CSRI SUMMER PROCEEDINGS 2022

CSRI Summer Program
The Center for Computing Research at Sandia National Laboratories

Editors:
S.K. Seritan and J.D. Smith
Sandia National Laboratories

November 11, 2022

SAND#: SAND2022-10280R

Preface

The **Computer Science Research Institute (CSRI)** brings university faculty and students to Sandia National Laboratories for focused collaborative research on Department of Energy (DOE) computer and computational science problems. The institute provides an opportunity for university researchers to learn about problems in computer and computational science at DOE laboratories, and help transfer results of their research to programs at the labs. Some specific CSRI research interest areas are: scalable solvers, optimization, algebraic preconditioners, graph-based, discrete, and combinatorial algorithms, uncertainty estimation, validation and verification methods, mesh generation, dynamic load-balancing, virus and other malicious-code defense, visualization, scalable cluster computers, beyond Moore's Law computing, exascale computing tools and application design, reduced order and multiscale modeling, parallel input/output, and theoretical computer science. The CSRI Summer Program is organized by CSRI and includes a weekly seminar series and the publication of a summer proceedings.

1. CSRI Summer Program 2022. In 2022, the CSRI summer program was executed in a hybrid fashion. This marked the first return to an in-person component since 2020. Most students were able to at least visit the Albuquerque or Livermore campuses during the internship. The summer program included included students from 1400 – the Center for Computing Research (CCR) and 8700 – the Center for Computation & Analysis for National Security. This year's program included the traditional *Summer Seminar Series* and *Summer Proceedings* and continued the third annual *Virtual Poster Blitz*. Additionally, this year we partnered with the Engineering Sciences Summer Institute (ESSI) to form the Computer Science and Analysis Institute (CSA), which provided four sessions on professional development with Dr. Jacquilyn Weeks from Word Tree Consulting. We also hosted a speaker from the Center of Cyber Defenders for a joint talk during our summer seminar series, and provided an opportunity to connect with a Sandia librarian for a virtual resource tour. Outside of technical work, we hosted a virtual escape room and provided tours around facilities such as data centers and the National Ignition Facility (NIF) at Lawrence Livermore National Laboratory.

2. Seminar Series. The CSRI Summer Seminar Series is a quintessential part of the CSRI Summer Intern Program Experience. Students are exposed to a broad showcase of research from across Sandia, enriching their knowledge of the labs while providing introductions to novel subject areas. The theme for 2022 was **Sandia's Mission Campaigns & Research Foundations and CSRI**. These talks were focused on presenting exciting research conducted by CSRI staff while simultaneously addressing the important question "Why is Sandia doing this research?" We extend our deepest thanks to the staff who spoke at the 2022 Seminar Series. These speakers and their talk titles are listed in Table 2.1.

Table 2.1: List of talks and speakers at the 2021 Seminar Series

Date	Name	Org	Title
6/14	Jim Stewart	1400	Sandia Computing & Information Science: Introduction & Overview.
6/21	Keita Teranishi	8753	Toward Scalable and Productive Resilience Models for Extreme Scale Computing Systems.
6/28	Asmeret Naugle	1462	Computational Social Science for National Security.
7/7	Tom Kroeger	8767	Demythifying Cyber Security.
7/12	Bert Debusschere Caitlin Curry	8351	The UQTk C++/Python Toolkit for Uncertainty Quantification: Overview and Applications.
7/19	Mauro Perego	1441	Ice Sheet Modeling: A Scientific Computing Application.
7/27	Stephen Bond	1442	Plasma Simulation Software Development for Fusion Power on the Z Machine.
8/2	Fred Rothganger	1421	Large-scale Brain Modeling.
8/9	Ojas Parekh	1464	Quantum Discrete Optimization (more than a jumble of words?).
8/15	Steve Plimpton	1444	Particle Applications and HPC.
8/23	Irina Tezaur	8734	An Overview of the CLDERA Grand Challenge LDRD Project: Developing a Novel Foundational Approach for Attributing Climate Impacts.
8/30	Megan Dahlhauser Kenneth Rudinger	1425	Quantum Characterization, Verification, Validation and Benchmarking.

3. Proceedings. All students and their mentors were strongly encouraged to contribute a technical article to the CSRI Proceedings. For many students, these proceedings are the first opportunity to write a research article. These proceedings serve both as documentation of summer research and also as research training, providing students the first draft of an article that could be submitted to a peer-reviewed journal. Each of these articles has been reviewed by a Sandia staff member knowledgeable in the technical area, with feedback provided to the authors.

Contributions to the 2022 CSRI Proceedings have been organized into three categories: *Computational & Applied Mathematics, Software & High Performance Computing* and *Applications.*

All participants and their mentors who have contributed their technical accomplishments to the proceedings should be proud of their work and we congratulate and thank them for participating. Additionally, we would like to thank those who reviewed articles for the proceedings. Their feedback is an extremely important part of the research training process and has significantly improved the proceedings quality. Our many thanks are extended to these reviewers: Patrick Blonigan ✧ Pete Bosler ✧ Ron Brightwell ✧ John Carpenter ✧ David Ching ✧ Matthew Curry ✧ Eric Cyr ✧ Saibal De ✧ Jed Duersch ✧ Danny Dunlavy ✧ Chris Eldred ✧ Kurt Ferreira ✧ Michail Gallis ✧ Jared Gearhart ✧ Gianluca Geraci ✧ Christian Glusa ✧ Oksana Guba ✧ Kaylin Hagopian ✧ Bill Hart ✧ Park Hays ✧ Jonathan Hu ✧ Lucas Kocia ✧ Hemanth Kolla ✧ Paul Kuberry ✧ Scott Levy ✧ Jonathan Lifflander ✧ Dave Littlewood ✧ Jay Lofstead ✧ Payton Lindsay ✧ Kathryn Maupin ✧ Megan McCarthy ✧ Stan Moore ✧ Miranda Mundt ✧ Kyle Neal ✧ Stephen Olivier ✧ Eric Parish

✧ Kevin Pedretti ✧ Kara Peterson ✧ Eric Phipps ✧ Steve Plimpton ✧ Teresa Portone ✧ Elaine Raybourn ✧ Drew Rohskopf ✧ Kenneth Rudinger ✧ Mohan Sarovar ✧ Tom Seidl ✧ Ember Sikorski ✧ Anh Tran ✧ Ryan Terpsma ✧ Irina Tezaur ✧ Brad Theilman ✧ Kevin Thompson ✧ Ray Tuminaro ✧ Craig Ulmer ✧ Craig Vineyard ✧ Ed Walsh ✧ Matthew Wong ✧ Mitch Wood ✧ Tian Yu Yen

4. Virtual Poster Blitz. In preparation for the proceedings, a *virtual poster blitz* was held on 7/21/2022 where all interns hired under the CSRI intern posting were invited to participate. These students submitted a summary slide of their current results or intended research for their summer project, and gave a two minute elevator-pitch style presentation to their peers. Other interns outside of CSRI were also invited to participate with their mentor's approval. This event provided an opportunity to formalize work directions, socialize their ideas, as well as offering a chance to network and interact with other interns and staff across the labs. The slides from the virtual poster blitz event are published under SAND number SAND2022-9886 O.

5. Professional Development with Word Tree Consulting and Dr. Jacquilyn Weeks. This year, in partnership with the CSA, our program invited Dr. Jacquilyn Weeks to give four presentations. Dr. Weeks is the founder of Word Tree Consulting and is an expert in helping scientists articulate their research ideas to a variety of audiences.

The first of this series of talks on 6/22 focused on writing a more efficient and effective journal articles. Students were given excellent advice on how to avoid stalls in writing and how to effectively plan for writing technical articles. Following in this theme, the followup talk on 6/27 explored best practices for tailoring writing toward a specific journal. Here, students learned that researching your journal and their language and style preferences are just as important as communicating your work clearly.

Leading into the Poster Blitz, the final two talks from Jacquilyn were on best-practice poster presentations and were held on 7/7 and 7/18. Presented in a workshop and interactive format, students learned to identify common problems with technical posters. Students were also given an opportunity to critique slides from previous years and learn how to build the best poster and presentation possible.

Throughout all four talks, Dr. Weeks provided excellent professional development advice, including tips on preparing resumes and how to apply for student awards.

Deepest Thanks. Holding the first hybrid summer intern program came with new challenges not seen from previous years. We had numerous technical glitches with teleconferencing equipment. The success of our program was primarily due to the hard work of our students and the dedication of their staff mentors. We would like to thank all students and mentors for their extraordinary patience and dedication. We would also like to thank the program managers for the CSRI Summer Intern Program, Michael Wolf (1465) and Jerry McNeish (8734). Their support has been critical throughout the organization of the seminars and the editing of these proceedings. Furthermore, the CSRI Summer Intern Program would not be possible without the administrative support of Greg Bristol, Becky March, Ashley Jones, Hailey Poole, Sandra Portlock, and Cookie Santamaria. We would also like to extend a very special thank you to Alyssa Skulborstad, coordinator of the CSA. Our cooperation allowed all of our summer interns this year to have new experiences and interactions with Sandia.

S.K. Seritan
J.D. Smith

November 11, 2022

Table of Contents

Articles

I. Computational & Applied Mathematics

Computational & Applied Mathematics are concerned with the design, analysis, and implementation of algorithms to solve mathematical, scientific, or engineering problems. Articles in this section describe methods to design new neural network architectures or algorithms for quantum computers, discretize and solve partial differential equations, couple multiphysics systems of equations, and analyze sensitivity & quantify uncertainty in complex systems.

1. *Ahmed, Tchoua* and *Lofstead* examine the impact of pseudo-random number generation on training machine learning models reproducibly.
2. *Bandy, Portone* and *Acquesta* explore a data-driven correction to the dynamics of systems epidemiology by embedding a neural network in a modified Susceptible-Infectious-Removed (SIR) model.
3. *Barnett, Tezaur* and *Mota* develop an extension to the Schwarz alternating method to couple projection-based reduced order models to full order models or other reduced order models.
4. *Davis, Geraci, Wildey* and *Motamed* train neural networks which exploit multifidelity data and show how to expand multifidelity datasets to improve and reduce stochasticity in neural network decisions.
5. *de Castro, Kuberry, Tezaur* and *Bochev* extend synchronous partitioning schemes for finite element methods to the coupling of projection-based reduced order models with either a finite element model or another reduced order model.
6. *Galioto, Safta, Jakeman* and *Gorodetsky* investigate the use of tensor-trains in variational inference algorithms for learning dynamical systems with quantified uncertainty.
7. *Gilani, Holman, Kallaugher* and *Parekh* analyze the quantum query complexity of triangle detection in the adjacency list model.
8. *Higgins, Boman, Loe* and *Yamazaki* compare the sketched Generalized Minimal Residual Method to alternate Krylov subspace methods and show potential applications to non-symmetric computational fluid dynamics.
9. *Larsen, Grace, Baczewski* and *Magann* develop a feedback-based quantum algorithm for ground state preparation on the Fermi-Hubbard model of correlated materials.
10. *Munoz, Pathak* and *Rackers* combine equivariant neural networks with Hellman-Feynman optimized basis sets to learn quantum mechanical densities that allow the evaluation of high-accuracy forces for quantum simulation.
11. *Muollo, Raybourn, Weirs, Milewicz, Mauldin* and *Otahal* apply a character-based approach to measuring semantic similarity with the aim to improve duplicate identification in natural language data.
12. *Rickord, Poldervaart, Wang, Dellana* and *Cardwell* developed an event sensor data pipeline to test spiking neural networks for motion segmentation in end-to-end neuromorphic systems at the edge.
13. *van Heyningen, Blonigan* and *Parish* explore the utility of weighted and penalized residual minimization formulations of reduced order models for steady hypersonic flows.

14. *Voronin, Tuminaro, Olson* and *MacLachlan* investigate a novel monolithic algebraic multigrid solver for the Stokes problem discretized with stable mixed-finite elements.

S.K. Seritan
J.D. Smith

November 11, 2022

MAKING RANDOMNESS PORTABLE ACROSS PROGRAMMING LANGUAGES

HANA AHMED*, ROSELYNE TCHOUA†, AND JAY LOFSTEAD‡

Abstract. Science is a practice of systematically studying something and offering data and evidence to reach a conclusion. With first principles simulations, basic physics are used to model some phenomena leading to consistent, repeatable results. With an incomplete physics model or models too complex or costly to run for a given task, machine learning and artificial intelligence (ML/AI) are being used to estimate what the missing physics would be if we could meet our goals with a first principles approach. Our work has been exploring how to ensure ML is capable of offering a scientific level of consistency so we can trust our science applications incorporating ML models.

Our earlier work examined the impact of pseudo-random numbers on model quality. For this study, we have examined the pseudo-random number generation algorithms used to seed essentially all ML algorithms to ensure that model generation can be performed by other scientists to achieve identical results.

1. Introduction. ML is having an outsized influence on seemingly every aspect of life today. Research is demonstrating the efficacy of ML models to solve problems that were previously difficult and/or slow to solve using other methods. The base statistical models ML uses should be consistent from run to run prompting the incorporation of a random element at least during model initialization. This randomness keeps models from being identical and can offer vastly improved quality by selected different, far more favorable starting conditions [1]. Even with this randomness, because we use pseudo-random number generator (PRNG) software rather than true random sources, we should be able to generate the same model again.

To ensure that work incorporating ML still meets the basic standard of science (i.e., others can arrive at the same results using the same data), properly recording and reporting *all* environment and starting conditions is necessary for another researcher to attempt to reproduce or replicate the work. One frequently missing parameter is the pseudo-random number seed used to initialize the models presented in research papers. In our previous work [1], we established the potentially extreme impact the pseudo-random number seed has on model accuracy. We were able to demonstrate that using the same seed, we were able to generate an identical model. The differences based on different seeds prompted further exploration to understand how these seeds can affect reproducibility of work that incorporates ML models.

Given anecdotal evidence that ML papers are frequently difficult to reproduce, we have been investigating the root causes, starting with the acknowledged lack of PRNG seeds. Our own experience has demonstrated that PRNG seeds are important but often missing characteristics; however, they are not the only factor [2]. In this work, we are taking a different approach in that we are attempting to take a model identically reproducible using a ML algorithm using one language, such as Python, and see if we can reproduce it using the same algorithm, configured identically, written in a different language, such as C++. With Python being a frequent ML exploratory tool given the rich, easy to use tools, we know and expect a large fraction of research publications use Python-based environments for their research explorations. The question we are pursuing is ***can a Python-built ML model be reproduced in C++ and vice versa?*** Our initial experimentation demonstrated extreme failures. A simple test to see if the PRNGs are out of sequence by printing out the next ten numbers showed no overlap between the C++ and Python code bases! This prompted

*Scripps College, hahmed@hmc.edu
†DePaul University, rtchoua@depaul.edu
‡Sandia National Laboratories, gflofst@sandia.gov

deeper explorations to determine what is necessary to make this portability possible. If the algorithms and initial conditions are the same, then the outcomes should be the same.

Ensuring we were using the exact same algorithm, we drop back to exploring simply the PRNGs themselves and their portability. With the potential impact on the generated model quality, understanding the portability of pseudo-random number generation—the starting point for essentially all ML algorithms—is core to portability and reproducibility, and therefore trust, of ML-incorporated science. Our initial results show a troubling lack of portability, despite controlling all the user-accessible parameters. This indicates the need for stronger documentation requirements to demonstrate that the science is repeatable. We offer insights into some of the challenges, the implications for other ML model generation environments (e.g., GPUs), and recommendations to close this gap preventing ML-infused science from reaching the true spirit of *science*.

The rest of the paper is organized as follows. First in Section 2, we provide an overview of PRNGs and the impact on ML model generation as well as a brief glimpse into challenges in reproducing ML publications by third parties.Next in Section 3, we describe the experiments we have run to test and demonstrate the issues at hand.In Section 4 we offer a discussion and recommendations before concluding in Section 5.

2. Related Work. PRNGs, the primary source of randomness in a machine learning algorithm, are themselves algorithms that generate sequences of pseudo-random values. These are distinct from hardware random number generators (HRNGs, also called true random number generators), which generate random values using physical rather than algorithmic processes. The advantages of HRNGs are surveyed by Stipčević et al. [13]. However, HRNGs generate a limited number of random bits per second and produce non-reproducible sequences, as described by Haahr [6], whereas PRNGs are lower cost and easier to implement, making PRNGs more desirable to many.

In previous work, we examine the impact of PRNG seeds on final model accuracy of common machine learning algorithms [1]. This work demonstrates that just by varying the PRNG seeds, the generated model accuracy can vary by up to 60 accuracy points or more. To be precise, the same data, configuration, and algorithm with a different seed could result in a generated model accuracy of 20% or 80%. Please refer to our previous paper for more details.

Both this paper and our previous paper conduct experiments using a Mersenne Twister type PRNG [8], which is considered a higher quality PRNG than common linear congruential engines (such as `std::rand()` offered in the C++ Standard Library) due to differences in speed, memory, range, and period length [7]. Additionally, the Philox PRNG, which is currently offered in the C++ and Python TensorFlow libraries, can guarantee reproducibility across high-performance computing platforms such as GPUs and parallel programs [10].

In the past, computer scientists have regenerated pseudo-random number sequences by recording the PRNG seed. Sen et al. [12] uses a "capture-and-replay" method to replay data races using the same PRNG seed. Frederickson et al. [5] uses a similar approach to reproduce Monte Carlo trees. Ahmed et al. [1] uses this approach in a series of experiments showing the extent to which the PRNG seed, train/test split ratio, and train and test data sets can impact final accuracy scores of various ML classification models on scientific data sets. Our paper extends on this line of work by showing that the PRNG seed value alone is not sufficient for reproducing pseudo-random numbers across different PRNG implementations and programming languages. Rather, additional information such as the PRNG state and parameter values are required to reproduce the desired number sequences.

Randomness in neural networks and the importance has also been investigated by Scardapane et al. [11]. Pham et al. [9] and Alahmari et al. [3] study the repeatability of deep learning models using Python's TensorFlow and Keras libraries given the presence of uncontrolled randomness. Pham et al. studies the variance in accuracy scores of deep learning training algorithms given changes in training data, algorithm, network, and PRNG seed. Alahmari et al. finds that while deep learning models can be repeated by saving the random initializations, computational variances may result in a different learning process for a given run. These papers experiment with ML algorithms unsurprisingly written in Python, using common ML libraries such as TensorFlow, Keras, and Scikit-learn. Our paper establishes the conditions under which pseudo-random number sequences generated by common Python implementations of PRNGs (namely, the Mersenne Twisters provided in Python's Random and NumPy libraries) can be reproduced by a C++ PRNG. We offer suggestions and modifications for ensuring that ML programs in Python generate pseudo-random number sequences that are reproducible in different implementations of the same PRNG—a Mersenne Twister—both in Python and in C++.

3. Experiments. To explore this question, we look at how ML algorithms generate and use pseudo-random numbers in Python and C++, the two most common ML algorithm languages. This study consists of three experiments with three different implementations of a single PRNG algorithm, where two implementations are made publicly available in Python and one in C++. In the first experiment, we attempt to reproduce integer sequences generated by all three Mersenne Twisters. In the second experiment, we pursue the same goal with floating point number sequences. Although we expect that using the same PRNG with identical PRNG seeds would produce identical pseudo-random number sequences, our experiments capture the necessary steps for achieving results that are as closely identical as possible across languages and implementations. Finally, we present validation that using the insights from these experiments, we can demonstrate portability across languages. The code, PRNG seeds, and PRNG state values used to obtain our experiment results are publicly available on GitHub.[1]

Our experiments are conducted on an Intel® Core^{TM} i7-8665U CPU @ 1.90GHz 2.11GHz, with 32 GB RAM. The operating system is Windows 10 and uses the C++ Clang Compiler for Windows (11.0.0) and C++ Clang-cl for v142 build tools (x64/x86). The Mersenne Twister parameters are the Python 3.10.5 Random values, the Python NumPy v1.23 values, and the C++11 standard values. Further experimentation using different numerical distributions is left for future work.

3.1. PRNG Algorithm. The following experiments were conducted using the same type of PRNG—a Mersenne Twister (also referred to as mt19937), which is a PRNG of 32-bit numbers with a state size of 19937 bits. We evaluate the reproducibility of random number sequences for the Python Random and NumPy libraries and the C++ Standard Library. In our first experiment, we record the pseudo-random number sequences produced when using integer functions. Although using the same seed value for the same PRNG algorithm should guarantee identical results, our first experiment finds that this is not always the case. So, we also record the pseudo-random number sequences generated when both the PRNG and initial state are identical. Our second experiment follows a similar layout, but testing the floating point generator functions rather than integers.

We noted while conducting this experiment set that in Python, calling a PRNG function multiple times consecutively generates different pseudo-random number sequences when the order and number of calls to the functions vary. In our experiments, we test the outcomes of resetting the PRNG seed prior to calling each python PRNG function in separate loops, as well as the outcomes of setting the seed value once at the beginning of the program

[1] https://github.com/hahmed17/mersenne-twister-comparison

Table 3.1

Integer outputs of three Mersenne Twisters with the same seeds. The seed for each Python Mersenne Twister is set once before all the functions are called together in a loop.

Pseudo-Random Number Sequences (first 3 numbers)				
Seed	**random. randint() (Python)**	**numpy. random. randint() (Python)**	**std::mt19937 (C++)**	**std::uniform_int_ distribution <u_int32_t> (C++)**
	1806341205	2588848963	2357136044	2357136044
0	2195908194	2678185683	2546248239	2546248239
	2046968324	3830135878	3071714933	3071714933
	1181241943	3143890026	1608637542	1608637542
42	3163119785	1914837113	3421126067	3421126067
	1812140441	3720198231	4083286876	4083286876
	3240433680	2501096769	2229904676	2229904676
1659656101	4102924446	2668794612	1504551105	1504551105
	2511712355	3792665969	2530607890	2530607890
	2980296527	3079973343	3408234313	3408234313
1659656104	3014878157	2173885170	1229771292	1229771292
	309439972	822372728	3057366171	3057366171
	265325381	1811291956	2282898853	2282898853
1659065468	197524735	2228452808	2671758029	2671758029
	1836161563	2043038973	3412855078	3412855078

and calling the Python PRNG functions consecutively. The first experiment examines the reproducibilty of pseudo-random integer generator functions, whereas the second experiment examines pseudo-random floating point generator functions.

