaul ENGAGE:***

Campus Recreation	New Student and Family Programs	School of Law, Pro Bono and
Center for Intercultural Programs	/ Orientation	Community Service Initiative
Center for Interreligious Engagement	Office of Diversity Education	Office of Student Involvement
Egan Urban Center	Peace Studies and Conflict Resolution Program	Student Leadership Institute
Humanities Center	Political Science Dept.	UMIN – DCSA/Vincentian Community Service/VIA
International Studies Department	Religious Studies Dept.	University Internship Program (UIP)
Mission and Values	Stearns Center for CBSL	Vincent on Leadership: The Hay Project
Office of Multicultural Student Success (OMSS)	School for Public Service	



## Distinctions

[About](#)   [Featured Stories](#)   [Words & Deeds](#)   [Quick Hits](#)   [Data Points](#)   [Faculty Profiles](#)

[DePaul University Distinctions](#) > [Featured Stories](#) > Technology of the Year

### “Technology of the Year”

9/27/2013

[Twitter](#)   [Facebook](#)   [Google Plus](#)



It's good to be the best.

In 2012, Web2py was named "Technology of the Year" by InfoWorld which, as the leading source of information on emerging enterprise technologies, picks each year's best hardware, software, development tools, and cloud services. An open-source framework for Web-based applications, Web2py was built by Massimo Di Pierro, an associate professor in the College of Computing and Digital Media.

According to the judges, Web2py “installs everything you need—even a Python interpreter—for building a Python-based Web application. [Python is a general purpose programming language]. Its creator's mission to build an easy-to-use framework extends throughout ... With all its built-in assistance, Web2py is as painless as it gets.”

The year before, Web2py won InfoWorld's Bossie Award for “best open source application development software.” The judges then were just as enthusiastic:

“Web2py is an intelligently designed, well-crafted framework that boasts a small footprint, an uncluttered API [application programming interface], excellent documentation, and a soup-to-nuts Web-based administration tool that also serves as a complete integrated development environment. Installation is easy, wizards help speed the creation of new applications, and complexity is abstracted away. Web2py is a powerfully capable framework with plenty of pleasant surprises under the hood—a standout among the pillars of Python.”

The story of Web2py began in the classroom.

“A few years ago, I was asked to teach a course on building applications that would run on the Internet,” Di Pierro recalls. “Since showing is better than telling, I built the software to demonstrate the ‘ins and outs’ of the process to students. As students graduated and went out into the world, they took Web2py with them. Because it's

### Recent Articles

[From Idea to Action: Careers in Health Care](#)

[From Advocacy to Empowerment](#)

[Ready, Set, Go](#)

[Just like the Pros](#)

[Social Justice in Brick and Mortar](#)

an open-source application, the software began to attract a community of developers.”

Over time, hundreds of people have contributed to Web2py, and Di Pierro is quick to credit collaborators. “Web2py is available for free for everyone to use, modify, and enhance,” he says. “People can—and do—improve the application all the time, making it better without them having to worry about intellectual property rights.”

To maintain the product’s integrity, Di Pierro trademarked the name.

“Web2py is distributed under the open source LGPL license, which allows others to modify it, build upon it, and even sell derivative products under certain conditions,” he says. “I review and approve all changes to the ‘official’ version. For me, keeping the product up-to-date is like running a company, except no one gets paid.”

Each month, more than 1,800 messages are posted on the collaborative Web2py site. Since developing the software in 2007, Di Pierro has posted 20,154 messages, most of them to answer questions. Web2py counts more than 6000 registered users (both people and companies) and an unknown number of unregistered users.

Here are just a few examples of how the framework has been used commercially:

- Websites with applications and databases which can be accessed remotely. For example, a dynamic real estate website can include tours of properties.
- Industrial robots that can be communicated with and controlled remotely.
- Alarm systems that can be turned on or off remotely.
- CrowdGrader, a new online “crowdsourcing” tool which allows students to grade their classmates’ homework and receive credit for the effort they put in.
- Cisco’s new penetration testing tool which determines whether a network is vulnerable to attacks.
- Disaster response and management systems. For example, Sahana Eden is a humanitarian platform that had been used to coordinate efforts during earthquakes in Haiti and Japan; flooding in Colombia, Venezuela, and Pakistan; wildfires in Chile; and food distribution by the United Nations.

In the preface to the “Complete Reference Manual” for Web2py, Di Pierro puts the product in a Vincentian context: “I believe that the ability to easily build high quality web applications is of critical importance for the growth of a free and open society. This prevents the biggest players from monopolizing the flow of information.”

Regularly updated by Di Pierro, the manual is in its fifth edition. An online version is free and available at <http://web2py.com/book>. Readers can edit the manual online, and then submit their changes to Di Pierro for possible inclusion in a future edition.

For a list of people who have contributed to Web2py, go to  
<http://web2py.com/init/default/who>

For a selection of web sites created on Web2py, go to <http://web2py.com/poweredyby>

#### *Distinctions*

1 E. Jackson  
Chicago, IL 60604





# My Videos

99 Videos

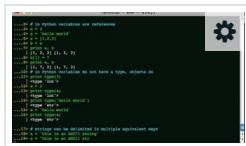
0 Appearances

99 Total

Sort: Date / Alphabetical / Plays / Likes / Comments / Duration



25:00

**Python 101**

from mdipierro / Added 3 years ago

4,988 Plays / 67 Likes / 5 Comments

Learn to program in python in less than 30 minutes.

[+ More details](#)**web2py 1.59 tutorial + auth + layouts + gae + shell**

from mdipierro / Added 5 years ago

9,266 Plays / 15 Likes / 3 Comments

This shows how to use the new web2py tools, make custom layouts and use the Google App Engine.

[+ More details](#)**DePaul IPD359 Week 1 - Web development with Python and web2py**

from mdipierro / Added 1 year ago

2,669 Plays / 13 Likes / 2 Comments

[+ More details](#)**What is going on with web2py?**

from mdipierro / Added 4 years ago

11.7K Plays / 11 Likes / 2 Comments

All of this can run on GAE if you want without changes in the code.

[+ More details](#)**DePaul IPD359 Week 2 - Web development with Python and web2py**

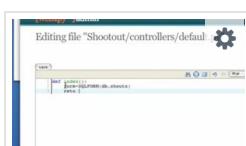
from mdipierro / Added 1 year ago

1,112 Plays / 9 Likes / 4 Comments

[+ More details](#)**DePaul IPD359 Week 4 - Web Development With Python and Web2py**

from mdipierro / Added 11 months ago

681 Plays / 9 Likes / 0 Comments

[+ More details](#)**web2py "Shootout" video tutorial**

from mdipierro / Added 6 years ago

23.1K Plays / 9 Likes / 2 Comments

This is an old video, watch this one instead:  
<http://www.vimeo.com/3703345>[+ More details](#)**Upload a video****Organize videos****BROWSE VIDEOS**

Here are all the videos **you** have uploaded to Vimeo. You can sort them, search them, and more. To batch edit or delete videos, visit the **Organizer**.



All the rest of your Vimeo stuff is down below, in the box conveniently labeled "Your Stuff."

**YOUR STUFF**

More stuff from you

**99 Videos****3 Albums****NEED HELP?**

If you have questions about what's on this page, look here first: **Help** / **FAQ**

**Keyboard Shortcuts**

← Prev page

→ Next page

K Prev video

J Next video

**My Videos**