3.2. Integers. This first experiment deals with the following pseudo-random integer generator functions: in Python, Random's `random.randint()` and NumPy's `numpy.random.randint()`; and in C++, the <random> library's `std::mt19937` generator and `std::uniform_int_distribution<u_int32_t>`. Each function draws pseudo-random integers from the discrete uniform distribution. For each integer function, we set the range of values to be either $[0, 4294967295]$ or $[0, 4294967296)$, where 4294967295 is the maximum value producable by `std::mt19937`. In the first experiment, we seed each of the Python Random, Python NumPy, and C++ Standard Library Mersenne Twisters with the same integer value five different times, and record the resulting pseudo-random number sequences. Note that the PRNG state for any of the Mersenne Twisters is not recorded at this stage. Table 3.1 contains samples of the pseudo-random number sequences generated by each PRNG function using five different seed values: 0, 42, and three arbitrary system `datetime()` generated integer values. For this set of outcomes, the Python Mersenne Twisters are seeded once prior to calling the PRNG functions.

We find that although the two C++ generators produce identical pseudo-random integer sequences, the Python function outputs are identical to neither each other nor to the C++ outputs. Our next step addresses this problem by configuring NumPy `numpy.random.randint()` to output 32-bit integers (this can be done by setting the `dtype` parameter to `'<u4'`) as the other PRNG functions already do by default. We then rerun each function using the same seed values as in the previous part of this experiment.

We find that we are able to reproduce the C++ integer sequences in NumPy when the output of `numpy.random.randint()` is set to a 32-bit integer. Next, we test whether the

Table 3.2
Integer outputs, NumPy configured to output a 32-bit integer, and Mersenne Twister seeds reset before each function is called in a loop.

Pseudo-Random Number Sequences (first 3 numbers)		
Seed	**random.randint() (Python)**	**numpy random.randint() (Python)**
0	3626764237 1806341205 2195908194	2357136044 2546248239 3071714933
42	2746317213 1181241943 958682846	1608637542 3421126067 4083286876
1659656101	1376662229 3240433680 402409861	2229904676 1504551105 2530607890
1659656104	2286574984 2980296527 3014878157	3408234313 1229771292 3057366171
1659656106	265325381 197524735 2734704416	2282898853 2671758029 3412855078

results hold when we re-structure our Python tests so that instead of seeding the Python Mersenne Twisters only once, we set the seed values before every call to a PRNG function. For space reasons, we omit the actual results, but can provide them on request.

These are promising results for the reproducibility of pseudo-random integers across programming languages: the Python NumPy generator produces the exact same pseudo-random number sequences as those produced in C++ with the same seeds (see Table 3.1). In conducting this series of tests without configuring Python NumPy to output 32-bit integers—and this time resetting the PRNG seed before each function call—we find that the resulting pseudo-random number sequences remain identical to those in C++.

However, we see that the Python Random library integer generator produces number sequences that share no overlap with the remaining three functions. In fact, the sequences in Table 3.2 differ from those in Table 3.1 for Python Random. This indicates that despite using the same seed value, the pseudo-random number sequences produced by Random's `random.randint()` function changes solely by changing when the PRNG seed is set.

We take one more step in attempting to address the Random library integer sequences by setting the Random Mersenne Twister state to be identical with those of the NumPy and C++ Mersenne Twisters for each seed value. This resulted in no change to the output integer sequences from Python Random's `random.randint()`, and we leave studying the difference between Python Random's Mersenne Twister from that of Python NumPy and C++ as a future work.

3.3. Floats. The second experiment evaluates pseudo-random float generation using the following PRNG function calls to Mersenne Twister generators in Python and C++: in Python, the Random library's `random.random()`, NumPy's `numpy.random.rand()`, and NumPy's `numpy.random.uniform()` functions; and in C++, the `<random>` library's `uniform_real_distribution<float>` function. Each function draws a random floating

Table 3.3

Floating point outputs of three Mersenne Twisters. The seed for each Python Mersenne Twister is set once before all functions are called together in a single loop.

Pseudo-Random Number Sequences (first 3 numbers)				
Seed	**random() (Python)**	**numpy rand() (Python)**	**numpy uniform() (Python)**	**std::mt19937 (C++)**
0	0.844421	0.548813	0.715189	0.548814
	0.258916	0.857945	0.847251	0.592845
	0.404934	0.645894	0.437587	0.715189
42	0.639426	0.374540	0.950714	0.37454
	0.223210	0.779691	0.596850	0.796543
	0.676699	0.155994	0.058083	0.950714
1659065463	0.263359	0.074773	0.047212	0.0747732
	0.621987	0.587986	0.067592	0.167238
	0.157480	0.052192	0.396804	0.0472126
1659065466	0.405821	0.014769	0.872470	0.0147693
	0.505245	0.392967	0.541198	0.229618
	0.530635	0.727041	0.599839	0.87247
1659065468	0.746031	0.299617	0.498290	0.299618
	0.982195	0.458595	0.780604	0.441102
	0.777537	0.185734	0.444176	0.498291

point number from a uniform distribution over $(0, 1]$. As in the first part of the experiment, each Mersenne Twister is initialized with an identical seed value five different times, where the seed values are: 0, 42, and three arbitrary system `datetime()` values. Table 3.3 contains samples of the pseudo-random number sequences generated by each Mersenne Twister at each of the five seed values. In this first set of tests, we again seed each Python Mersenne Twister once before calling all PRNG functions in a single loop.

From this first set of tests, we find that none of the four PRNG functions produce identical pseudo-random float sequences. We note that while the sequences differ between NumPy `numpy.random.rand()` and C++ `uniform_real_distribution <float>`, both sets of sequences overlap in their first number for each given seed. We conjectured that this discrepancy may be resolved by resetting the PRNG seeds before calling both functions. So, we conducted another set of tests with all four of our float generation functions, this time resetting the PRNG seed before each PRNG function call. Results are found in Table 3.4.

In our next set of tests, in addition to using the same seed value for each Python Mersenne Twister, we also use the same initial PRNG state. For each seed value, we first generate a pseudo-random float sequence using NumPy `numpy.random.rand()`. We are then able write the state value to the other three Mersenne Twisters, using `numpy.random.get_state()` to get the initial state values, then `numpy.random.set _state()` and `random.setstate()` to load the state into the NumPy and Random Mersene Twisters, and the `<<` operator to pass state values into the C++ `std:: mt19937` generator. Results of this test set are shown in Table 3.5.

Controlling both the PRNG seed and state and resetting the seed and state to their original values before generating each sequence, we find that all three Python PRNG functions generate identical floating point sequences using this particular method (see Tables 3.4 and 3.5). Furthermore, the sequences overlap with those generated in C++ (see Table 3.3). However, the results from both languages are not identical; only every other floating point generated by the C++ Mersenne Twister appears in the Python sequences. In an additional

Table 3.4
Floating point outputs of three Python Mersenne Twister functions with the same seeds. The seed for each Mersenne Twister is reset before each function is called in separate loops.

Pseudo-Random Number Sequences (first 3 numbers)			
Seed	**random.random() (Python)**	**numpy random.rand() (Python)**	**numpy random.uniform() (Python)**
0	0.844421 0.258916 0.404934	0.548813 0.857945 0.645894	0.715189 0.847251 0.437587
42	0.639426 0.223210 0.676699	0.374540 0.779691 0.155994	0.950714 0.596850 0.058083
1659065463	0.263359 0.621987 0.157480	0.074773 0.587986 0.052192	0.047212 0.067592 0.396804
1659065466	0.405821 0.505245 0.530635	0.014769 0.392967 0.727041	0.872470 0.541198 0.599839
1659065468	0.746031 0.982195 0.777537	0.299617 0.458595 0.185734	0.498290 0.780604 0.444176

test where we fixed only the seed value (not the initial state) for each function, then reset the seed value before calling each function in separate loops, we found that while neither set of NumPy results change, Random's `random.random()` function generates completely different outputs. Identifying the root of these inconsistencies is left as future work.

Given the regular interval for matching floats, we suggest as a work-around using only every other float in the C++ sequences to obtain results that are reproducible by Python Mersenne Twisters. We conduct a set of tests where we for every call to `std::uniform_real_distribution<float>` we generate an extra float to make up for the every-other difference between C++ and NumPy. Results of this test set are in Table 3.6.

Given our adjustments to our C++ code, we are able to obtain float sequences that are essentially identical to those generated by Python Mersenne Twisters (see Table 3.7).

3.4. K-Means portability between Python and C++. We use a from-scratch implementation of the k-means clustering algorithm developed in Python and translated to C++. Both k-means implementations use a PRNG to generate integers and randomly initialize the starting centroids. In both implementations, a single call to a PRNG integer functon is made in a single loop, eliminating the need to reset the PRNG seed at any time. Each k-means clustering model is fitted and tested on the same instances from the Iris data set [4], and we test a variety of seed and cluster values.

We compare the output of C++ and Python k-means clustering models, where the Python algorithm calls Python Random's `randint()` function. The final k-means model accuracies differ from one another by as much as 37.619% in accuracy points (as seen in Table 3.7). We then adjust our Python k-means implementation to call the Python NumPy's `randint()` function. Using the NumPy PRNG, we specify the output type to be a 32-bit integer, and use the same seed values and states as in the initial experiment. Now, our Python k-means clustering models guarantee identical accuracy as our C++ models (note

Table 3.5

Floating point outputs of Python Random and NumPy uniform Mersenne Twisters with the same seeds and initial states. The seed is reset before each function is called in separate loops.

Pseudo-Random Number Sequences (first 3 numbers)		
Seed	**random.random() (Python)**	**numpy.random.uniform() (Python)**
0	0.548813 0.715189 0.602763	0.548813 0.715189 0.602763
42	0.374540 0.950714 0.731993	0.374540 0.950714 0.731993
1659065463	0.074773 0.047212 0.304362	0.074773 0.047212 0.304362
1659065466	0.014769 0.872470 0.064023	0.014769 0.872470 0.064023
1659065468	0.299617 0.498290 0.865369	0.299617 0.498290 0.865369

that the accuracy scores from our Python k-means models are rounded off).

The adjusted code enables a C++ model to be recreated in Python, and vice versa. For the opposite direction the C++ code would need to make similar adjustments outputting both the seed and state for reading into the Python code. With these adjustments, the model generates identical accuracy scores for both C++ and Python.

In order to replicate ML model accuracy using pseudo-random float generation (i.e., using `random.random()` or `numpy.random.rand()` in Python, and `uniform_real _distribution<float>` in C++), the necessary adjustments to the C++ implementation requires using the same PRNG seed and state values. For convenience in our code setup, we place each of the seed and state values in separate text files and read them in for initializing the k-means algorithm. The Python implementation is adjusted to use NumPy's PRNG state and output both the seed and state values into the text files that can now be part of the reproducibility artifacts.

3.5. Monte Carlo Simulations. Our Monte Carlo simulation estimates the integral of e^{-x^2} over the interval $[0, 1]$. Both implementations estimate over 1000 intervals. In Python, we use the `numpy.random.rand()` function; and in C++, the `std::uniform_real_distribution<float>` function with a `std::mt19937` generator. We run our C++ simulation with no adjustments to the PRNG call, and receive an output of 0.738151 while Python yields 0.7402087652084175. To compensate for the every other match, in the next step of our test, we add an extra call to the C++ Mersenne Twister, where the second pseudo-random float generated per iteration is unused. We also load the PRNG state used by our NumPy Mersenne Twister into our C++ Mersenne Twister. With these adjustments, we run the C++ simulation again, and this time receive an output of 0.740209—effectively equivalent to the Python simulation's output.

3.6. C++ Compilers: Intel, GNU, and LLVM. We tested our C++ PRNG functions on three different compilers—Intel, GNU, and LLVM—and verified that each compiler produces the same random number sequences (given the same PRNG, functions,

Table 3.6
Floating point outputs in of C++ Mersenne Twister with same seed and initial state, and an extra call is made to the C++ Mersenne Twister.

Pseudo-Random Number Sequences (first 3 numbers)	
Seed	**std::uniform_real_distribution<float> (C++)**
0	0.548814 0.715189 0.602763
42	0.37454 0.950714 0.731994
1659065463	0.0747732 0.0472126 0.304362
1659065466	0.0147693 0.87247 0.0640231
1659065468	0.299618 0.498291 0.865369

Table 3.7
C++ vs. Python k-means clustering models.

Seed	**# of clusters**	**C++ accuracy**	**Python accuracy (pre-adjustments)**	**Python accuracy (post-adjustments)**
1659578083	2	54.2857%	6.6667%	54.2857%
1659560144	4	38.0952%	44.7619%	38.0952%
1659560381	6	0.0	39.0476	0.0

and seed values).

4. Discussion and Future Work. These experiments establish guidelines for reproducing pseudo-random number sequences using different implementations of the same PRNG, a Mersenne Twister, in two different programming languages, Python and C++. Our first experiment identifies two possible steps for reproducing integers with different PRNG functions in Python and C++: modifying sequences NumPy's `numpy.random.randint()` to generate 32-bit integers, and/or reinitializing the PRNG seed each time prior to calling `numpy.random.randint()` in a loop. Unfortunately, we found that it is nearly impossible to reproduce integer sequences generated by Python Random's `random.randint()` using any other Mersenne Twister, and we leave the root cause of this inconsistency in Python Random as a future work.

Our second experiment addresses floating point sequences generated by PRNG functions in Python Random, Python NumPy, and C++ `<random>`. We find that using the same seed and state does in fact guarantee reproducible floating point values generated by the Python Random and NumPy Mersenne Twisters. However, it is neary impossible to reproduce in Python the floating point numbers generated by a C++ Mersenne Twister, as only every other float generated in C++ sequences overlaps with the Python sequences. This difference can be worked around by either: making a second call to the C++ PRNG to compensate for the every-other difference; or, because integer reproducibility can be guaranteed by the results in our first experiment, generating a pseudo-random integer and dividing by the maximum value to obtain a float that can be reproduced by the integer generators in Python. We leave determining the root cause of the discrepancy between C++ and Python pseudo-random floats as a future work.

The third experiment both demonstrates the divergent accuracy in an ML model and validates that the proposed adjustments can generate identical accuracy values. Additionally, this behavior was demonstrated for a Monte Carlo integration. Overall, we find that reproducible pseudo-random number sequences are not guaranteed across these different implementations: Python Random and Python NumPy integers; Python Random and C++ Standard Library integers; and Python and C++ Standard Library floats. However, this paper establishes the necessary steps for guaranteeing reproducible integers using Python NumPy and C++ Standard Library, and reproducible floats using Python Random, Python NumPy, and C++.

5. Conclusion. Trusting that the work performed by one researcher can be recreated by another is the foundation of science. ML has added complexity to this process that has not been well understood leading to high quality work omitting key pieces of information necessary for another to recreate the same results. Our own work attempting to reproduce seemingly well documented ML-based research has led to failures and prompted this deeper look into what is truly necessary to achieve a *science*-based ML outcome. With PRNG sequences playing a role not just in ML algorithms, but also in Monte Carlo simulations as well as a host of other kinds of science codes, proper *and complete* documentation to enable another researcher to reproduce the same results is tricky. We hope this work sheds light on another reproducibility challenge and potential solutions

REFERENCES

[1] H. Ahmed and J. Lofstead, *Managing randomness to enable reproducible machine learning*, in Proceedings of the 5th International Workshop on Practical Reproducible Evaluation of Computer Systems, 2022, pp. 15–20.

[2] H. Ahmed, R. Tchoua, and J. Lofstead, *Failure sources in machine learning for medicine—a study*, in Proceedings of the 2nd International Workshop on E-Science Research Leading to Negative Results, 2022.

[3] S. S. Alahmari, D. B. Goldgof, P. R. Mouton, and L. O. Hall, *Challenges for the repeatability of deep learning models*, IEEE Access, 8 (2020), pp. 211860–211868.

[4] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[5] P. Frederickson, R. Hiromoto, and J. Larson, *A parallel monte carlo transport algorithm using a pseudo-random tree to guarantee reproducibility*, Parallel Computing, 4 (1987), pp. 281–290.

[6] M. Haahr, *random. org: Introduction to randomness and random numbers*, 1999.

[7] H. Haramoto, M. Matsumoto, and P. L'Ecuyer, *A fast jump ahead algorithm for linear recurrences in a polynomial space*, in International Conference on Sequences and Their Applications, Springer, 2008, pp. 290–298.

[8] M. Matsumoto and T. Nishimura, *Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Trans. Model. Comput. Simul., 8 (1998), p. 3–30.

[9] H. V. Pham, S. Qian, J. Wang, T. Lutellier, J. Rosenthal, L. Tan, Y. Yu, and N. Nagappan, *Problems and opportunities in training deep learning software systems: An analysis of variance*, in Proceedings of the 35th IEEE/ACM international conference on automated software engineering, 2020, pp. 771–783.

[10] J. K. Salmon, M. A. Moraes, R. O. Dror, and D. E. Shaw, *Parallel random numbers: as easy as 1, 2, 3*, in Proceedings of 2011 international conference for high performance computing, networking, storage and analysis, 2011, pp. 1–12.

[11] S. Scardapane and D. Wang, *Randomness in neural networks: an overview*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7 (2017), p. e1200.
[12] K. Sen, *Race directed random testing of concurrent programs*, in Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2008, pp. 11–21.
[13] M. Stipčević and Ç. K. Koç, *True random number generators*, in Open Problems in Mathematics and Computational Science, Springer, 2014, pp. 275–315.

VALIDATING NEURAL-NETWORK-CORRECTED DYNAMICAL SYSTEMS

RILEIGH BANDY*, TERESA PORTONE †, AND ERIN ACQUESTA‡

Abstract. The true dynamics of systems epidemiology are often governed by complex interactions that we cannot model exactly, and so we often use classic compartmental models (e.g., SIR model). However, high model error caused by simplifying assumptions can severely limit the predictive capability of these models for epidemic outbreaks. We explore the reduction of model error through a data-driven model correction—embedding a small neural network (NN) which takes current population counts as inputs—into a modified SIR model. This approach bridges the gap between application-specific, physics-based model inadequacy representations and general, statistical representations of model error. Physics-based model inadequacy representations can extrapolate beyond calibration regimes but are time-consuming to develop, while statistical representations of model error are easy and rapid to develop but have limited extrapolative capability. However, validation of the resulting corrected model is nontrivial. We investigate the robustness of common validation metrics and how the NN architecture and phenomenology captured in calibration data influence predictive capability.

1. Introduction. George Box famously stated, "since all models are wrong the scientist cannot obtain a 'correct' one by excessive elaboration." Instead, to find the most useful model of a phenomenon, one must reduce complexity without overly increasing inaccuracies. In computational modeling of infectious diseases, agent-based models, which model each individual and relationships as a network, have been widely used to evaluate infection control policies and inform the development of containment strategies [17]. While agent-based models are heterogeneous and can capture complex network interactions, they can be intractable due to computational expense and/or incomplete knowledge of the network. Therefore, low-fidelity compartmental models are often used, where the community population is assigned to compartments like susceptible, infectious, and removed. Compartmental models are significantly cheaper than agent-based models, but they are derived by making several strong simplifying assumptions (e.g., homogeneous and uniform mixing across the host population), which can lead to model error. Model error arises from inaccuracies in the model, such as simplifying assumptions or missing dependencies. These inaccuracies lead to discrepancy between model predictions and observed reality, casting into doubt their validity. Therefore, accurate assessments of model error are essential for evaluating the credibility of model predictions.

When the model error is too large to confidently use a low-fidelity model for predictions, we do not have to completely throw out the model. Instead, we can construct model corrections to reduce the discrepancy. A common approach is to add a stochastic term to the model predictions [8, 7]. This approach is nonintrusive, so it can be applied to a variety of models without requiring knowledge about the intricacies of the model. However, a baseline level of knowledge about compartmental disease models and the source of the model error is often available, and leveraging this information is often necessary for proper treatment of the uncertainty. It is important to respect the intrinsic nature of the model discrepancy when constructing a model correction if it is to reliably represent uncertainty in the predictions. Another approach is to embed application-specific, theory-informed inadequacy representations into the model [13, 12]. While this technique is broadly applicable, the inadequacy representation is highly specialized to the model it is embedded into and requires expert knowledge to construct. When we have limited knowledge pertaining to the missing

*Department of Computer Science, University of Colorado Boulder, rileigh.bandy@colorado.edu,
†Sandia National Laboratories, tporton@sandia.gov
‡Sandia National Laboratories, eacques@sandia.gov

information we are trying to capture, inadequacy representations can be hard to design. Therefore, we combine the two approaches by embedding a data-driven correction into the model to produce a so-called universal differential equation (UDE) [15]. Baseline knowledge gives us insight into where to embed the correction in the model, and data informs the form of the correction. Specifically, we embed a neural-network (NN) into a compartmental disease model to capture an unknown, nonlinear transition between compartments, as was done in [3].

Before a neural-network-corrected model can be used by decision makers, it must go through quantitative validation to confirm that the model predictions are in agreement with observations. Compartmental disease models are dynamical systems which evolve in time; correspondingly, the model correction, in the form of an embedded NN, evolves along with it. It is important to understand the time horizon over which the corrected model is valid. However, identifying the appropriate validation metric from the plethora of options in the literature provides yet another challenge since the window of validation will be sensitive to that choice of validation metric. There are several sources of uncertainty in the modeled system, including the values of disease parameters and uncertainties in observations. Maupin et al. recommend using all available uncertainty estimates, so we limit our investigation to probabilistic validation metrics [11]. We explore a range of validation metrics to see which one is the most appropriate for our application, and how the NN architecture and calibration scenario affect validity.

The rest of the paper is organized as follows. In Section 2, we introduce the noisy "true" model and our neural-network-corrected model. In Section 3, the details of the calibration are presented, and our probabilistic validation metrics are defined in Section 4. Our results are presented in Section 5, and concluding remarks are given in Section 6.

2. Compartmental Models.

The SIR model is a widely implemented dynamical model of infectious diseases, and it is the fundamental basis that captures the key driving factors governing systems epidemiology [6]. As the SIR model is the mathematical foundation upon which most high-fidelity, more complex epidemiology models are constructed, we will focus on the research challenges laid out by running controlled numerical experiments on the low-fidelity SIR compartmental model. It consists of three compartments: susceptible, infectious, and removed, and movement between the compartments is modeled by the following ordinary differential equations (ODEs):

$$\begin{aligned}\frac{dS(t)}{dt} &= -\beta \frac{I(t)}{N_{pop}} S(t) \\ \frac{dI(t)}{dt} &= \beta \frac{I(t)}{N_{pop}} S(t) - \gamma I \\ \frac{dR(t)}{dt} &= \gamma I(t).\end{aligned} \tag{2.1}$$

The SIR model is a closed system where individuals are not entering or exiting the community, and $S(t)$ is the susceptible population count, $I(t)$ is the infectious population count, and $R(t)$ is the removed population count. Therefore, the total population $N_{pop} = S(t) + I(t) + R(t)$ is fixed. The transition from susceptible to infectious is the force of infection function $\lambda(t) = \beta \frac{I(t)}{N_{pop}}$, where β is the probability of infection per unit time of contact, and the transition from infectious to removed γ is the inverse of the average duration of infectiousness.

However, there are many more states individuals can fall into (e.g., exposed, infected but asymptomatic, vaccinated, or isolating). The omission of those states and their influence on

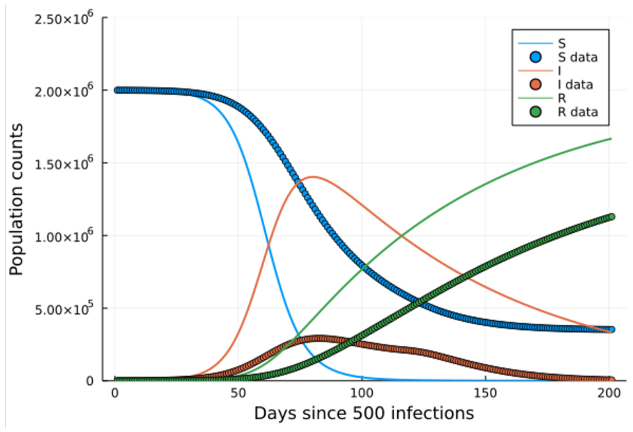


Fig. 2.1: Depiction of the discrepancy between the low-fidelity SIR model and the “true” model caused by the model error in the SIR model. Here the “true” model is the SIRQ (Susceptible-Infectious-Removed-Quarantined) model defined below without noise.

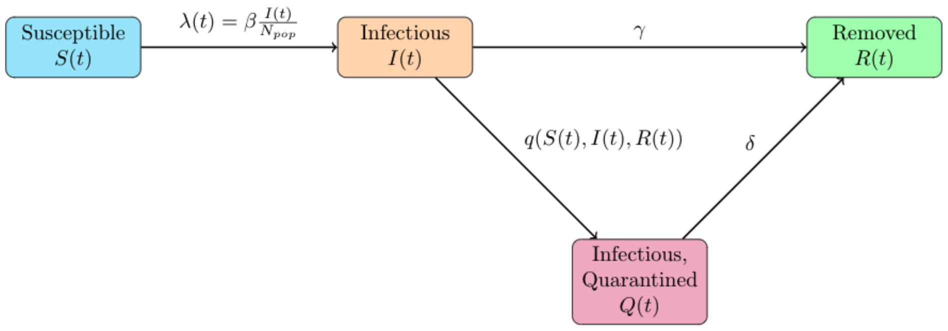


Fig. 2.2: Graph of the SIRQ compartmental disease model.

the susceptible, infectious, and removed populations leads to high model error, as is depicted in Figure 2.1, where the SIR model predictions are compared to data from a “true”, more complex model, where the model error in the SIR model leads to significant discrepancies with the data.

We reduce model error by enriching the SIR model with more information. For example, with every novel outbreak contact tracers provide the first line of defense in controlling the spread of infection by isolating known cases and quarantining the corresponding presumptive positive contacts. For ease in communication, we simply refer to this as quarantining the infectious population. We can intrusively correct the SIR model to include a quarantined compartment and form the so-called SIRQ model, but we do not know the form of the transition from infectious to quarantined. The quarantine rate is dependent on people’s individual risk tolerance and community guidelines, and it is likely a nonlinear function of the current state of the disease in the community. That is, it is not a constant in time, as transition rates are typically assumed to be in compartmental models of disease spread. Therefore, we will use data to inform the transition from infectious to quarantined. A graphical representation of the SIRQ model is shown in Figure 2.2. The transition from infectious to quarantined is the function of quarantine intervention $q(\cdot)$, and the transition from quarantined to removed δ is the inverse of the average time a person is in the quarantined compartment.

2.1. Noisy "True" Model. Ideally, our experimental observations would come from real data of a notional virus, but it is extremely hard to identify all of the sources of uncertainty in real-world observations. Notably, we would have to deal with underreporting [6, 10]. Therefore, we create a "true" model as a testbed to generate observations. The "true" model is a system of UDEs, which are differential equations defined in full or part by a universal approximator [15]—in this case, a NN to capture the unknown, nonlinear transition from infectious to quarantined. For our synthetic data the form of the NN is known and fixed. The "true" UDEs will be stochastic differential equations (SDEs) to represent the underlying uncertainty in the model parameters $\boldsymbol{\theta} = (\beta, \gamma, \delta)$ and the NN function approximation $q(\cdot)$. Although deterministic compartmental models assume a single value for the model parameters, they represent an aggregation of population statistics and may even vary in time. To represent this uncertainty, the model parameters and NN function approximation are defined as random variables. The resulting noisy "true" model is the set of four SDEs

$$\begin{aligned}
dS(t) &= -\beta \frac{I(t)}{N_{pop}} S(t)dt - \sigma_\beta \frac{I(t)}{N_{pop}} S(t) dW(t) \\
dI(t) &= \left(\beta \frac{I(t)}{N_{pop}} S(t) - \gamma I(t) - q(S(t), I(t), R(t)) \frac{I(t)}{N_{pop}}\right) dt \\
&\quad + \left(\sigma_\beta \frac{I(t)S(t)}{N_{pop}} - \sigma_\gamma I(t) - \sigma_q \frac{I(t)}{N_{pop}}\right) dW(t) \\
dR(t) &= (\gamma I(t) + \delta Q(t))\, dt + (\sigma_\gamma I(t) + \sigma_\delta Q(t))\, dW(t) \\
dQ(t) &= \left(q(S(t), I(t), R(t)) \frac{I(t)}{N_{pop}} - \delta Q(t)\right) dt + \left(\sigma_q \frac{I(t)}{N_{pop}} + \sigma_\delta Q(t)\right) dW(t),
\end{aligned} \tag{2.2}$$

where $W(t)$ denotes a Wiener process, $N_{pop} = S(t)+I(t)+R(t)+Q(t)$, $\beta = 0.15$, $\gamma = 0.013$, $\delta = 0.01$, $\sigma_\beta = 0.05$, $\sigma_\gamma = \sigma_\delta = 0.04$, $\sigma_q = 0.01$, and $q(S(t), I(t), R(t))$ is a NN function approximator with inputs $S(t)$, $I(t)$, and $R(t)$, two hidden layers with ten nodes per layer, and a single output.

In our numerical experiments, we assume the community starts with a total population about the size of New Mexico, $2,000,000$ people, where $I(0) = 500$ and $R(0) = Q(0) = 10$. We simulate the noisy "true" model for $N = 201$ days and sample the states daily. While we have data for the susceptible population, it is not available in real-world data, so we will remove the susceptible population from our observations. Therefore, our quantities of interest (QoIs) will be infectious, removed, and quarantined counts, and the noisy "true" observations $\boldsymbol{D}$ will be a $3N$ vector of infectious, recovered, and quarantined counts from day zero to day 200 with a timestep of one day. As discussed in [16], disease trends on the scale of a week or more are of most interest to decision makers. Additionally, fitting to noisy data without accounting for uncertainty in the observations can lead to overfitting. To address these considerations, we apply a moving average filter during preprocessing, and the resulting filtered observations are denoted $\bar{\boldsymbol{D}}$. We use the discrepancy between the filtered and noisy observations to estimate the uncertainty in the observations. This procedure is detailed in Appendix A.

2.2. Neural-Network-Corrected Model. Our enriched SIRQ model can also be described by a system of UDEs, but they are ODEs containing an NN to capture the unknown, nonlinear transition from infectious to quarantined. Our neural-network-corrected

SIRQ model is a closed system with a constant total population. The resulting UDEs are

$$\begin{aligned}
\frac{dS(t)}{dt} &= -\beta \frac{I(t)}{N_{pop}} S(t) \\
\frac{dI(t)}{dt} &= \beta \frac{I(t)}{N_{pop}} S(t) - \gamma I(t) - q(S(t), I(t), R(t); \boldsymbol{\Phi}) \frac{I(t)}{N_{pop}} \\
\frac{dR(t)}{dt} &= \gamma I(t) + \delta Q(t) \\
\frac{dQ(t)}{dt} &= q(S(t), I(t), R(t); \boldsymbol{\Phi}) \frac{I(t)}{N_{pop}} - \delta Q(t),
\end{aligned} \tag{2.3}$$

where $N_{pop} = S(t) + I(t) + R(t) + Q(t) = 2,000,000$, and the unknown NN parameters are $\boldsymbol{\Phi}$. The appropriate value for the initial conditions of the model was also considered. Given that the model is representative of more macroscopic, smooth trends, it is not expected to match the noisy data exactly. To this end, the initial condition was set to match the initial value of the filtered data. In the future, the initial values of the populations could be included in the calibration process.

The unknown model parameters are $\boldsymbol{\theta} = (\beta, \gamma, \delta)$. The number of elements in the NN parameter vector $\boldsymbol{\Phi}$ depends on the NN architecture. For this study, we have three inputs $S(t)$, $I(t)$, and $R(t)$ and a single output, but we will explore three different, densely connected formations of hidden layers: 2x2, 2x10, and 4x2. The 2x2 contains two hidden layers with two nodes per layer, and it represents a very small NN architecture with $J = 11$ parameters. The 2x10 contains two layers with ten nodes per layer, and it represents the unknown, true NN architecture with $J = 51$ parameters. The 4x2 contains four layers with two nodes per layer, and it represents a NN architecture with more complex relationships and with $J = 23$ parameters. This will allow us to study to what degree the depth and width of the NN impact validity. Note that the "true" noisy data is generated with a 2x10 NN, the architecture with the most parameters.

3. Calibration.

Decision makers take into account the trend in the infectious population when determining a threshold for action [18]. For example, an exponential increase might cause them to intervene with mask mandates or stay-at-home orders. To understand how features included in the calibration data impact the extrapolative capability of the corrected model predictions, we split the calibration observations into three groups of infectious behavior: linear, exponential, and quadratic increase. These three regimes represent three distinct phases of the disease and are depicted in Figure 3.1.

Calibration of the model and NN parameters is performed in a Bayesian context to incorporate information about parameter and observational uncertainties. The goal for calibration is to find the maximum a posteriori (MAP) estimate for the model parameters $\boldsymbol{\theta} = (\beta, \gamma, \delta)$ and NN parameters $\boldsymbol{\Phi}$. However, the nonlinearity of the NN leads to a complicated posterior distribution with many local minima. To obtain a characterization of the posterior and its many local minima, we run an ensemble of calibrations with 100 different initializations of $\boldsymbol{\Phi}$; resulting in 100 ensembles of the MAP estimate. A completely random sampling of NN parameters $\boldsymbol{\Phi}$ can lead to extreme numerical issues (e.g., stiffness or ill-conditions) causing the solver to fail and the optimization to error out. To address this, an initialization procedure was developed prior to MAP optimization to seed $\boldsymbol{\Phi}$ at reasonable, though still random, values, which is detailed in Appendix B.

After the initialization procedure, we run the MAP optimization. In the Bayesian context, the loss function is

$$loss_{\text{MAP}}(\boldsymbol{\theta}, \boldsymbol{\Phi} \mid \boldsymbol{D}_c) = -\log \pi_{like} - \log \pi_{prior}, \tag{3.1}$$

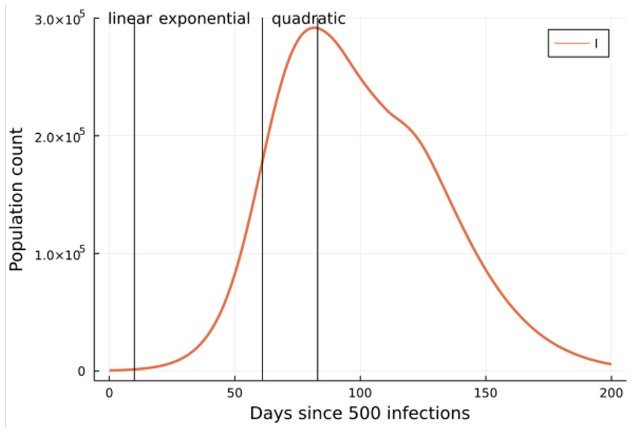


Fig. 3.1: Distinction of linear, exponential, and quadratic behavior in the infectious population.

which is proportional to the negative log posterior distribution. Here $\boldsymbol{D}_c \subseteq \boldsymbol{D}$ is a vector of the noisy observations for a given calibration scenario. The prior distribution is

$$\pi_{prior} = \left(\prod_{i=1}^{3} \mathcal{U}(0,1)\right)\left(\prod_{j=1}^{J} \mathcal{N}\left(0, 50^2\right)\right), \tag{3.2}$$

where J is the number of NN parameters. The prior on the NN parameters is a noninformative, minimalist prior, which Gelman et al. define as the least informed type of prior [5], because we do not have any constraining information. The prior on the model parameters is a maximum entropy prior because we know that the transition rates between compartments must be nonnegative and no greater than one. The likelihood distribution is

$$\pi_{like} = p\left(\boldsymbol{D}_c \mid \boldsymbol{\theta}, \boldsymbol{\Phi}\right) = \mathcal{N}\left(\boldsymbol{D}_c - Y_M(\boldsymbol{\theta}, \boldsymbol{\Phi}) \mid \boldsymbol{0}, \Sigma_\epsilon\right), \tag{3.3}$$

where $\Sigma_\epsilon = \operatorname{diag}\left((\boldsymbol{c}_1 + \boldsymbol{c}_2 \boldsymbol{D}_c)^2\right)$ is the covariance matrix of the estimated observational error as determined by the procedure in Appendix A. We acknowledge that it is very likely the observational uncertainty is not Gaussian; however, a more complex data distribution was out of scope for this project. We perform the training with an ADAM solver and $100,000$ epochs [9], and we return the values of $\boldsymbol{\theta}$ and $\boldsymbol{\Phi}$ that minimize $loss_{MAP}$ as $\boldsymbol{\theta}_{\mathrm{MAP}}$ and $\boldsymbol{\Phi}_{\mathrm{MAP}}$. After calibration, we are left with and ensemble of 100 values for $\boldsymbol{\theta}_{\mathrm{MAP}}$ and $\boldsymbol{\Phi}_{\mathrm{MAP}}$. However, some of these samples may have gotten stuck in a local minimum that is significantly inconsistent with the calibration data. Therefore, the ensembles undergo post-processing detailed in Appendix C to filter out such outliers before validation, and we are left with $M \leq 100$ ensembles.

4. Validation.

In general, validation checks that model predictions are consistent with observations from the true system. A good validation process should test the model against relevant and reliable data from the true system, such as testing a model beyond the calibration tests (i.e., for edge cases and extrapolations). The validation process should build confidence in the model's accuracy. However, there are several approaches for determining if model predictions and observations are consistent, and the best approach is often application-specific. We will focus on probabilistic validation metrics because we want to account for uncertainties in the modeled system. In this application, we are particularly interested in how long after the calibration timeframe the model can make extrapolative predictions. We investigate four validation metrics: the Mahalanobis distance, quantiles, the predictive assessment, and the reliability metric.

Before detailing the validation metrics, let us define a standard notation that will be used for all of the metrics. The time horizon included in a given validation test is $T = [t_0, t_N]$, where t_0 is the beginning of the calibration scenario, and t_N is the end of the simulated time horizon. The linear behavior begins at $t_0 = 0$ days, the exponential behavior begins at $t_0 = 11$ days, and the quadratic behavior begins at $t_0 = 62$ days. The number of timesteps in T is N_v, which is also the timespan since our time unit is a day. The full set of validation data is $\boldsymbol{D}_v \subseteq \boldsymbol{D}$, which is a $3N_v$ vector. The validation data at a single timestep is denoted by $\boldsymbol{D}_v(t)$, and it is a vector of the three QoIs. We denote a single QoI from the validation data as $\boldsymbol{D}_v(t, f)$, where $f \in (1, 2, 3)$ corresponding to infectious, removed, and quarantined populations. The neural-network-corrected model evaluated over the time horizon T is defined as $Y_M(\boldsymbol{\theta}, \boldsymbol{\Phi}; T)$, which is a $3N_v$ vector as well. The model prediction at a single timestep is denoted by $Y_M(\boldsymbol{\theta}, \boldsymbol{\Phi}; t)$, and it is a vector of the three QoIs. We denote a single QoI prediction as $Y_M(\boldsymbol{\theta}, \boldsymbol{\Phi}; t, f)$.

The Mahalanobis distance measures the average number of standard deviations the observations are from a Gaussian distribution centered at the model predictions:

$$\bar{d} = \frac{1}{3N_v}\sqrt{(\boldsymbol{D}_v - \boldsymbol{P})^T \Sigma_{\boldsymbol{P}}^{-1} (\boldsymbol{D}_v - \boldsymbol{P})}. \tag{4.1}$$

Here $\boldsymbol{P} = \frac{1}{M}\sum_{k=1}^{M} Y_M(\boldsymbol{\theta}^k_{\text{MAP}}, \boldsymbol{\Phi}^k_{\text{MAP}}; T)$ is the mean model predictions from M ensembles, and $\Sigma_{\boldsymbol{P}}$ is the ensemble covariance. A value of $\bar{d} \leq 2$, corresponding to a 95% confidence interval (CI), is generally accepted as valid [11]. We take the Mahalanobis distance at each time interval, successively including more observation times to see how long the model predictions are valid. This results in calculating Equation (4.1) with $t_N = t_0$, then $t_N = t_0 + 1$, and so on until $t_N = 200$, which is the maximum validation time, or $\bar{d} > 2$. We return m, which is the first time the Mahalanobis distance exceeds the tolerance.

Quantiles compare observations of the QoIs to the quantiles of the model predictions at a given time t, assuming our model distribution is unimodal. We detail the calculations in Algorithm 1, where α is the Bonferroni corrected tolerance corresponding to a 95% CI. We return q, which is the first time an observation is outside of the middle 95% CI.

The predictive assessment calculates the probability that a validation observation $\boldsymbol{D}_v(t, f)$ is an outcome of the model given the calibration data $\boldsymbol{D}_c$:

$$\begin{aligned} p(\boldsymbol{D}_v(t,f) \mid \boldsymbol{D}_c) = \int_{\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}} \Big(& p(\boldsymbol{D}_v(t,f) \mid \boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}) \\ & p(Y_M(\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}; t, f) \mid \boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}) \\ & p(\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}} \mid \boldsymbol{D}_c)\Big) d\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}, \end{aligned} \tag{4.2}$$

where $p(\boldsymbol{D}_v(t, f) \mid \boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}})$ is the likelihood that the validation observation is an outcome of the model, which is defined by the estimated observational error defined in Equation (A.2) as

$$\mathcal{N}\left(\boldsymbol{D}_v(t,f) - Y_M(\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}; t, f) \mid 0, (c_1(f) + c_2(f)\boldsymbol{D}_v(t,f))^2)\right). \tag{4.3}$$

Our predictive assessment is almost identical to the posterior predictive assessment [4], but we have an ensemble of $\boldsymbol{\theta}$ and $\boldsymbol{\Phi}$ values sampled at several local optima in the posterior space. Then, we compare this probability to the possible model predictions. In particular,

Algorithm 1 Quantile Validation

```
function QUANTILE_VALIDATION($t_0$, $M$, $t_N = 200$)
    $\alpha \leftarrow \frac{0.05}{3N_v}$
    $q \leftarrow -1$
    for $t$ in $t_0 : t_N$ do
        for $f$ in $1:3$ do
            $Y_{M\_ens} \leftarrow$ [] (empty array)
            for $k$ in $1:M$ do
                $Y_{M\_ens}$.append $(Y_M(\boldsymbol{\theta}_{\text{MAP},k}, \boldsymbol{\Phi}_{\text{MAP},k}; t, f))$
            end for
            $sorted \leftarrow \text{sort}(Y_{M\_ens})$
            $lower\_bound \leftarrow \text{quantile}(sorted, \alpha/2.0)$
            $upper\_bound \leftarrow \text{quantile}(sorted, 1-\alpha/2.0)$
            if $\boldsymbol{D}_v(t,f) < lower\_bound$ or $\boldsymbol{D}_v(t,f) > upped\_bound$ then
                $q \leftarrow t$
                BREAK
            end if
        end for
    end for
    return $q$
end function
```

we are interested in how much of the distribution corresponds to model predictions less likely than $\boldsymbol{D}_v(t,f)$, which is given by the γ-value.

$$\gamma = 1 - \int_S p(y^* \mid \boldsymbol{D}_c) dy^*, \tag{4.4}$$

where $S = \{y^* : p(y^* \mid \boldsymbol{D}_c) \geq p(\boldsymbol{D}_v(t,f) \mid \boldsymbol{D}_c)\}$ is the β-highest probability density credibility region [12]. If $\gamma < \frac{0.05}{3N_v}$, the model is generally deemed invalid for predicting the true system. We go through $\forall \boldsymbol{D}_v(t,f) \in \boldsymbol{D}_v$ chronologically and return p, which is the first time the β-highest probability density credibility region is less than $\frac{0.05}{3N_v}$.

The instantaneous reliability metric provides the degree of agreement between a validation observation and model predictions for a given QoI f and instance in time t:

$$r(t,f) = p\left(\left|Y_M\left(\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{\Phi}_{\text{MAP}}; t, f\right) - \boldsymbol{D}_v(t,f)\right| < \epsilon(t,f)\right) = \frac{1}{M}\sum_{k=1}^{M} I_k(t,f), \tag{4.5}$$

where

$$I_k(t,f) = \begin{cases} 1 & \text{if } d_k(t,f) < \epsilon(t,f) \\ 0 & \text{otherwise} \end{cases}, \tag{4.6}$$

$$d_k(t,f) = \left|Y_M\left(\boldsymbol{\theta}^k_{\text{MAP}}, \boldsymbol{\Phi}^k_{\text{MAP}}; t, f\right) - \boldsymbol{D}_v(t,f)\right|, \tag{4.7}$$

and $\epsilon(t,f) = 2\left(c_1(f) + c_2(f)\boldsymbol{D}_v(t,f)\right)$ is the discrepancy threshold, which corresponds to a 95% CI around the observational error. If we had replicate observations, that uncertainty could be incorporated as detailed in [1], but without replicates we use the estimated observational error to select $\epsilon(t,f)$. We go through $\forall f \in (1,2,3)$ for each t and $\forall t \in N_v$

chronologically and check $r(t,f) < \tau$, where $\tau = 0.9$ is the tolerance interval. We return r, which is the first time $r(t,f) < \tau$, or when the probability the model is within two standard deviations of the data falls below τ.

While these four metrics are all probabilistic, they each uniquely account for uncertainties and make different assumptions. The Mahalanobis distance can only account for uncertainty in the observations or the predictions but not both, and it assumes that the uncertainty is a Gaussian distribution. Quantiles only accounts for uncertainty in the predictions, and it assumes the uncertainty is unimodal. The predictive assessment captures uncertainty in the observations and the predictions, and it does not make assumptions about the uncertainty distributions. However, in practice the observational error is assumed to be a probability density function that can be easily sampled. The reliability metric also captures uncertainty in the observations and the predictions, but without replicate observations the observational uncertainty must define the discrepancy threshold representing the acceptable prediction error [1]. Consequently, the reliability metric requires the user to select a few application-specific parameters, the appropriate values of which can be challenging to know *a priori*, while the other three metrics require the user to input a CI.

5. Results. We ran numerical experiments for every combination of NN architecture (2x2, 2x10, and 4x2) and calibration scenario (linear, exponential, and quadratic) for two replicates of the noisy "true" model, resulting in 18 calibration campaigns. Then we calculated the time when each validation metric failed for the 18 simulations. While we would not have replicate data for real-world observations, we include replicates to show how disparate two samples can be when the observational uncertainty is more complex than the commonly assumed additive-Gaussian uncertainty.

Results from the first sample are depicted in Figure 5.1, Figure 5.3, and Figure 5.5, while results from the second sample are depicted in Figure 5.2, Figure 5.4, and Figure 5.6. The first sample's removed population data at 200 days after 500 infections is about two times as large as the second sample's removed population. Similarly, the first sample's quarantined population peaks at around $900,000$ people then steadily decreases, while the second sample's quarantined population peaks at around $600,000$ people and gradually decreases. The infectious population data from the two samples is analogous, which is not surprising due to the relatively small amount of observational error propagated to the infectious population in comparison to the removed and quarantined populations (see Figure A.1).

The calibration data was very influential in the extrapolative capability of the neural-network-corrected model. Results from the linear behavior calibration scenario for two samples of the noisy "true" model are shown in Figure 5.1 and Figure 5.2. The linear calibration scenario was uninformative for developing an extrapolative neural-network-corrected model, and there were a couple of reasons why. First, we only had 10 timesteps of calibration data, which is a very small amount of data for calibrating a data-driven correction, especially a NN with many parameters. Second, the little calibration data we had was not descriptive causing non-identifiability. Several NN parameter combinations fit the calibration data and assumed the linear trend would continue, which quickly resulted in a large discrepancy beyond the linear regime.

Results from the exponential behavior calibration scenario for two samples of the noisy "true" model are shown in Figure 5.3 and Figure 5.4. Given the exponential calibration data, the neural-network-corrected model could usually extrapolate for a few days beyond the calibration time horizon, but it struggled to capture the overall trend in the removed population.

Results from the quadratic behavior calibration scenario for two samples of the noisy "true" model are shown in Figure 5.5 and Figure 5.6. The neural-network-corrected model failed to fit the quadratic calibration data; this is presumably due to the increasing amount of noise in the removed population throughout time. Even the predictive assessment failed during the calibration time horizon, which indicates there is a discrepancy between our estimated observational error distribution and the true observational error. The predictive assessment marginalizes out the estimated observational error when calculating the probability that an observation is an outcome of the calibrated model; assuming the calibration converged, the predictive assessment failing on calibration data indicates that the observations were outside the estimated observational error distribution.

Overall, the NN architecture did not have a large impact on model predictions or, consequently, validation. We hypothesize that the 2x2 NN architecture is a sufficient approximation for the true underlying 2x10 NN given that the model predictions are almost indistinguishable.

In the context of this problem, we want a validation metric to determine if the corrected model can make reliable predictions while adequately accounting for uncertainties in the model and data. The Mahalanobis distance and quantiles are not ideal for this application because they did not account for uncertainty in the data, and they failed within the calibration time horizon for every calibration scenario. The Bayesian calibration accounted for observational error in Equation (3.3), which allowed the model predictions some flexibility in matching the calibration data. Then that flexibility was not transmitted to the Mahalanobis distance or quantiles, and the metrics failed due to them not accounting for observational error. The predictive assessment and reliability metric were both able to account for observational error, but the reliability metric failed in calibration for the exponential and quadratic calibration regimes, which was surprising since the observational error distribution used in the likelihood was also used to inform the discrepancy threshold $\epsilon(t, f)$. The predictive assessment tracked with intuition, and it deemed the corrected model invalid when the CI around the predictions started deviating from the data significantly. However, because the predictive assessment favors scenarios where the data is high probability in the posterior predictive space, a model can be valid with a strong bias in the mean but very large uncertainty. This may be undesirable, and a tolerable degree of uncertainty in model predictions should be considered for a given application problem.

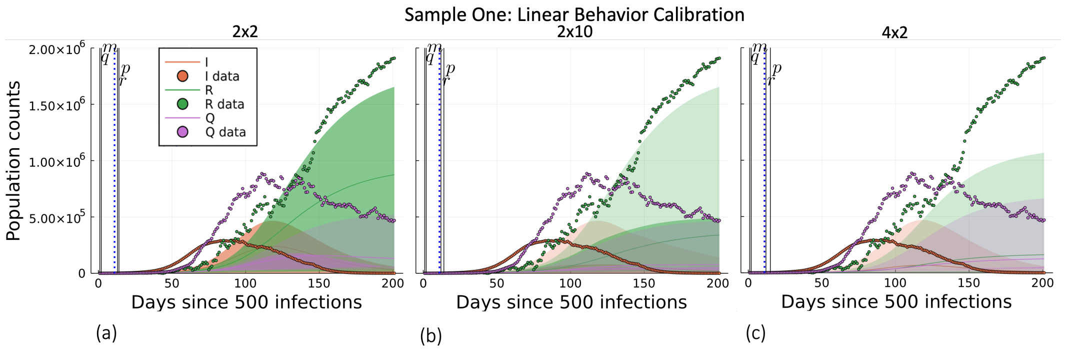


Fig. 5.1: Neural-network-corrected model mean predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to filtered noisy "true" model data (dots) for the linear calibration scenario. The vertical blue dashed line delineates the end of the linear calibration time horizon, and the vertical black lines represent when each validation metric fails. (m: Mahalanobis distance, q: quantiles, p: predictive assessment, and r: reliability)

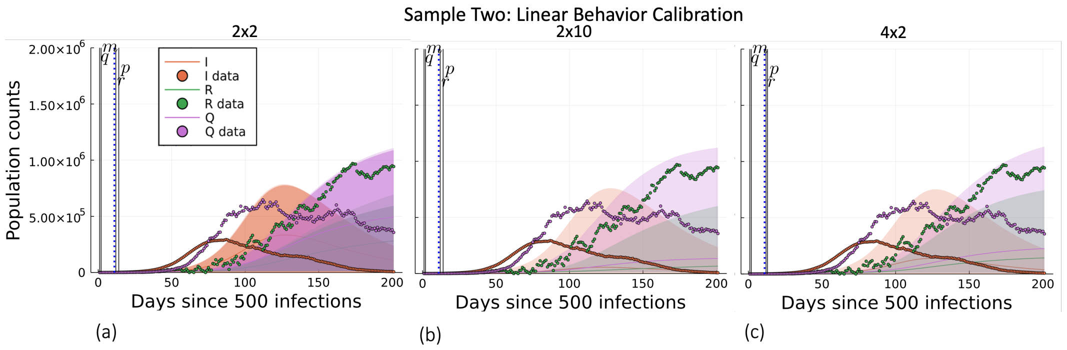


Fig. 5.2: Model predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to observations (dots) for the quadratic calibration scenario. The blue dashed line shows the end of the calibration time horizon, and the black lines show when validation metrics fail.

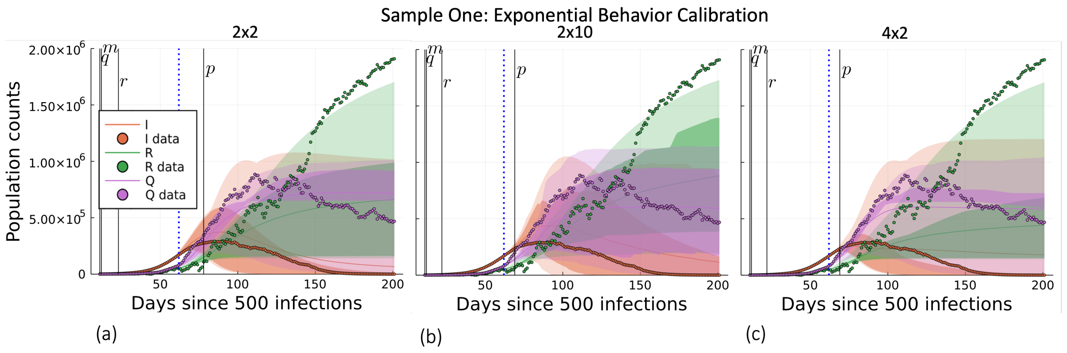


Fig. 5.3: Model predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to observations (dots) for the quadratic calibration scenario. The blue dashed line shows the end of the calibration time horizon, and the black lines show when validation metrics fail.

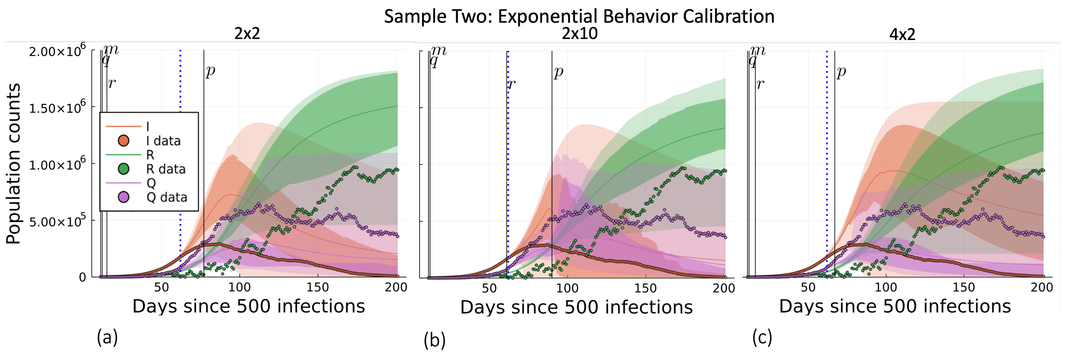


Fig. 5.4: Model predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to observations (dots) for the quadratic calibration scenario. The blue dashed line shows the end of the calibration time horizon, and the black lines show when validation metrics fail.

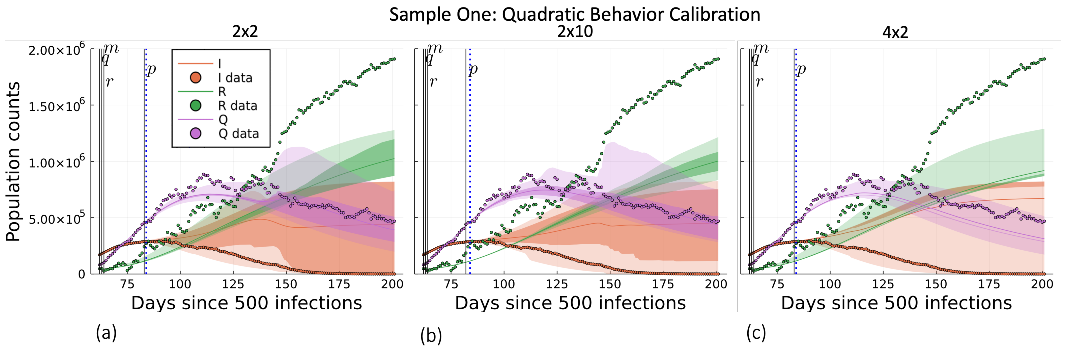


Fig. 5.5: Model predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to observations (dots) for the quadratic calibration scenario. The blue dashed line shows the end of the calibration time horizon, and the black lines show when validation metrics fail.

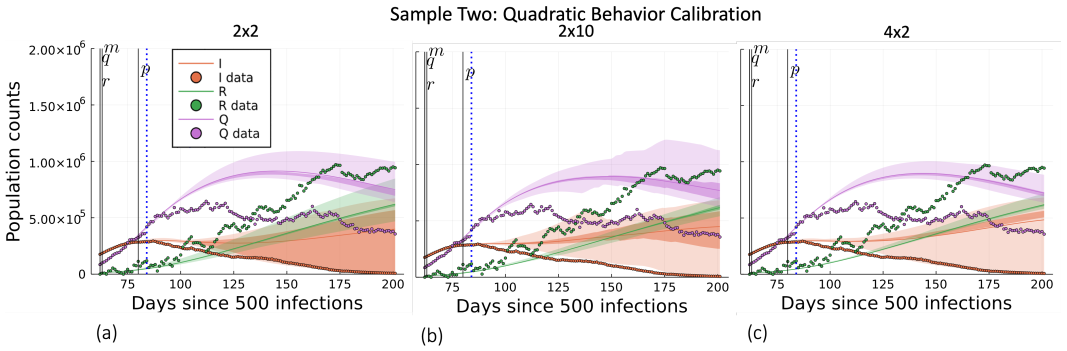


Fig. 5.6: Model predictions (solid curves) plotted with 50% and 95% CIs (shaded regions) compared to observations (dots) for the quadratic calibration scenario. The blue dashed line shows the end of the calibration time horizon, and the black lines show when validation metrics fail.

6. Conclusions. Embedding a NN into the SIRQ model reduced model error while retaining the interpretability of the compartmental model, allowing the neural-network-corrected model to make extrapolative predictions in time. While the corrected model did not perfectly reproduce the observations, given descriptive calibration data, it could predict infectious, removed, and quarantined trends days into the future. Overall, the NN architecture did not have a large impact on model predictions or, consequently, validation. Quantitative, probabilistic validation was used to determine when discrepancies in the model predictions and observations were too large to confidently use the model in place of observations. Besides determining the time horizon post-calibration that the model can be reliably used for prediction, the validation procedures used here can inform how often a model needs to be updated with new observation data to remain predictive. For instance, in this case the model was only valid for a few days into the future, so it should be recalibrated every two to three days. On the other hand, for models with longer time horizons of validity post-calibration, data can be incorporated less frequently.

In the future we would like to further explore how the calibration data influences model predictions by mixing the calibration regimes. Given 50 days of calibration data, is it better to have more quadratic, exponential, or linear behavior? It would be interesting to compare the performance of the neural-network-corrected model to a standard SIR model. We would like to extend this framework to embed a Bayesian neural network (BNN) into the model instead of a standard NN. Then calibration would return a true posterior distribution over the model parameters and the BNN parameters. We would also like to explore more sophisticated approaches for estimating the observational error (i.e., estimating the variance with ACCRUE [2] or calibrating the variance constants [16]).

7. Acknowledgments. This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. LDRD #222355. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government

Appendix A. Preprocessing. We use a moving average filter to reduce the short-term noise fluctuations and highlight the long-term trends in the observations from the noisy "true" model. Each noisy observation $\boldsymbol{D}(t,f) \in \boldsymbol{D}$, where $t \in [0, 200]$ is the time and $f \in (1,2,3)$ is the QoI, is transformed to the mean of a window of neighbors. We selected a window of size nine, which results in a lower bound on the time neighborhood of $l = \max(t-4, 0)$ and an upper bound on the time neighborhood of $u = \min(t+4, 200)$. The resulting smoothed observations are

$$\bar{\boldsymbol{D}}(t,f) = \frac{1}{W}\sum_{n=l}^{u} \boldsymbol{D}(t+n,f) \quad \forall f \in (1,2,3), \tag{A.1}$$

where W is the size of the time neighborhood around $\boldsymbol{D}(t,f)$. A comparison of the noisy observations and the smoothed observations is shown in Figure A.1. The noisy and filtered infectious observations are almost indistinguishable because the noise in the infectious population is an order of magnitude less than the noise in the recovered or quarantined population.

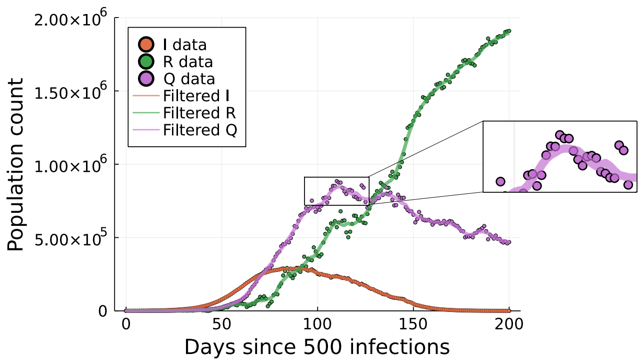


Fig. A.1: A comparison of the noisy observations (dots) and filtered observations (thick lines).

We estimate the distribution of the observational error on the QoIs I, R, and Q by examining the error we filtered out with Equation (A.1), so $\varepsilon(t,f) = \boldsymbol{D}(t,f) - \bar{\boldsymbol{D}}(t,f)$. Judging by Figure A.2(a), $\varepsilon(\cdot)$ is time dependent and roughly symmetric about zero. We hypothesize that $\varepsilon(t, f)$ depends on the size of the QoI at time t; Figure A.2(b) shows that the absolute value of $\varepsilon(\cdot)$ for the quarantined population scales with the noisy quarantined population count. Therefore, we expect the variance of observational error to contain some constant multiplied by the noisy data. Modeling the observational error as a constant multiple of the QoI was not sufficient to completely encapsulate $\varepsilon(\cdot)$, though, so an additional term was added to the variance.

We estimate the observational error on each QoI to be

$$\mathcal{N}\left(0, \left(c_1(f) + c_2(f)\boldsymbol{D}(t, f)\right)^2\right), \tag{A.2}$$

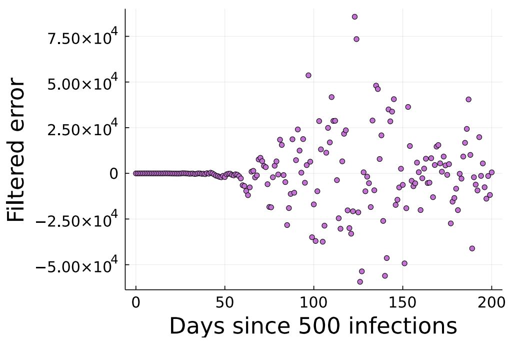
where $c_2(f)$ is the 90th percentile of the normalized absolute error's distribution, and $c_1(f)$ is the 90th percentile distance a noisy calibration observation is outside of a 95% CI around the smoothed observation:

$$\begin{aligned} c_1(f) = \text{quantile}\Bigg(\min\Big\{ &\left|\boldsymbol{D}(t, f) - (\bar{\boldsymbol{D}}(t, f) + 2c_2(f)\bar{\boldsymbol{D}}(t, f))\right|, \\ &\left|\boldsymbol{D}(t, f) - (\bar{\boldsymbol{D}}(t, f) - 2c_2(f)\bar{\boldsymbol{D}}(t, f))\right| \Big\}, 0.9\Bigg), \end{aligned} \tag{A.3}$$

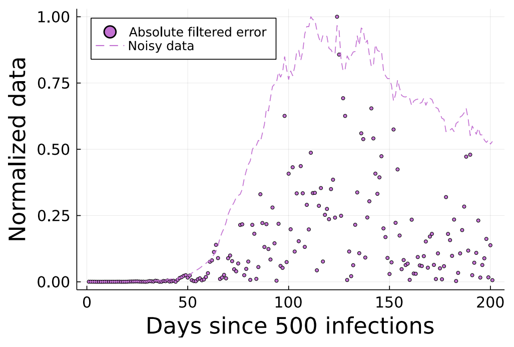
where $t = (0, 1, \ldots, 200)$. The conglomerate observational error distribution from all of the QoIs is

$$\epsilon \sim \mathcal{N}\left(\boldsymbol{0}, \operatorname{diag}\left((\boldsymbol{c}_1 + \boldsymbol{c}_2\boldsymbol{D})^2\right)\right), \tag{A.4}$$

where $\boldsymbol{D}$ is $3N$ vector of noisy observations because there are N observations for each of the three QoIs. The $3N$ multiplicative trend vector is $\boldsymbol{c}_2$, and it can be constructed by repeating $c_2(1)$ for the first N elements, placing $c_2(2)$ for the N through $2N - 1$ elements, and duplicating $c_2(3)$ for the last N elements. Similarly, the $3N$ additive trend vector is $\boldsymbol{c}_1$, and it can be constructed using $c_1(1)$, $c_1(2)$, and $c_1(3)$. The estimated observational uncertainty about the filtered observations is depicted in Figure A.3. Note, there is an uncertainty bound around the infectious population; however, it is too small to see at this scale. Additionally, it is very likely the observational uncertainty is not Gaussian, but a more complex data distribution was out of scope for this project.



(a)



(b)

Fig. A.2: a) The filtered error from observations of the quarantined population. b) Comparison of the trend in observations of the quarantined population (dashed curve) to the filtered absolute error (dots).

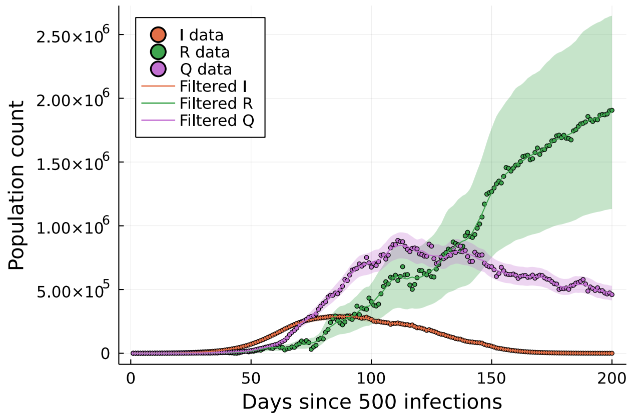


Fig. A.3: Filtered observations (solid curves) plotted with 95% CIs (shaded regions) compared to noisy "true" observations (dots).

Appendix B. Initial Regression. Beginning the MAP optimization with completely random NN parameters $\mathbf{\Phi}$ causes numerical issues for the model and thus the optimization. Our solution is a four part initialization procedure, which gets us in a reasonable physical regime for the MAP optimization later and mitigates numerical issues. First, we initially train the NN to match a constant value. We calibrate the SIRQ model with a constant rate parameter for the transition from infectious to quarantined rather than a time-varying function for q. The resulting ODEs are

$$\begin{aligned} \frac{dS(t)}{dt} &= -\frac{\beta I(t)S(t)}{N_{pop}} \\ \frac{dI(t)}{dt} &= \frac{\beta I(t)S(t)}{N_{pop}} - \gamma I(t) - q_{const}\frac{I(t)}{N_{pop}} \\ \frac{dR(t)}{dt} &= \gamma I(t) + \delta Q(t) \\ \frac{dQ(t)}{dt} &= q_{const}\frac{I(t)}{N_{pop}} - \delta Q(t), \end{aligned} \tag{B.1}$$

which are Equation (2.3) with q_{const} substituted for the NN output $q(\cdot)$. This optimization results in minimizing the mean-squared error (MSE) loss function

$$loss_1(\boldsymbol{\theta}, q_{const} \mid \bar{\boldsymbol{D}}_c) = -\log \mathcal{N}\left(\log(\bar{\boldsymbol{D}}_c) - \log(Y_M(\boldsymbol{\theta}, q_{const}; T_c)) \mid \mathbf{0}, 0.5\mathbb{I}\right), \tag{B.2}$$

where $T_c = [t_0, t_{N_c}]$ is the calibration time horizon, $\bar{\boldsymbol{D}}_c \subseteq \bar{\boldsymbol{D}}$, and $\bar{\boldsymbol{D}}_c$ is a vector of the filtered observations from the calibration scenario. The corresponding vector of model predictions given the model parameters and q_{const} is $Y_M(\boldsymbol{\theta}, q_{const}; T_c)$, and $\mathbb{I}$ is the identity matrix. Initial optimizations are performed with respect to the logarithm of the state variables based on previous experience that doing so leads to much more rapid convergence to optimum values. The improved performance may have to do with the fact that the QoIs can vary by orders of magnitude, and rescaling helps mitigate larger values dominating the calibration. However, the uncertainty in the log-transformed values are unknown, so the MSE is used, and the smoothed data is used to mitigate the chance of fitting to noise in the initial regression process. We use DiffEqFlux in Julia to perform the regression with an BFGS solver and 10,000 epochs [14], and we store the values of $\boldsymbol{\theta}$ and q_{const} that minimize $loss_1$ as $\boldsymbol{\theta}^{(1)}_{\min}$ and $q^{(1)}_{const,\min}$.

Next, we calibrate the NN parameters $\mathbf{\Phi}$ to minimize the MSE between the NN output and $q^{(1)}_{const,\min}$:

$$loss_2(\mathbf{\Phi} \mid q^{(1)}_{const,\min}) = \frac{1}{N_c} \sum_{i=t_0}^{t_{N_c}} \left(q(S(i), I(i), R(i); \mathbf{\Phi}) - q^{(1)}_{const,\min} \right), \tag{B.3}$$

where N_c is the number of days in the calibration scenario. We perform the regression with an BFGS solver and $10,000$ epochs, and we store the values of $\mathbf{\Phi}$ that minimize $loss_2$ as $\mathbf{\Phi}^{(2)}_{\min}$. Similar to the q_{const} calibration against filtered calibration data, we provide a warm start to the MAP optimization for $\boldsymbol{\theta}$ and $\mathbf{\Phi}$ by first running a calibration with the log-transformed QoIs. This helps seed the MAP optimization in a favorable location so that it converges with a finite, but still large number of optimization steps. We optimize the log model predictions starting with $\boldsymbol{\theta} = \boldsymbol{\theta}^{(1)}_{\min}$ and $\mathbf{\Phi} = \mathbf{\Phi}^{(2)}_{\min}$. This results in the loss function

$$loss_3(\boldsymbol{\theta}, \mathbf{\Phi} \mid \bar{\boldsymbol{D}}_c) = -\log \pi_{like} - \log \pi_{prior}, \tag{B.4}$$

which is proportional to the negative log posterior distribution. The prior distribution is given in Equation (3.2). The likelihood distribution is

$$\pi_{like} = p\left(\bar{\boldsymbol{D}}_c \mid \boldsymbol{\theta}, \mathbf{\Phi}\right) = \mathcal{N}\left(\log(\bar{\boldsymbol{D}}_c) - \log(Y_M(\boldsymbol{\theta}, \mathbf{\Phi}); T_c) \mid \mathbf{0}, 0.5\mathbb{I}\right), \tag{B.5}$$

where $Y_M(\boldsymbol{\theta}, \mathbf{\Phi}; T_c)$ is a vector of model prediction given the model parameters and NN parameters. We perform the regression with an ADAM solver and $100,000$ epochs, and we return the values of $\boldsymbol{\theta}$ and $\mathbf{\Phi}$ that minimize $loss_3$ as $\boldsymbol{\theta}^{(3)}_{\min}$ and $\mathbf{\Phi}^{(3)}_{\min}$. Note, $\boldsymbol{\theta}^{(3)}_{\min}$ and $\mathbf{\Phi}^{(3)}_{\min}$ are the values that returned the smallest $loss_3$ over all $100,000$ epochs not necessarily the $\boldsymbol{\theta}$ and $\mathbf{\Phi}$ values after the $100,000^{\text{th}}$ epoch. After the first optimization of $loss_3$, we perform a second optimization of $loss_3$ with initial parameters as the return of the previous optimization in an attempt to escape a local minimum that the optimization could get stuck in if we simply doubled the number of epochs and ran a single optimization with $loss_3$.

Appendix C. Post-processing. After calibration, we have an ensemble of 100 $\boldsymbol{\theta}_{\text{MAP}}$ and $\boldsymbol{\phi}_{\text{MAP}}$ vectors, but some of these samples got trapped in a local minimum far away from the MAP estimate and ultimately their calibration was unsuccessful as evidenced by a clear deviation in model predictions from the calibration data. When the calibration fails, the MSE between the calibration observations and the model predictions is relatively larger than the other samples' MSE. We filter these outliers with the Interquartile Rule; if the sample's MSE exceeds the heuristic threshold of

$$Q_3 + 1.5(Q_3 - Q_1), \tag{C.1}$$

where Q_3 is the 0.75 quantile and Q_1 is the 0.25 quantile of the ensemble MSEs, we classify the sample as an outlier and remove it. This results in $M \leq 100$ ensembles, where M is 100 less the outliers, which averaged $M = 84$.

REFERENCES

[1] D. Ao, Z. Hu, and S. Mahadevan, *Dynamics model validation using time-domain metrics*, Journal of Verification, Validation and Uncertainty Quantification, 2 (2017).

[2] E. Camporeale and A. Carè, *Accrue: Accurate and reliable uncertainty estimate in deterministic models*, International Journal for Uncertainty Quantification, 11 (2021).

[3] R. Dandekar, C. Rackauckas, and G. Barbastathis, *A machine learning-aided global diagnostic and comparative tool to assess effect of quarantine control in covid-19 spread*, Patterns, 1 (2020), p. 100145.

[4] A. Gelman, X.-L. Meng, and H. Stern, *Posterior predictive assessment of model fitness via realized discrepancies*, Statistica sinica, (1996), pp. 733–760.

[5] A. Gelman, D. Simpson, and M. Betancourt, *The prior can often only be understood in the context of the likelihood*, Entropy, 19 (2017), p. 555.

[6] H. W. Hethcote, *The mathematics of infectious diseases*, SIAM review, 42 (2000), pp. 599–653.

[7] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, *Computer model calibration using high-dimensional output*, 103 (2008), pp. 570–583.

[8] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, 63 (2001), pp. 425–464.

[9] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[10] H. Lau, T. Khosrawipour, P. Kocbach, H. Ichii, J. Bania, and V. Khosrawipour, *Evaluating the massive underreporting and undertesting of covid-19 cases in multiple global epicenters*, Pulmonology, 27 (2021), pp. 110–115.

[11] K. A. Maupin, L. P. Swiler, and N. W. Porter, *Validation metrics for deterministic and probabilistic data*, Journal of Verification, Validation and Uncertainty Quantification, 3 (2018).

[12] R. E. Morrison, T. A. Oliver, and R. D. Moser, *Representing model inadequacy: A stochastic operator approach*, SIAM/ASA Journal on Uncertainty Quantification, 6 (2018), pp. 457–496.

[13] T. A. Oliver, G. Terejanu, C. S. Simmons, and R. D. Moser, *Validating predictions of unobserved quantities*, 283 (2015), pp. 1310–1335.

[14] C. Rackauckas, M. Innes, Y. Ma, J. Bettencourt, L. White, and V. Dixit, *Diffeqflux.jl - A julia library for neural differential equations*, CoRR, abs/1902.02376 (2019).

[15] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, *Universal differential equations for scientific machine learning*, arXiv preprint arXiv:2001.04385, (2020).

[16] C. Safta, J. Ray, E. Acquesta, T. A. Catanach, K. S. Chowdhary, B. Debusschere, E. Galvan, G. Geraci, M. Khalil, and T. Portone, *Characterization of partially observed epidemics-application to covid-19*, tech. rep., Sandia National Lab.(SNL-CA), Livermore, CA (United States); Sandia National . . . , 2020.

[17] M. Tracy, M. Cerdá, and K. M. Keyes, *Agent-based modeling in public health: current applications and future directions*, Annual review of public health, 39 (2018), p. 77.

[18] L. Tribe and R. Smith, *Modelling global outbreaks and proliferation of covid-19*, May 2020.

THE SCHWARZ ALTERNATING METHOD FOR THE SEAMLESS COUPLING OF NONLINEAR REDUCED ORDER MODELS AND FULL ORDER MODELS

JOSHUA BARNETT*, IRINA TEZAUR , AND ALEJANDRO MOTA†

Abstract. Projection-based model order reduction allows for the parsimonious representation of full order models (FOMs), typically obtained through the discretization of a set of partial differential equations (PDEs) using conventional techniques (e.g., finite element, finite volume, finite difference methods) where the discretization may contain a very large number of degrees of freedom. As a consequence of this more compact representation, the resulting projection-based reduced order models (ROMs) can achieve considerable computational speedups, which are especially useful in real-time or multi-query analyses. One known deficiency of projection-based ROMs is that they can suffer from a lack of robustness, stability and accuracy, especially in the predictive regime, which ultimately limits their useful application. Another research gap that has prevented the widespread adoption of ROMs within the modeling and simulation community is the lack of theoretical and algorithmic foundations necessary for the "plug-and-play" integration of these models into existing multi-scale and multi-physics frameworks. This paper describes a new methodology that has the potential to address both of the aforementioned deficiencies by coupling projection-based ROMs with each other as well as with conventional FOMs by means of the Schwarz alternating method [41]. Leveraging recent work that adapted the Schwarz alternating method to enable consistent and concurrent multi-scale coupling of finite element FOMs in solid mechanics [35, 36], we present a new extension of the Schwarz framework that enables FOM-ROM and ROM-ROM coupling, following a domain decomposition of the physical geometry on which a PDE is posed. In order to maintain efficiency and achieve computation speedups, we employ hyper-reduction via the Energy-Conserving Sampling and Weighting (ECSW) approach [9]. We evaluate the proposed coupling approach in the reproductive as well as in the predictive regime on a canonical test case that involves the dynamic propagation of a traveling wave in a nonlinear hyper-elastic material.

1. Introduction.

Projection-based model order reduction is a promising data-driven strategy for reducing the computational complexity of numerical simulations by restricting the search of the solution to a low-dimensional space spanned by a reduced basis constructed from a limited number of high-fidelity simulations and/or physical experiments/observations. While recent years have seen extensive investments in the development of projection-based reduced order models (ROMs) and other data-driven models, these models are known to suffer from a lack of robustness, stability and accuracy, especially in the predictive regime. Moreover, a unified and rigorous theory for integrating these models in a "plug-and-play" fashion into existing multi-scale and multi-physics coupling frameworks (e.g., the Department of Energy's Energy Exascale Earth System Model (E3SM) [12]) is lacking at the present time.

This paper presents and evaluates an approach aimed at addressing both of the aforementioned shortcomings by advancing the Schwarz alternating method [41] as a mechanism for coupling together a variety of models, including full order finite element models and projection-based ROMs constructed using the proper orthogonal decomposition (POD)/Galerkin method. The Schwarz alternating method is based on the simple idea that if the solution to a partial differential equation (PDE) is known in two or more regularly-shaped domains comprising a more complex domain, these local solutions can be used to iteratively build a solution on the more complex domain. Our coupling approach thus consists of several ingredients, namely: (1) the decomposition of the physical domain into overlapping or non-overlapping subdomains, which can be discretized in space by disparate meshes and in time by different time-integration schemes with different time-steps, (2) the definition of transmission boundary conditions on subdomain boundaries, and (3) the itera-

*Department of Mechanical Engineering, Stanford University, jb0@stanford.edu
†Sandia National Laboratories, ikalash@sandia.gov

tive solution of a sequence of subdomain problems in which information propagates between the subdomains through the aforementioned transmission conditions. Without loss of generality, we develop and prototype the method in the context of a generic transient dynamic solid mechanics problem, defined by an arbitrary constitutive model embedded within the governing PDEs. Following an overlapping or non-overlapping domain decomposition (DD) of the underlying geometry, we use the POD/Galerkin method with Energy-Conserving Sampling and Weighting (ECSW)-based hyper-reduction [9] to reduce the problem in one or more subdomains. We then employ the Schwarz alternating method to couple the resulting subdomain ROMs with each other or with finite element-based full order models (FOMs) in neighboring subdomains. We demonstrate that a careful formulation and implementation of the transmission conditions in the ROMs being coupled is essential to the coupling method.

The methodology described in this paper is related to several existing coupling approaches developed in recent years. First, while this work is a direct extension of the recently-developed Schwarz-based methodology for concurrent multi-scale FOM-FOM coupling in solid mechanics [35, 36], it includes a number of advancements, including the extension of the coupling framework to: (1) FOM-ROM coupling, (2) ROM-ROM coupling, and (3) non-overlapping subdomains. Among the earliest authors to develop an iterative Schwarz-based DD approach for coupling FOMs with ROMs are Buffoni *et al.* [3]. The approach in [3] is unlike ours in that attention is restricted to Galerkin-free POD ROMs, developed for the Laplace equation and the compressible Euler equations. Other authors to consider Galerkin-free FOM-ROM and ROM-ROM couplings are Cinquegrana *et al.* [4] and Bergmann *et al.* [2]. The former approach [4] focuses on overlapping DD in the context of a Schwarz-like iteration scheme, but, unlike our approach, requires matching meshes at the subdomain interfaces. The latter approach [2], termed zonal Galerkin-free POD, defines a minimization problem to minimize the difference between the POD reconstruction and its corresponding FOM solution in the overlapping region between a ROM and a FOM domain, and is developed/investigated in the context of an unsteady flow and aerodynamic shape optimization. While the method developed [2] is not based on the Schwarz alternating formulation, the recent related work [22] by Iollo *et al.* demonstrates that a similar optimization-based coupling scheme is equivalent to an overlapping alternating Schwarz iteration for the case of linear elliptic PDE. A true POD-Greedy/Galerkin non-overlapping Schwarz method for the coupling of projection-based ROMs developed for the specific case of symmetric elliptic PDEs is presented by Maier *et al.* in [34].

While the focus herein is restricted to projection-based ROMs, it is worth noting that the Schwarz alternating method has recently been extended to the case of coupling Physics-Informed Neural Networks (PINNs) to each other following a DD in [28, 29]. The methods proposed in these works, termed D3M [28] and DeepDDM [29], inherit the benefits of DD-based ROM-ROM couplings, but are developed primarily for the purpose of improving the efficiency of the neural network training process and reducing the risk of overfitting, both of which are due to the global nature of the neural network “basis functions”.

We end our literature overview by remarking that a number of non-Schwarz-based ROM-ROM and/or FOM-ROM coupling methods have been developed in recent years, including [1, 46, 15, 20, 27, 32, 33, 5, 6, 24, 23, 39, 8, 43, 40, 18]. The majority of these approaches are based on Lagrange multiplier or flux matching coupling formulations, and focus on either simple linear elliptic PDEs or fluid problems. We omit a detailed assessment of these references from this paper for the sake of brevity.

The remainder of this paper is organized as follows. In Section 2, we provide the variational formulation of the generic solid dynamics problem considered herein, and describe its spatio-temporal discretization. Section 3 details our nonlinear model reduction methodology for this problem, which relies on the POD/Galerkin approach for model reduction and the ECSW method [9] for hyper-reduction. We describe the Schwarz alternating method for FOM-FOM, FOM-ROM and ROM-ROM coupling in Section 4. In Section 5, we evaluate the performance of the proposed Schwarz-based coupling methodology on a problem involving dynamic wave propagation in a one-dimensional (1D) hyper-elastic bar whose material properties are described by the nonlinear Henky constitutive model, characterized by a logarithmic strain tensor [14]. We conclude this paper with a summary and a discussion of some future research directions (Section 6).

2. Solid mechanics problem formulation. Consider the Euler-Lagrange equations for a generic dynamic solid mechanics problem in its strong form:

$$\operatorname{Div} \boldsymbol{P} + \rho_0 \boldsymbol{B} = \rho_0 \ddot{\boldsymbol{\varphi}} \quad \text{in} \quad \Omega \times I. \tag{2.1}$$

In (2.1), $\Omega \in \mathbb{R}^d$ for $d \in \{1,2,3\}$ is an open bounded domain, $I := \{t \in [0,T]\}$ is a closed time interval with $T > 0$, and $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}, t) : \Omega \times I \to \mathbb{R}^d$ is a mapping, with $\boldsymbol{X} \in \Omega$ and $t \in I$. The symbol $\boldsymbol{P}$ denotes the first Piola-Kirchhoff stress and $\rho_0 \boldsymbol{B} : \Omega \to \mathbb{R}^d$ is the body force, with ρ_0 denoting the mass density in the reference configuration. The over-dot notation denotes differentiation in time, so that $\dot{\boldsymbol{\varphi}} := \frac{\partial \boldsymbol{\varphi}}{\partial t}$ and $\ddot{\boldsymbol{\varphi}} := \frac{\partial^2 \boldsymbol{\varphi}}{\partial t^2}$. Embedded within $\boldsymbol{P}$ is a constitutive model, which can range from a simple linear elastic model to a complex micro-structure model, e.g., that of crystal plasticity. Herein, we focus on nonlinear hyper-elastic constitutive models such as the Henky model [14]. The details of this model are provided in Section 5.

Suppose that we have the following initial and boundary conditions for the PDEs (2.1):

$$\begin{gathered} \boldsymbol{\varphi}(\boldsymbol{X}, t_0) = \boldsymbol{X}_0, \;\; \dot{\boldsymbol{\varphi}}(\boldsymbol{X}, t_0) = \boldsymbol{v}_0 \text{ in } \Omega, \\ \boldsymbol{\varphi}(\boldsymbol{X}, t) = \boldsymbol{\chi} \text{ on } \partial\Omega_{\boldsymbol{\varphi}} \times I, \;\; \boldsymbol{P}\boldsymbol{N} = \boldsymbol{T} \text{ on } \partial\Omega_{\boldsymbol{T}} \times I. \end{gathered} \tag{2.2}$$

In (2.2), it is assumed the outer boundary $\partial\Omega$ is decomposed into a Dirichlet and traction portion, $\partial\Omega_{\boldsymbol{\varphi}}$ and $\partial\Omega_{\boldsymbol{T}}$, respectively, with $\partial\Omega = \partial\Omega_{\boldsymbol{\varphi}} \cup \partial\Omega_{\boldsymbol{T}}$ and $\partial\Omega_{\boldsymbol{\varphi}} \cap \partial\Omega_{\boldsymbol{T}} = \emptyset$. The prescribed boundary positions or Dirichlet boundary conditions are $\boldsymbol{\chi} : \partial\Omega_{\boldsymbol{\varphi}} \times I \to \mathbb{R}^3$. The symbol $\boldsymbol{N}$ denotes the unit normal on $\partial\Omega_{\boldsymbol{T}}$. In this work, we will assume without loss of generality that $\boldsymbol{\chi}$ is not changing in time.

It is straightforward to show that the weak variational form of (2.1) with initial and boundary conditions (2.2) is

$$\int_I \left[\int_\Omega (\operatorname{Div} \boldsymbol{P} + \rho_0 \boldsymbol{B} - \rho_0 \ddot{\boldsymbol{\varphi}}) \cdot \boldsymbol{\xi} \; \mathrm{d}V + \int_{\partial_{\boldsymbol{T}}\Omega} \boldsymbol{T} \cdot \boldsymbol{\xi} \; \mathrm{d}S \right] \mathrm{d}t = 0, \tag{2.3}$$

where $\boldsymbol{\xi}$ is a test function in $\mathcal{V} := \left\{ \boldsymbol{\xi} \in W_2^1(\Omega \times I) : \boldsymbol{\xi} = \boldsymbol{0} \text{ on } \partial_{\boldsymbol{\varphi}}\Omega \times I \cup \Omega \times t_0 \cup \;\Omega \times t_1 \right\}$.

Discretizing the variational form (2.3) in space using the classical Galerkin finite element method (FEM) [19] yields the following semi-discrete matrix problem:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{f}^{\text{int}}(\boldsymbol{u}, \dot{\boldsymbol{u}}) = \boldsymbol{f}^{\text{ext}}. \tag{2.4}$$

In (2.4), $\boldsymbol{M}$ denotes the mass matrix, $\boldsymbol{u} := \boldsymbol{\varphi}(\boldsymbol{X}, t) - \boldsymbol{X}$ is the displacement, $\ddot{\boldsymbol{u}}$ is the acceleration (also denoted by $\boldsymbol{a}$), $\boldsymbol{f}^{\text{ext}}$ is a vector of applied external forces, and $\boldsymbol{f}^{\text{int}}(\boldsymbol{u}, \dot{\boldsymbol{u}})$ is the vector of internal forces due to mechanical and other effects inside the material, where $\dot{\boldsymbol{u}}$ (also denoted by $\boldsymbol{v}$) is the velocity. In the present work, the semi-discrete equation (2.4) is advanced forward in time using the Newmark-β time-integration scheme [37]. We will assume the FOM (2.4) has size $N \in \mathbb{N}$, that is, $\boldsymbol{u} \in \mathbb{R}^N$. For convenience, in subsequent discussion, we transform the Dirichlet boundary condition (2.2) for the position into a Dirichlet boundary for the displacement, so that the boundary condition imposed on $\partial\Omega_{\boldsymbol{\varphi}} \times I$ is $\boldsymbol{u} = \boldsymbol{u}_D$, with $\boldsymbol{u}_D$ being independent of time.

3. Model order reduction. Projection-based model order reduction is a promising, physics-based technique for reducing the computational cost associated with high-fidelity models such as (2.4). The basic workflow for building a projection-based ROM for a generic nonlinear semi-discrete problem of the form (2.4) consists of three steps: (1) calculation of a reduced basis, (2) projection of the governing equations onto the reduced basis, and (3) hyper-reduction of the nonlinear terms in the projected equations. Herein, we employ the POD [42, 16] for the reduced basis generation step (step 1), the Galerkin projection method for the projection step (step 2), and the ECSW method [9] for the hyper-reduction step (step 3). Each of these steps is described succinctly below.

3.1. Proper orthogonal decomposition (POD). The POD is a mathematical procedure that, given an ensemble of data and an inner product, constructs a basis for the ensemble that is optimal in the sense that it describes more energy (on average) of the ensemble in the chosen inner product than any other linear basis of the same dimension M. The ensemble $\boldsymbol{w}^s \in \mathbb{R}^N : s = 1, ..., S$ is typically a set of S instantaneous snapshots of a numerical solution field, collected for S values of a parameter of interest, and/or at S different times. For solid mechanics problems, a natural choice for the snapshots is $\boldsymbol{w}^s = \boldsymbol{u}^s$, where the ensemble $\{\boldsymbol{u}^s\}$ denotes a set of snapshots for the displacement field. It is noted that one can use in place of on addition to $\{\boldsymbol{u}^s\}$ snapshots of the velocity ($\{\boldsymbol{v}^s\}$) and/or acceleration ($\{\boldsymbol{a}^s\}$) fields.

Following the so-called "method of snapshots" [42, 16], a POD basis $\boldsymbol{\Phi}_M \in \mathbb{R}^{N\times M}$ of dimension M is obtained by performing a singular value decomposition (SVD) of a snapshot matrix $\boldsymbol{W} := \left[\boldsymbol{w}^1, ..., \boldsymbol{w}^S\right] \in \mathbb{R}^{N\times S}$ such that $\boldsymbol{W} = \boldsymbol{\Phi\Sigma V}^T$ and defining $\boldsymbol{\Phi}_M$ as the matrix containing the first M columns of $\boldsymbol{\Phi}$. Letting

$$\mathcal{E}_{\text{POD}}(M) := \frac{\sum_{i=1}^{M} \sigma_i^2}{\sum_{i=1}^{S} \sigma_i^2} \tag{3.1}$$

denote the energy associated with a POD basis of size M, where σ_i denotes the i^{th} singular value of $\boldsymbol{W}$, the basis size M is typically selected based on an energy criterion, where $100\mathcal{E}_{\text{POD}}(M)$ is the percent energy captured by a given POD basis.

As discussed in Section 3.2, it is often desirable to construct the POD basis $\boldsymbol{\Phi}_M$ such that this basis satisfies homogeneous Dirichlet boundary conditions at some pre-defined indices $\boldsymbol{i}_d \in \mathbb{N}^d$ where $d < N$. It is straightforward to accomplish this by simply zeroing out the snapshots at the Dirichlet degrees of freedom (dofs) prior to calculating the SVD, that is, setting $\boldsymbol{w}^i(\boldsymbol{i}_d) = \boldsymbol{0}$ for $i = 1, ..., S$.

3.2. Galerkin projection. As discussed in [9, 45], Galerkin projection is in general the method of choice for solid mechanics and structural dynamics problems, as it preserves the Hamiltonian structure of the underlying system of PDEs [26]. The method starts by approximating the FOM displacement solution to (2.4) as

$$\boldsymbol{u} \approx \tilde{\boldsymbol{u}} = \bar{\boldsymbol{u}} + \boldsymbol{\Phi}_M \hat{\boldsymbol{u}}, \tag{3.2}$$

where $\hat{\boldsymbol{u}} \in \mathbb{R}^M$ is the vector of unknown modal amplitudes, to be solved for in the ROM and $\bar{\boldsymbol{u}} \in \mathbb{R}^N$ is a (possibly time-dependent) reference state. Similar approximations can be made for the velocity and acceleration fields, $\boldsymbol{v} := \dot{\boldsymbol{u}}$ and $\boldsymbol{a} := \ddot{\boldsymbol{u}}$, respectively, namely:

$$\begin{aligned} \boldsymbol{v} &\approx \tilde{\boldsymbol{v}} = \bar{\boldsymbol{v}} + \boldsymbol{\Phi}_M \hat{\boldsymbol{v}}, \\ \boldsymbol{a} &\approx \tilde{\boldsymbol{a}} = \bar{\boldsymbol{a}} + \boldsymbol{\Phi}_M \hat{\boldsymbol{a}}. \end{aligned} \tag{3.3}$$

Here, $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{a}}$ are defined analogously to $\bar{\boldsymbol{u}}$, and similarly for $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{a}}$.

In the present formulation, the reference states $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{a}}$ are used to prescribe strongly Dirichlet boundary conditions within the ROM for the displacement, velocity and acceleration fields, respectively. This is done following the approach of Gunzburger *et al.* [13]. Suppose the Dirichlet boundary conditions of interest are $\boldsymbol{u}(\boldsymbol{i}_d) = \boldsymbol{u}_D$, $\boldsymbol{v}(\boldsymbol{i}_d) = \boldsymbol{v}_D$ and $\boldsymbol{a}(\boldsymbol{i}_d) = \boldsymbol{a}_D$. Let $\boldsymbol{i}_u$ denote the unconstrained indices at which the solution to (2.4) is sought. It is straightforward to see from (3.2)–(3.3) that, if the POD modes $\boldsymbol{\Phi}_M = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_M]$ are calculated such that $\boldsymbol{\phi}_i(\boldsymbol{i}_d) = \boldsymbol{0}$ for $i = 1, ..., M$ and

$$\begin{array}{ll} \bar{\boldsymbol{u}}(\boldsymbol{i}_d) = \boldsymbol{u}_D, & \bar{\boldsymbol{u}}(\boldsymbol{i}_u) = \boldsymbol{0}, \\ \bar{\boldsymbol{v}}(\boldsymbol{i}_d) = \boldsymbol{v}_D, & \bar{\boldsymbol{v}}(\boldsymbol{i}_u) = \boldsymbol{0}, \\ \bar{\boldsymbol{a}}(\boldsymbol{i}_d) = \boldsymbol{a}_D, & \bar{\boldsymbol{a}}(\boldsymbol{i}_u) = \boldsymbol{0}, \end{array} \tag{3.4}$$

the ROM solution will satisfy the prescribed Dirichlet boundary conditions in the strong sense.

The ROM for (2.4) is obtained by substituting the decompositions (3.2)–(3.3) into the FOM equations (2.4), and projecting these equations onto the reduced basis $\boldsymbol{\Phi}_M$. It is straightforward to verify that doing this and moving all Dirichlet dofs to the right-hand side of the resulting system of equations yields a semi-discrete problem of the form

$$\hat{\boldsymbol{M}}_{uu}\hat{\boldsymbol{a}} + \hat{\boldsymbol{f}}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) = \hat{\boldsymbol{f}}_u^{\text{ext}}, \tag{3.5}$$

where

$$\begin{array}{c} \hat{\boldsymbol{M}}_{uu} := \boldsymbol{\Phi}_{M,u}^T \boldsymbol{M}_{uu} \boldsymbol{\Phi}_{M,u},\ \boldsymbol{f}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) := \boldsymbol{\Phi}_{M,u}^T \boldsymbol{f}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}), \\ \boldsymbol{f}_u^{\text{ext}} := \boldsymbol{\Phi}_{M,u}^T(\boldsymbol{f}_u^{\text{ext}} - \boldsymbol{M}_{ud}\bar{\boldsymbol{a}}_d). \end{array} \tag{3.6}$$

In (3.6), $\boldsymbol{\Phi}_{M,u} := \boldsymbol{\Phi}_M(\boldsymbol{i}_u, :)$, $\boldsymbol{M}_{uu} := \boldsymbol{M}(\boldsymbol{i}_u, \boldsymbol{i}_u)$, $\boldsymbol{f}_u^{\text{int}} := \boldsymbol{f}^{\text{int}}(\boldsymbol{i}_u)$, $\boldsymbol{f}_u^{\text{ext}} := \boldsymbol{f}^{\text{ext}}(\boldsymbol{i}_u)$, $\boldsymbol{M}_{ud} := \boldsymbol{M}(\boldsymbol{i}_u, \boldsymbol{i}_d)$ and $\bar{\boldsymbol{a}}_d := \bar{\boldsymbol{a}}(\boldsymbol{i}_d)$.

Remark 1. It is noted that, while all three variables $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{a}}$ appear in the ROM system (3.6), we are not considering the displacement, velocity and acceleration variables as independent fields with separate generalized coordinates in our ROM construction approach. In particular, $\hat{\boldsymbol{v}} := \frac{d\hat{\boldsymbol{u}}}{dt}$ and $\hat{\boldsymbol{a}} := \frac{d^2\hat{\boldsymbol{u}}}{dt^2}$ in (3.3). Taking this approach ensures consistency between the displacement, velocity and acceleration solutions computed within the ROM.

3.3. Hyper-reduction via ECSW. The POD/Galerkin approach to model reduction described in Sections 3.1 and 3.2 is not efficient for nonlinear problems, as the projection of the nonlinear terms appearing in the ROM system (3.5) requires algebraic operations that scale with the dimension of the original full order model N. This problem can be circumvented through the use of a procedure known as hyper-reduction. Here, we rely on a specific "project-then-approximate" hyper-reduction approach known as ECSW [9]. We select this approach, as it has been shown in [9] to preserve the Lagrangian structure associated with Hamilton's principle for second-order dynamical systems of the form (2.4). The resulting hyper-reduced ROM, termed HROM, is thus able to preserve the numerical stability properties of the Newmark-β time-integration scheme applied to advance the ROM system (3.5) forward in time.

In the context of solid mechanics, ECSW can be described as cubature-based approach with the goal of accurately estimating the projected internal forcing term (the "project" part of "project-then-approximate") $\hat{\boldsymbol{f}}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}})$ in (3.5) as opposed to directly estimating the nonlinear forcing term and then projecting the result with the appropriate basis. This approximation can be rewritten as a summation of each of the element-wise contributions

$$\begin{aligned} \hat{\boldsymbol{f}}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) &= \textstyle\sum_{e \in \mathcal{E}} \boldsymbol{\Phi}_M^T \boldsymbol{L}_e^T \boldsymbol{f}_{u,e}^{\text{int}}(\boldsymbol{L}_e \tilde{\boldsymbol{u}}, \boldsymbol{L}_e \tilde{\boldsymbol{v}}), \\ \hat{\boldsymbol{K}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) &= \textstyle\sum_{e \in \mathcal{E}} \boldsymbol{\Phi}_M^T \boldsymbol{L}_e^T \boldsymbol{K}_e(\boldsymbol{L}_e \tilde{\boldsymbol{u}}, \boldsymbol{L}_e \tilde{\boldsymbol{v}}) \boldsymbol{L}_e \boldsymbol{\Phi}_M. \end{aligned} \tag{3.7}$$

In (3.7):

- $\boldsymbol{f}_{u,e}^{\text{int}}$ is the contribution to the internal forcing vector due to mesh element e;
- $\boldsymbol{K}$, $\hat{\boldsymbol{K}}$ and $\boldsymbol{K}_e$ are the tangent stiffness matrix, reduced tangent stiffness matrix, and contribution to the tangent stiffness matrix due to element e respectively, which are formed for use in implicit time-stepping methods;
- $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$ is the set of all mesh elements e_i given by the finite element discretization of the FOM; and
- $\boldsymbol{L}_e \in \{0,1\}^{n_d^e \times N}$ is a Boolean matrix selecting the n_d^e degrees of freedom associated with mesh element e.

The next step is to relax the equality in (3.7) such that we only approximate the reduced internal forcing vector (or tangent stiffness matrix) by sampling a subset of the mesh elements and weighting them appropriately. This can be written as

$$\begin{aligned} \hat{\boldsymbol{f}}_u^{\text{int}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) &\approx \textstyle\sum_{e \in \tilde{\mathcal{E}}} \xi_e \boldsymbol{\Phi}_M^T \boldsymbol{L}_e^T \boldsymbol{f}_{u,e}^{\text{int}}(\boldsymbol{L}_e \tilde{\boldsymbol{u}}, \boldsymbol{L}_e \tilde{\boldsymbol{v}}), \\ \hat{\boldsymbol{K}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) &\approx \textstyle\sum_{e \in \tilde{\mathcal{E}}} \xi_e \boldsymbol{\Phi}_M^T \boldsymbol{L}_e^T \boldsymbol{K}_e(\boldsymbol{L}_e \tilde{\boldsymbol{u}}, \boldsymbol{L}_e \tilde{\boldsymbol{v}}) \boldsymbol{L}_e \boldsymbol{\Phi}_M, \end{aligned} \tag{3.8}$$

where $\tilde{\mathcal{E}} \subset \mathcal{E}$ such that $|\tilde{\mathcal{E}}| = N_e \leq N$ and $\xi_e \in \mathbb{R}_{\geq 0}$ is the positive weight associated with sampled mesh element $e \in \tilde{\mathcal{E}}$. In order to determine both the sampled mesh elements and their weights, a system of matrices and vectors is formed using N_h snapshots of $\boldsymbol{u}^s$ and $\boldsymbol{v}^s$ – the same snapshots used to compute the POD modes $\boldsymbol{\Phi}_M$. Using these snapshots, we form the following quantities:

$$\begin{aligned} \tilde{\boldsymbol{u}}^s &= \boldsymbol{\Phi}_M \boldsymbol{\Phi}_M^T (\boldsymbol{u}^s - \bar{\boldsymbol{u}}) + \bar{\boldsymbol{u}}, \\ \tilde{\boldsymbol{v}}^s &= \boldsymbol{\Phi}_M \boldsymbol{\Phi}_M^T (\boldsymbol{v}^s - \bar{\boldsymbol{v}}) + \bar{\boldsymbol{v}}, \\ c_{se} &= \boldsymbol{\Phi}_M^T \boldsymbol{L}_e^T \boldsymbol{f}_{u,e}^{\text{int}}(\boldsymbol{L}_e \tilde{\boldsymbol{u}}^s, \boldsymbol{L}_e \tilde{\boldsymbol{v}}^s), \\ d_s &= \boldsymbol{\Phi}_M^T \boldsymbol{f}_{u,e}^{\text{int}}(\tilde{\boldsymbol{u}}^s, \tilde{\boldsymbol{v}}^s), \end{aligned} \tag{3.9}$$

where $\tilde{\boldsymbol{u}}^s$ and $\tilde{\boldsymbol{v}}^s$ are the reconstruction of the FOM snapshots using the affine approximations in (3.2) and (3.3).

Given the quantities defined above, we can construct a system of equations

$$\boldsymbol{C}\boldsymbol{\xi} = \boldsymbol{d}, \quad \boldsymbol{\xi} \in \mathbb{R}_{\geq 0}^N, \tag{3.10}$$

where

$$\boldsymbol{C} := \begin{pmatrix} c_{11} & \ldots & c_{1N} \\ \vdots & \ddots & \vdots \\ c_{N_h 1} & \ldots & c_{N_h N} \end{pmatrix} \in \mathbb{R}^{MN_h \times N}, \quad \boldsymbol{d} := \begin{pmatrix} d_1 \\ \vdots \\ d_{N_h} \end{pmatrix} \in \mathbb{R}^{MN_h}$$

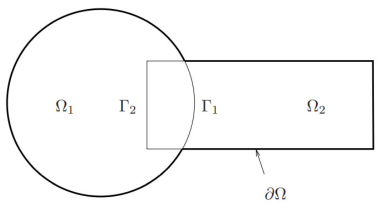
Equation (3.10) is satisfied by the choice $\boldsymbol{\xi} = \boldsymbol{1}$. This choice results in a reduced set of element equivalent to the FOM, providing no increase in computational efficiency because N_e, defined as the number of non-zero weights in $\boldsymbol{\xi}$ sampling the finite element mesh, is the same as the number of mesh elements. Instead, the equality in the equation (3.10) is relaxed to instead solve a constrained optimization problem

$$\boldsymbol{\xi} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^N} ||\boldsymbol{C}\boldsymbol{x} - \boldsymbol{d}||_2 \text{ subject to } \boldsymbol{x} \geq \boldsymbol{0}, \tag{3.11}$$

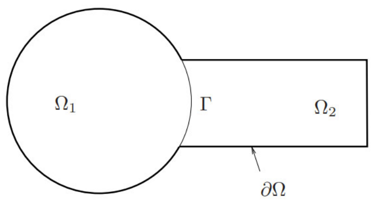
which, for the purposes of the present work, is solved using MATLAB's `lsqnonneg` function, with an early termination criterion with a solution step size tolerance of 10^{-4}. By relaxing the equality and including an early termination condition, the system can be solved approximately with a sparse solution for $\boldsymbol{\xi}$ such that, ideally, $N_e \ll N$. Note that our early termination criterion differs from the typical early termination criterion, in which the algorithm terminates once $||\boldsymbol{C\xi} - \boldsymbol{d}||_2/||\boldsymbol{d}||_2$ is less than some chosen tolerance (< 1). As a result, all of the HROM results presented later in the present work have a consistent termination criterion with respect to its MATLAB implementation; however, the relative error tolerance of the selected reduced elements will differ.

4. The Schwarz alternating method for multi-scale coupling.

In this section, we describe two variants of the Schwarz alternating method for concurrent multiscale coupling. The first is based on an overlapping DD (Figure 4.1(a)) and the second is based on a non-overlapping DD (Figure 4.1(b)). Consider without loss of generality a partition of the domain Ω into two open subdomains Ω_1 and Ω_2, as shown in Figure 4.1, and suppose we are interested in applying the method to a single-physics semi-discretized PDE of the form (2.4).



(a) Overlapping



(b) Non-overlapping

Fig. 4.1. *Illustration showing overlapping and non-overlapping domain decomposition.*

It is important to note that, to avoid a space-time discretization for the application of the Schwarz method, it proves convenient to subdivide the time domain into "controller time-intervals", $I_0 := [t_0, t_1]$, $I_1 := [t_1, t_2]$,..., as shown in Figure 4.2. The controller time-intervals are convenient markers for events of interest, and for synchronization of the Schwarz algorithm applied only in the space domain. They also define the periods or intervals in which the solutions of the initial boundary value problem (2.4) are determined by means of the Schwarz alternating method; effectively, the Schwarz iteration process is converged within a controller time interval I_N before advancing to the next controller time interval I_{N+1}. The interested reader is referred to [36] and [17] for details.

4.1. Overlapping Schwarz formulation.

Suppose a DD has been performed such that Ω_1 and Ω_2 are overlapping, that is, such that $\Omega_1 \cap \Omega_2 \neq \emptyset$, as shown in Figure 4.1(a). In this case, the Schwarz alternating iteration for the specific case of (2.4) with Dirichlet boundary conditions on $\partial\Omega$ takes the following form within each time-interval I_N (Figure 4.2):

$$
\begin{cases}
\boldsymbol{M}_1\ddot{\boldsymbol{u}}_1^{(n+1)} + \boldsymbol{f}_1^{\text{int};(n+1)} = \boldsymbol{f}_1^{\text{ext};(n+1)}, \text{ in } \Omega_1,\\
\boldsymbol{u}_1^{(n+1)} = \boldsymbol{u}_D, \text{ on } \partial_{\boldsymbol{\varphi}}\Omega_1\backslash\Gamma_1,\\
\boldsymbol{u}_1^{(n+1)} = \boldsymbol{u}_2^{(n)}, \text{ on } \Gamma_1,\\
\dot{\boldsymbol{u}}_1^{(n+1)} = \dot{\boldsymbol{u}}_2^{(n)}, \text{ on } \Gamma_1,\\
\ddot{\boldsymbol{u}}_1^{(n+1)} = \ddot{\boldsymbol{u}}_2^{(n)}, \text{ on } \Gamma_1,
\end{cases}
\quad
\begin{cases}
\boldsymbol{M}_2\ddot{\boldsymbol{u}}_2^{(n+1)} + \boldsymbol{f}_2^{\text{int};(n+1)} = \boldsymbol{f}_2^{\text{ext};(n+1)}, \text{ in } \Omega_2,\\
\boldsymbol{u}_2^{(n+1)} = \boldsymbol{u}_D, \text{ on } \partial_{\boldsymbol{\varphi}}\Omega_2\backslash\Gamma_2,\\
\boldsymbol{u}_2^{(n+1)} = \boldsymbol{u}_1^{(n+1)}, \text{ on } \Gamma_2,\\
\dot{\boldsymbol{u}}_2^{(n+1)} = \dot{\boldsymbol{u}}_1^{(n+1)}, \text{ on } \Gamma_2,\\
\ddot{\boldsymbol{u}}_2^{(n+1)} = \ddot{\boldsymbol{u}}_1^{(n+1)}, \text{ on } \Gamma_2.
\end{cases}
\tag{4.1}
$$

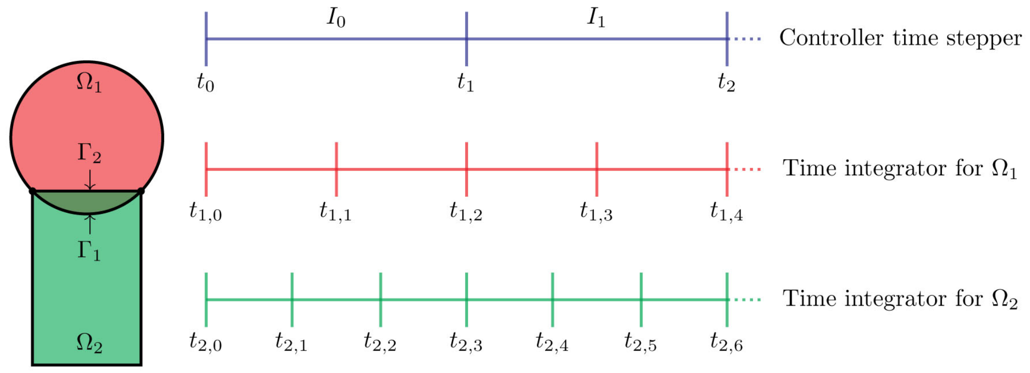


FIG. 4.2. *Illustration of controller time-stepper-based time advancement of a coupled problem involving two overlapping subdomains Ω_1 and Ω_2.*

In (4.1), (n) for $n = 0, 1, 2, ...$ denotes the Schwarz iterations number, and we have introduced the short-hand $\boldsymbol{f}_i^{\text{int}} := \boldsymbol{f}_i^{\text{int}}(\boldsymbol{u}, \dot{\boldsymbol{u}})$. The reader can observe that the transmission boundary conditions on the Schwarz boundaries Γ_1 and Γ_2 are of the Dirichlet type. The Schwarz Dirichlet boundary conditions are applied for not just the displacement field $\boldsymbol{u}$, but also for the velocity and acceleration fields, $\boldsymbol{v}$ and $\boldsymbol{a}$, as we found in our earlier work [35, 36] that these additional conditions are essential for maintaining accuracy within the acceleration and velocity fields when performing Schwarz-based coupling. The iteration (4.1) continues until convergence is reached. Herein, convergence of the method is declared when $||\boldsymbol{u}^{(n+1)} - \boldsymbol{u}^{(n)}||_2/||\boldsymbol{u}^{(n)}|| < \delta$, $||\boldsymbol{v}^{(n+1)} - \boldsymbol{v}^{(n)}||_2/||\boldsymbol{v}^{(n)}||_2 < \delta$ and $||\boldsymbol{a}^{(n+1)} - \boldsymbol{a}^{(n)}||_2/||\boldsymbol{a}^{(n)}||_2 < \delta$ (where $\boldsymbol{v}^{(n)} := \dot{\boldsymbol{u}}^{(n)}$ and $\boldsymbol{a}^{(n)} := \ddot{\boldsymbol{u}}^{(n)}$), for some specified Schwarz tolerance δ. It can be shown [36, 30] that the sequence defined by (4.1) will converge to the solution of the underlying single-domain problem provided that problem is well-posed, and the overlap is non-empty ($\Omega_1 \cap \Omega_2 \neq \emptyset$).

4.2. Non-overlapping Schwarz formulation. In certain applications, one is interested in performing coupling in a non-overlapping fashion (Figure 4.1(b)), e.g., for multiphysics applications such as fluid-structure interaction (FSI). As noted in Section 4.1, the overlapping formulation (4.1) will not converge in the case the overlap region is empty. This situation can be remedied by changing the transmission conditions applied on the Schwarz boundary Γ shown in Figure 4.1(b). Specifically, it can be shown [47, 10] that applying the following alternating Dirichlet-Neumann (or, more-specifically, displacement-traction) iteration in each controller time-interval I_N (Figure 4.2) gives rise to a convergent sequence of iterations:

$$\begin{cases} \boldsymbol{M}_1\ddot{\boldsymbol{u}}_1^{(n+1)} + \boldsymbol{f}_1^{\text{int};(n+1)} = \boldsymbol{f}_1^{\text{ext};(n+1)}, \text{ in } \Omega_1, \\ \boldsymbol{u}_1^{(n+1)} = \boldsymbol{u}_D, \text{ on } \partial_{\boldsymbol{\varphi}}\Omega_1\backslash\Gamma, \\ \boldsymbol{u}_1^{(n+1)} = \boldsymbol{\lambda}_{n+1}(\boldsymbol{u}), \text{ on } \Gamma, \\ \dot{\boldsymbol{u}}_1^{(n+1)} = \boldsymbol{\lambda}_{n+1}(\boldsymbol{v}), \text{ on } \Gamma, \\ \ddot{\boldsymbol{u}}_1^{(n+1)} = \boldsymbol{\lambda}_{n+1}(\boldsymbol{a}), \text{ on } \Gamma, \end{cases} \quad \begin{cases} \boldsymbol{M}_2\ddot{\boldsymbol{u}}_2^{(n+1)} + \boldsymbol{f}_2^{\text{int};(n+1)} = \boldsymbol{f}_2^{\text{ext};(n+1)}, \text{ in } \Omega_2, \\ \boldsymbol{u}_2^{(n+1)} = \boldsymbol{u}_D, \text{ on } \partial_{\boldsymbol{\varphi}}\Omega_2\backslash\Gamma, \\ \boldsymbol{T}_2^{(n+1)} = \boldsymbol{T}_1^{(n+1)}, \text{ on } \Gamma, \end{cases} \tag{4.2}$$

where

$$\boldsymbol{\lambda}_{n+1}(\boldsymbol{u}) = \theta\boldsymbol{u}_2^{(n)} + (1-\theta)\boldsymbol{\lambda}_n(\boldsymbol{u}) \text{ on } \Gamma, \text{ for } n \geq 1, \tag{4.3}$$

and similarly for $\boldsymbol{\lambda}_{n+1}(\boldsymbol{v})$ and $\boldsymbol{\lambda}_{n+1}(\boldsymbol{a})$, with $\theta \in (0,1]$ denoting a so-called relaxation parameter. In our present implementation, $\boldsymbol{\lambda}_n(\cdot)$ in (4.3) is initialized to zero, i.e., $\boldsymbol{\lambda}_0(\cdot) = \mathbf{0}$. As for the overlapping case, the iteration (4.2) continues until convergence is reached. If $\theta = 1$ in (4.3), there is no relaxation and the Schwarz boundary condition for Ω_1 in (4.2) reduces to a regular Dirichlet boundary condition. It has been demonstrated in the literature [47, 10, 7, 25] that selecting $\theta < 1$ (i.e., including relaxation) can reduce the number of Schwarz iterations required for convergence in certain applications.

It is noted that the formulation (4.2) is not unique; in particular, it is possible to create a convergent Schwarz iteration for the non-overlapping DD case if one uses Robin-Robin transmission conditions on the Schwarz boundary Γ, as shown in [31]. Here, we chose to develop the alternating Dirichlet-Neumann formulation over the Robin-Robin formulation because Dirichlet (displacement) and Neumann (traction) boundary conditions are readily available in most solid mechanics codes, unlike Robin boundary conditions.

Remark 2. Although this is not explicitly stated herein in order to avoid introducing overly complex notation, we note that the Schwarz boundary conditions appearing in (4.1) and (4.2) require the definition of projection operators $P_{\Omega_j \to \Gamma_i}[\cdot]$ and $P_{\Omega_j \to \Gamma}[\cdot]$, which take the solution in Ω_j, and project and interpolate it onto Γ_i or Γ. For more details on how this projection operator can be constructed, the interested reader is referred to [35, 36]. For 1D problems, such as those considered herein, the required projection and interpolation is trivial. It is noted that, in multiple spatial dimensions, the non-overlapping Schwarz formulation (4.2) requires the development of operators for consistent transfer (interpolation) of traction boundary conditions using the concept of prolongation/restriction. We plan to investigate the usage of the Compadre toolkit [38] for this task.

Remark 3. The Schwarz iterations (4.1) and (4.2) as written are performing something that is often referred to as the multiplicative Schwarz algorithm [11], meaning that the Schwarz iteration in subdomain Ω_2 at time-step $n+1$ depends on the Schwarz solution in subdomain Ω_1 at time-step $n+1$. The Schwarz iteration can be modified to achieve what is commonly referred to as the additive Schwarz method [11] by applying boundary conditions from the n^{th} Schwarz iteration in Ω_1 to the $(n+1)^{st}$ Ω_2 sub-problem. If this change is made, the Schwarz iteration sequences (4.1) and (4.2) can be parallelized over the number of subdomains. While we do not consider the additive variant of the Schwarz method in the present work, preliminary results have suggested that the method does not reduce solution accuracy and can achieve speed-ups if parallelized appropriately.

4.3. Extension to ROM-ROM and FOM-ROM coupling. As mentioned earlier, this paper presents two novel extensions of Schwarz-based concurrent coupling: (1) the extension of the original overlapping multi-scale coupling formulation [36] to the non-overlapping DD case (Section 4.2), and (2) the extension of the Schwarz coupling framework to the case when projection-based ROMs are being coupled to each other as well as to FOMs.

For the latter extension, it is particularly important to be able to prescribe time-varying Dirichlet boundary conditions strongly within a given subdomain ROM. We achieve this by applying the approach detailed in Section 3.2. In particular, the reference states $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{a}}$ in (3.2)–(3.3) are updated in each Schwarz iteration requiring application of a Dirichlet boundary condition on a Schwarz boundary. Given this approach, it is straightforward to extend the formulations (4.1) and (4.2) to projection-based ROMs, by simply replacing the FOM semi-discrete equations in these iterations with their ROM analogs (3.5).

Equally important to our Schwarz-based coupling strategy is that the boundary nodes on which the Schwarz transmission boundary conditions are imposed be included in the sample mesh when one is using hyper-reduction (Section 3.3). In our ROM boundary condition formulation (see Section 3.2), this is done automatically, as the boundary nodes are effectively removed from the ROM solve and incorporated them into the reference states $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{a}}$ appearing in (3.5).

Another important detail in performing Schwarz-based coupling involving projection-based ROMs relates to the snapshot collection strategy. Ideally, one would collect snapshots by performing simulations on the subdomains being coupled independently, that is, without running a coupled problem on the two (or more) domains. Snapshot collection strategies of this sort (e.g., using approaches such as oversampling [43]) will be examined in future work. In the present work, snapshots are generated in each subdomain by running a FOM-FOM coupled problem using the same DD as the targeted ROM-ROM or FOM-ROM coupling, saving the converged solutions within each subdomain, and utilizing these solutions for the construction of subdomain-local POD bases. We note that one could alternatively perform a single-domain FOM simulation on the full domain Ω, and use snapshots from that simulation restricted and/or interpolated onto each subdomain to generate subdomain POD bases; this second approach is not considered herein.

5. Numerical results: nonlinear wave propagation problem. The alternating Schwarz-based coupling strategy described in Section 4 is evaluated on a 1D wave propagation problem for the solid dynamics PDEs given in Section 2. Consider a simple beam geometry of length 1, so that $\Omega = (0,1) \in \mathbb{R}$. We will assume that this geometry is clamped at both ends, that is, $u(0,t) = u(1,t) = 0$ for all $t \geq 0$. The problem is initiated by prescribing an initial condition $u(x,0) = f(x)$ for $x \in \Omega$, where $f \in C^0(\Omega)$. In the numerical results presented herein, we consider two initial conditions:

$$f(x) = \frac{a}{2} \exp\left[-\frac{(x-b)^2}{2s^2}\right], \tag{5.1}$$

and

$$f(x) = a\left[\tanh(-b(x-s)) + \tanh(b(x-1+s))\right], \tag{5.2}$$

for $a, b, s \in \mathbb{R}$. We will refer to (5.1) as the "Symmetric Gaussian" initial condition, and to (5.2) as the "Rounded Square" initial condition. For both initial conditions, the problem is run until a final time of $T = 1.0 \times 10^{-3}$.

As demonstrated in Section 3.3 of [36], if Ω is assumed to be made up of a linear elastic material, it is possible to derive an exact analytical solution to the governing PDEs in terms of the initial condition $f(x)$ using the method of superposition. Since our objective herein is to evaluate the proposed model reduction coupling methodology on a nonlinear problem, we will assume Ω is made up of a nonlinear hyper-elastic material whose behavior is described by the Henky constitutive model [14] with Young's modulus $E = 1$GPa and density $\rho = 1000$ kg/m^3. The defining feature of Henky-type material models is that they possess a quadratic energy density function and that they use a logarithmic strain. For the 1D case, this reduces to

$$\begin{aligned} \lambda(x,t) &:= \frac{\partial\left[u(x,t)+x\right]}{\partial x}, \\ \varepsilon(x,t) &:= \log \lambda(x,t), \\ W(x,t) &:= \frac{E}{2}\varepsilon(x,t)^2, \end{aligned} \tag{5.3}$$

where λ is the stretch, ε is the strain, and W is the energy density. Henky-type material models are popular because they are seen as the natural extension of linear elasticity to the finite deformation regime, given that

$$\begin{aligned} \varepsilon(x,t) &:= \lambda(x,t) - 1, \\ W(x,t) &:= \frac{E}{2}\varepsilon(x,t)^2, \end{aligned} \tag{5.4}$$

for a linear elastic model, where its strain may be interpreted as an approximation to the logarithmic strain of the Henky model. Since the Henky material model variant of the problem of interest does not have an exact analytical solution, we began our study by first verifying that convergence of the single-domain FOM solution to the problem with respect to both mesh and time-step refinement is achieved. The details of this study are omitted herein for the sake of brevity.

Several snapshots of the solution to the Symmetric Gaussian version of our model problem are plotted in Figure 5.3. The reader can observe that the initial condition splits in two and propagates to both the left and the right, eventually reflecting back from the clamped boundaries. The behavior of the solution to the Rounded Square version of our model problem is similar. For both problem variants, a sharp gradient forms within the acceleration field over time. Although simple and 1D, the nonlinear wave propagation problem is a challenging test case for coupling methods, as artifacts introduced by these schemes will be clearly evident in the numerical solution. We additionally remark that our model problem is particularly difficult for ROMs, which are known to struggle on traveling wave problems and problems with sharp gradients. Here, we explore the possibility of mitigating this difficulty by: (1) restricting the ROM to a part of the domain and coupling it to a FOM in the remainder of the domain, as well as (2) constructing several spatially local ROMs and coupling them to each other.

While the Schwarz alternating method enables the coupling of different non-conformal meshes with different resolutions and different time-integration schemes with different time-steps, as shown in [35, 36], we chose, again for the sake of brevity, to focus the present study on varying the number of POD modes in each ROM subdomain and restricted attention to a non-overlapping DD of Ω into two subdomains, Ω_1 and Ω_2. As mentioned in the Introduction section, the present work is the first to formulate and evaluate the non-overlapping variant of our Schwarz alternating method for coupling, as all of our past work focused on overlapping coupling [35, 36]. For results demonstrating ROM-ROM and FOM-ROM coupling by means of the overlapping Schwarz alternating method, the interested reader is referred to [44].

Let Δx_i, Δt_i, M_i and $N_{e,i}$ denote the mesh resolution, time-step, POD basis size and number of sample mesh points in subdomain Ω_i for $i = 1, 2$, and let ΔT denote the controller time-step. Recall that the Schwarz tolerance is denoted by δ (see Section 4.1). Our study employed the following DD, mesh-resolutions, time-steps and Schwarz tolerance:

$$\begin{gathered} \Omega_1 = [0, 0.6], \;\; \Omega_2 = [0.6, 1], \\ \Delta x_1 = \Delta x_2 = 1.0 \times 10^{-3}, \\ \Delta t_1 = \Delta t_2 = \Delta T = 1.0 \times 10^{-7}, \\ \delta = 1.0 \times 10^{-11}. \end{gathered} \tag{5.5}$$

Our choice of DD is motivated by the observation that a much steeper gradient forms in the acceleration field in the left part of the domain than in the right during the time-interval considered (see Figure 5.3). This suggests that it may be possible to get away with using a relatively small ROM in Ω_2 while still maintaining accuracy, whereas likely a larger ROM or a FOM will be required in Ω_1.

In both subdomains, an implicit Newmark-β time-integrator with parameters $\beta = 0.49$

and $\gamma = 0.9$ [37] was used to advance the solution forward in time[1]. It can be shown [36] that the time-step required to resolve the wave and satisfy the Courant-Friedrichs-Lewy (CFL) condition is $\Delta t_{\text{CFL}} = \frac{\Delta x}{c}$ where $c = \sqrt{\frac{E}{\rho}}$ is the speed of the wave. Numerical experiments suggest that the time-step required to resolve the wave is in general one order of magnitude smaller than Δt_{CFL}. For the mesh resolutions and material parameters being considered, $\Delta t_{\text{CFL}} = 1.0 \times 10^{-6}$, which justifies our choice of time-step in (5.5).

To assess performance of our coupling method, we have written a MATLAB code that implements the discretization methods, model reduction schemes and coupling formulations described above. In this implementation, the non-negative least squares problem defining the hyper-reduction weights within the ECSW algorithm (see Section 3.3) is solved using MATLAB's `lsqnonneg` function with an early termination condition, as described in Section 3.3. We present results for both a reproductive (Section 5.1) and a predictive (Section 5.2) variant of our model problem. These problem variants are described in more detail below.

To assess our coupled models' accuracy, we report the mean-square error (MSE) for each of the solution fields: displacement, velocity and acceleration. For the displacement field, we calculate this quantity using the following formula:

$$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{u}}_i) := \frac{\sqrt{\sum_{n=1}^{S} ||\tilde{\boldsymbol{u}}_i^n - \boldsymbol{u}_i^n||_2^2}}{\sqrt{\sum_{n=1}^{S} ||\boldsymbol{u}_i^n||_2^2}}, \tag{5.6}$$

where $\boldsymbol{u}_i^n$ is the FOM displacement solution (the reference solution) at time t_n in subdomain Ω_i, and $\tilde{\boldsymbol{u}}_i^n$ is the ROM displacement solution at time t_n in subdomain Ω_i. As implied by (5.6), MSEs for coupled ROM-ROM solutions are calculated by performing an analogous coupled FOM-FOM simulation, and computing errors in each subdomain ROM with respect to its corresponding subdomain FOM in the FOM-FOM coupling. The formula (5.6) can also be used to calculate errors in a FOM-ROM coupling with respect to a FOM-FOM simulation. Assuming without loss of generality that the FOM is prescribed in Ω_1, (5.6) is modified by replacing $\tilde{\boldsymbol{u}}_1$ with the FOM solution in Ω_1 in the FOM-ROM coupling. The MSE for the velocity and acceleration fields in subdomain Ω_i are denoted by $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{v}}_i)$ and $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{a}}_i)$, respectively, and defined analogously to (5.6).

Also reported in the following subsections is the total number of Schwarz iterations required for convergence, denoted by N_S, and the CPU time for each run. All test cases were run in MATLAB in serial on one of two Linux RHEL workstations located at Sandia National Laboratories. The reproductive cases described in Section 5.1 were run on a RHEL8 Intel(R) Xeon(R) CPU E5-2650 v3 2.30GHz machine, whereas the predictive cases described in Section 5.2 were run on a RHEL7 Intel(R) Xeon(R) CPU E7-4880 v2 2.50GHz machine. Since a constant time-step of 1.0×10^{-7} was used for all of our runs, the average number of Schwarz iterations per time-step can easily be calculated as $N_S/10,000$.

5.1. Reproductive ROM results. We first study the performance of the non-overlapping variant of the proposed alternating Schwarz coupling formulation (see Section 4.2) in the reproductive regime. We prescribe as the initial condition for our problem the Symmetric Gaussian (5.1), with $a = 1.0 \times 10^{-3}$, $b = 0.5$ and $s = 0.02$, and employ the DD, mesh resolutions, time-steps and Schwarz tolerance given in (5.5). To generate snapshots for building our ROMs, we use the non-overlapping Schwarz alternating method to perform a FOM-FOM coupled simulation in $\Omega := \Omega_1 \cup \Omega_2$. A total of S =10,001 snapshots of

[1]We chose $\beta = 0.49$ and $\gamma = 0.9$ within the Newmark-β scheme, as these parameter values introduce some numerical dissipation into the discretization, which our FOM-based convergence study revealed is necessary to stabilize the rapidly-varying acceleration solution.

the displacement are collected between time 0 and the final time $T = 1.0 \times 10^{-3}$ within each subdomain at time-increments of 1.0×10^{-7}. These snapshots are used to build POD bases for our subdomain ROMs and HROMs. For comparison and to provide a baseline, we also perform a simulation involving a single-domain FOM, from which we build a single-domain ROM and HROM. Note that, unless specified otherwise, the snapshots used for POD refer to displacement snapshots.

Figure 5.1(a) shows the snapshot energy $\mathcal{E}_{\text{POD}}$ (3.1) as a function of the basis size for the single-domain and two subdomain ROM cases. For the single-subdomain case, 64 POD modes capture 99.99% of the snapshot energy. For the two subdomain case, 54 and 21 POD modes are required to capture 99.99% of the snapshot energy in Ω_1 and Ω_2, respectively. Interestingly, if one wishes to create a basis that captures 100% of the snapshot energy, significantly more modes are required (509 modes for the single-domain case, and 414/145 modes in Ω_1/Ω_2 for the two subdomain case). As expected, fewer modes are needed to reproduce the solution in Ω_2 than in Ω_1.

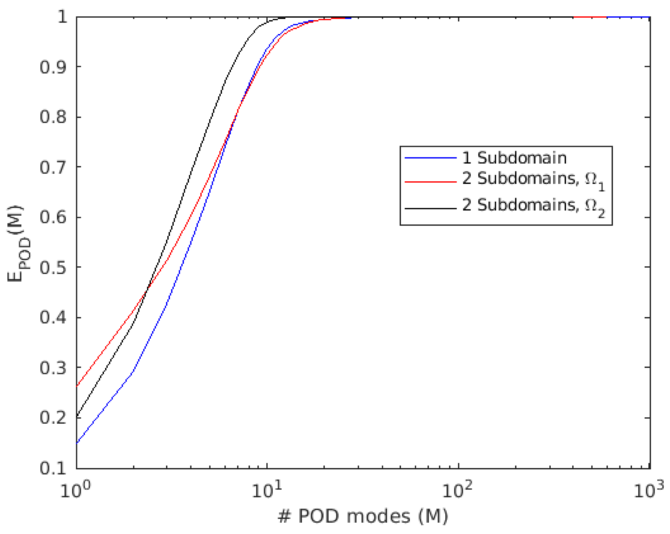


Fig. 5.1. *Energy decay for POD modes constructed from displacement snapshots for the reproductive Symmetric Gaussian variant of the nonlinear wave propagation problem.*

Table 5.1 reports results for several single-domain ROMs and their hyper-reduced variants, termed HROMs, as well as results for several FOM-ROM, ROM-ROM, FOM-HROM and HROM-HROM couplings. While the snapshot energies plotted in Figure 5.1 suggest that a ROM consisting of less than 70 POD modes should be adequate in reproducing the snapshot set, the reader can observe by examining Table 5.1 that a single-domain ROM comprised of 60 POD modes is fairly inaccurate, achieving an MSE of almost 50% for the acceleration field. Since such results are unacceptable in most engineering applications, we decided to consider larger ROMs having $\mathcal{O}(100)$ modes in our coupling studies.

Prior to generating the results summarized in Table 5.1, we performed a parameter sweep study to determine the "optimal" value of the relaxation parameter θ in (4.2), that is the value of θ that yielded the smallest number of Schwarz iterations, N_S, for non-overlapping FOM-FOM and FOM-ROM couplings. This study revealed that the value of θ that minimizes N_S is 1, which corresponds to (4.2) with no relaxation. All results reported herein hence used a value of $\theta = 1$. The HROMs evaluated were generated using the ECSW hyper-reduction method described in Section 3.3. A total of $N_h = 20$ snapshots were used to generate the state basis used to train the reduced mesh. These snapshots were created by selecting every 500^{th} displacement snapshot used to generate the POD bases.

A number of noteworthy observations can be made by examining Table 5.1. We begin by discussing the relative performance of the various models. All couplings summarized in Table 5.1 were run on the same Linux RHEL8 workstation, to ensure consistency, as discussed earlier. The reader can observe by examining this table that all coupled models evaluated converged on average in less than 3 Schwarz iterations per time-step. This indicates that the coupling method has not introduced a significant amount of overhead. Whereas the FOM-ROM couplings converge in approximately the same number of total Schwarz iterations, N_S, as the FOM-FOM coupling, the FOM-HROM, ROM-ROM and HROM-HROM models require more Schwarz iterations to reach convergence. It is interesting to observe that the larger (300/80 mode) ROM-ROM is actually slightly faster than the smaller (200/80 mode) ROM-ROM. While, at first glance, this result may seem counter-intuitive, the reason for this behavior is clear when comparing N_S for the two couplings: the larger ROM-ROM requires fewer Schwarz iterations to converge. Most likely, this is due to the larger ROM in Ω_1 being more accurate. This result is consistent with a previously-observed trend, which showed that coupling less accurate models with each other in general requires more Schwarz iterations to reach convergence [35, 36]. Despite requiring more Schwarz iterations to converge, the FOM-HROM and HROM-HROM couplings outperform the FOM-FOM coupling in terms of CPU time by between 12.5-32.6%. This is a win in the case where a FOM-FOM coupling is the gold standard, as would occur if the problems in Ω_1 and Ω_2 were discretized by two disparate codes, making it impossible to perform a monolithic simulation on Ω as a single domain. While all of the couplings except the smaller HROM-HROM coupling are slower than the single-domain FOM, our preliminary results suggest that speedups over a single-domain FOM are possible through an additive Schwarz implementation of our method (see Remark 3). This variant of the Schwarz alternating method will be explored in a subsequent publication.

Having discussed performance, we now turn our attention to accuracy. First, convergence is observed with basis refinement for all the models except the larger HROM-HROM. The reader can observe that the MSEs in Ω_2 are lower for all ROM-ROM and HROM-HROM couplings despite the fact that fewer modes are employed within this subdomain. This is due to the solution in Ω_2 being far smoother than the solution in Ω_1 for the time interval considered (see Figure 5.3). Among the most accurate models involving HROMs are the FOM-HROMs considered, which achieve errors of $\mathcal{O}(0.01\%)$ or less in the displacement solution and of $\mathcal{O}(1\%)$ or less in the acceleration solution. It is interesting and encouraging to remark that the larger FOM-ROM model is incredibly accurate, achieving an MSE of $\mathcal{O}(10^{-11})$ in the displacement solution. While our HROM-HROM couplings achieve reasonable errors for the displacement and velocity fields, non-trivial errors of $\mathcal{O}(10\%)$ appear to be unavoidable when using these models given the fixed early termination criterion used in our implementation of ECSW. In interpreting this result, it is important to recognize that all couplings involving ROMs and HROMs are at least as accurate as the single-domain ROMs and HROMs evaluated in Table 5.1. This result suggests that the errors we are seeing in the coupled models are due to the inaccuracy of the individual subdomain models, rather than errors introduced by our coupling framework. It may be possible to improve the accuracy of coupled ROMs/HROMs by decomposing the spatial domain into more subdomains and creating/coupling a larger number of spatially-localized ROMs/HROMs. It may also be possible to improve the HROM-HROM results in Table 5.1 by simply changing the termination tolerance within MATLAB's `lsqnonneg` algorithm used to solve (3.9). As discussed in Section 3.3, we did not do this here, as our goal wasto have a consistent termination

TABLE 5.1
Reproductive non-overlapping Schwarz coupling results, with $\theta = 1$. The POD modes for the couplings involving ROMs and HROMs were constructed from snapshots of only the displacement field. Couplings outperforming the FOM-FOM model in terms of CPU-time and with reasonable errors are high-lighted in green.

Model	M_1/M_2	$N_{e,1}/N_{e,2}$	CPU time (s)	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{u}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{u}}_2)$	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{v}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{v}}_2)$	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{a}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{a}}_2)$	N_S
FOM	$-/-$	$-/-$	1.871×10^3	$-/-$	$-/-$	$-/-$	–
ROM	$60/-$	$-/-$	1.398×10^3	$1.659 \times 10^{-2}/-$	1.037×10^{-1}	$4.681 \times 10^{-1}/-$	–
HROM	$60/-$	$155/-$	5.878×10^2	$1.730 \times 10^{-2}/-$	$1.063 \times 10^{-1}/-$	$4.741 \times 10^{-1}/-$	–
ROM	$200/-$	$-/-$	1.448×10^3	$2.287 \times 10^{-4}/-$	$4.038 \times 10^{-3}/-$	$4.542 \times 10^{-2}/-$	–
HROM	$200/-$	$428/-$	9.229×10^2	$8.396 \times 10^{-4}/-$	$8.947 \times 10^{-3}/-$	$7.462 \times 10^{-2}/-$	–
FOM-FOM	$-/-$	$-/-$	2.345×10^3	–	–	–	24,630
FOM-ROM	$-/80$	$-/-$	2.341×10^3	$2.171 \times 10^{-6}/$ 1.253×10^{-5}	$3.884 \times 10^{-5}/$ 2.401×10^{-4}	$2.982 \times 10^{-4}/$ 2.805×10^{-3}	25,227
FOM-HROM	$-/80$	$-/130$	2.085×10^3	$2.022 \times 10^{-4}/$ 5.734×10^{-4}	$1.723e \times 10^{-3}/$ 5.776×10^{-3}	$7.421 \times 10^{-3}/$ 3.791×10^{-2}	29,678
FOM-ROM	$-/200$	$-/-$	2.449×10^3	$4.754 \times 10^{-12}/$ 7.357×10^{-11}	$1.835 \times 10^{-10}/$ 4.027×10^{-9}	$5.550 \times 10^{-9}/$ 1.401×10^{-7}	24,630
FOM-HROM	$-/200$	$-/252$	2.352×10^3	$1.421 \times 10^{-5}/$ 4.563×10^{-4}	$1.724 \times 10^{-4}/$ 2.243×10^{-3}	$9.567 \times 10^{-4}/$ 1.364×10^{-2}	27,156
ROM-ROM	$200/80$	$-/-$	2.778×10^3	$4.861 \times 10^{-5}/$ 3.093×10^{-5}	$1.219 \times 10^{-3}/$ 4.177×10^{-4}	$1.586 \times 10^{-2}/$ 3.936×10^{-3}	27,810
HROM-HROM	$200/80$	$315/130$	1.769×10^3	$3.410 \times 10^{-3}/$ 6.662×10^{-4}	$4.110 \times 10^{-2}/$ 6.432×10^{-3}	$2.485 \times 10^{-1}/$ 4.307×10^{-2}	29,860
ROM-ROM	$300/80$	$-/-$	2.646×10^3	$2.580 \times 10^{-6}/$ 1.292×10^{-5}	$6.226 \times 10^{-5}/$ 2.483×10^{-4}	9.470×10^{-4} 2.906×10^{-3}	25,059
HROM-HROM	$300/80$	$405/130$	1.938×10^3	6.960×10^{-3} 7.230×10^{-4}	6.328×10^{-2} 7.403×10^{-3}	3.137×10^{-1} 4.960×10^{-2}	29,896

criterion for all HROMs being evaluated.

In an effort to provide a complete picture of the relative computational cost and accuracy of the models evaluated, we show in Figure 5.2 a Pareto plot, which plots the CPU time in seconds versus the average displacement MSE over all subdomains being coupled. The reader can observe that, while the single-domain ROMs and HROMs are the fastest, our coupling methodology enables us to achieve lower errors by performing ROM-ROM and FOM-ROM couplings. Future work will examine ways to improve the accuracy and efficiency of FOM-HROM and HROM-HROM couplings, which are not optimal according to Figure 5.2. It is not possible to make a definitive conclusion about the general utility of FOM-HROM and HROM-HROM couplings until we have evaluated our coupling methodology in two or three spatial dimensions. In multiple spatial dimensions, we anticipate seeing a greater benefit from hyper-reduction, as well as a greater potential for generating "optimal" (MSE and CPU-time minimizing) DDs and ROM/FOM assignments, in the spirit of [2].

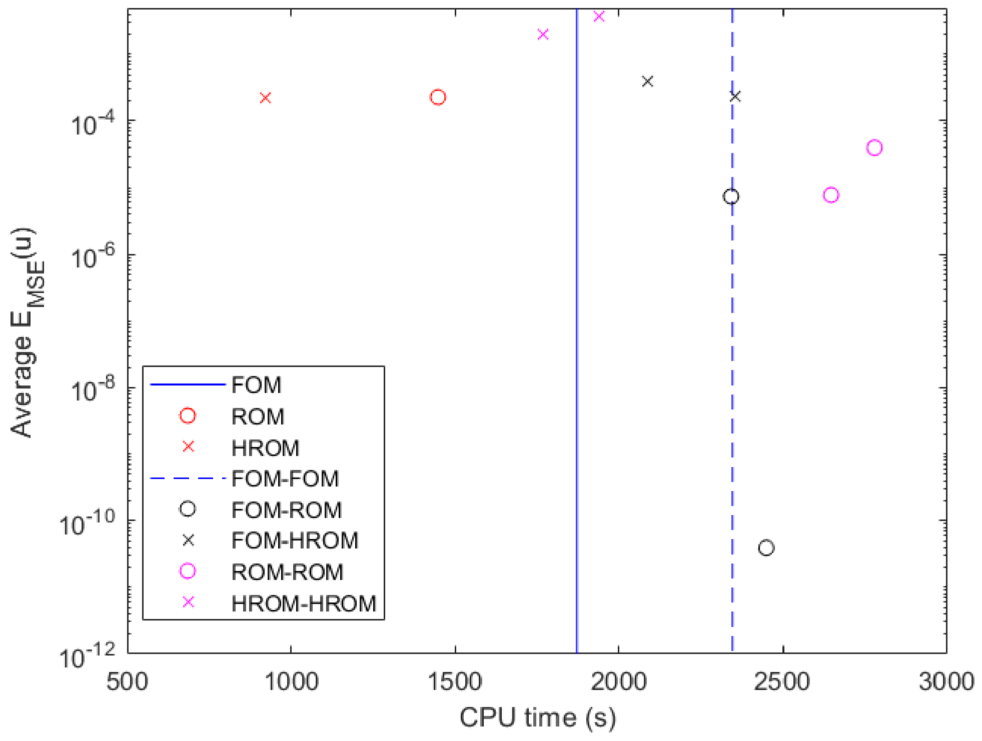
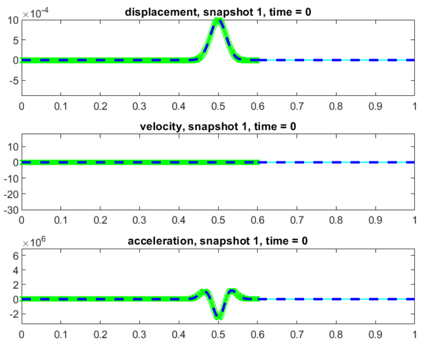
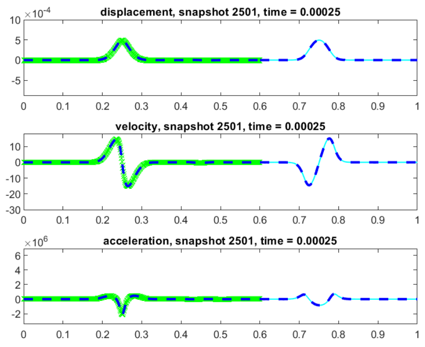


Fig. 5.2. *Pareto plot showing CPU time in seconds versus the average displacement MSE over all subdomains being coupled for the reproductive couplings evaluated in Table 5.1.*

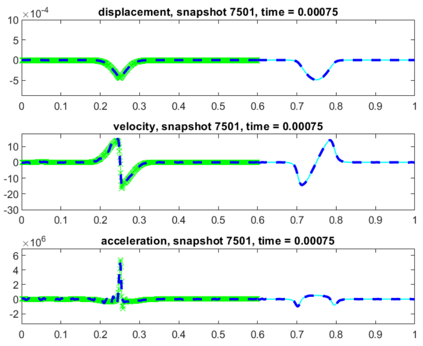
In Figure 5.3, we plot the solution computed using our non-overlapping Schwarz coupling method for the smaller HROM-HROM coupling summarized in Table 5.1. In this figure, the solution in Ω_1 is shown in green, and the solution in Ω_2 is shown in cyan. It is evident that the coupled solutions are indistinguishable from the single-domain FOM solution in the full domain Ω, shown in blue. This indicates that the coupling method has not introduced any spurious artifacts into the discretization.



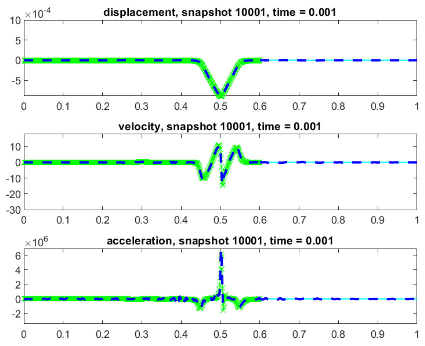
(a) $t = 0$



(b) $t = 2.5 \times 10^{-4}$



(c) $t = 7.5 \times 10^{-4}$



(d) $t = 1.0 \times 10^{-3}$

FIG. 5.3. *Plots of the HROM-HROM solutions with $M_1 = 200$, $M_2 = 80$, $N_{e,1} = 315$ and $N_{e,2} = 130$ (see Table 5.1) compared to a single-domain FOM solution. The coupled solutions in Ω_1 and Ω_2 are shown in green and cyan, respectively, whereas the single-domain FOM solution is shown in blue. No coupling artifacts are observed in the displacement, velocity and acceleration solutions.*

5.1.1. Alternate POD bases.

The reader can observe by inspecting Figure 5.3 that, whereas the displacement and velocity solutions are relatively smooth, sharp gradients form and propagate in the acceleration field over the course of the simulation. Since these gradients are difficult to capture using POD modes, significantly larger MSEs are observed in the acceleration field than in the displacement and velocity fields (see Table 5.1). This observation motivated us to explore a simple idea for improving on the results in Table 5.1 by augmenting the POD basis used in constructing each of our ROMs with modes calculated using snapshots of the acceleration field. Let $\mathbf{\Phi}_M^{\boldsymbol{u}}$ and $\mathbf{\Phi}_M^{\boldsymbol{a}}$ denote POD bases of size M computed from displacement and acceleration modes, respectively. Our procedure for generating a combined displacement-acceleration POD bases involved first computing $\mathbf{\Phi}_M^{\boldsymbol{u}}$ and $\mathbf{\Phi}_M^{\boldsymbol{a}}$ independently. Once this was done, an SVD was performed of the augmented matrix $[\mathbf{\Phi}_M^{\boldsymbol{u}}, \mathbf{\Phi}_M^{\boldsymbol{a}}]$ to remove potential linear dependencies between the two bases, and the resulting basis was truncated as desired to yield a displacement-acceleration POD basis.

Remark 4. The idea to include snapshots of the acceleration field within the POD basis of a ROM is related to the idea of including so-called "time difference quotients" (numerical estimates of the time-derivative of the primary solution field) in the generation of POD modes [21]. These difference quotients can be thought of as representing the right-hand side of a nonlinear PDE being solved, and were demonstrated in [21] to lead to ROMs with improved convergence rates.

Unfortunately, while the "displacement-acceleration" basis generation approach discussed above improved the errors by approximately four orders of magnitude for the FOM-ROM couplings we tried, the approach was actually deleterious when it came to FOM-HROM, ROM-ROM and HROM-HROM couplings. The reason for this is unknown at the present time, and is something we plan to look into in future work.

5.2. Predictive ROM results. Having evaluated the proposed coupling methodology on a reproductive test case, we now turn our attention to a predictive variant of the nonlinear wave propagation problem. First, snapshots are generated by simulating the problem with the Symmetric Gaussian (5.1) initial condition with parameters $a = 1.0 \times 10^{-3}$, $b = 0.5$ and $s = 0.02$ up to time $T = 1.0 \times 10^{-3}$, as before. A total of 10,001 snapshots of the displacement field are generated at increments of 1.0×10^{-7}. POD modes are constructed from these snapshots, and used to predict the solution of our model problem with a different initial condition, namely the Rounded Square (5.2), with parameters $a = 5.0 \times 10^{-4}$, $b = 100$ and $s = 0.6$. As with the Symmetric Gaussian variant of this problem, the predictive simulation is run until time $T = 1.0 \times 10^{-3}$.

Before building any ROMs and performing any couplings, we assess the projection error for the Rounded Square nonlinear wave propagation problem, defined as

$$\mathcal{E}_{\text{proj}}(\boldsymbol{u}, \boldsymbol{\Phi}_M) := \frac{||\boldsymbol{u} - \boldsymbol{\Phi}_M(\boldsymbol{\Phi}_M^T\boldsymbol{\Phi}_M)^{-1}\boldsymbol{\Phi}_M^T\boldsymbol{u}||_2}{||\boldsymbol{u}||_2}, \tag{5.7}$$

where $\boldsymbol{u}$ denotes the snapshots of the displacement field. The projection error can be computed in a similar way for the velocity and acceleration snapshots, and is denoted by $\mathcal{E}_{\text{proj}}(\boldsymbol{v}, \boldsymbol{\Phi}_M)$ and $\mathcal{E}_{\text{proj}}(\boldsymbol{a}, \boldsymbol{\Phi}_M)$, respectively. Equation (5.7) provides a straightforward and inexpensive way to evaluate a given basis's ability to represent a solution without having to construct and run an entire projection-based ROM. Figure 5.4 shows a plot of the projection error for the Rounded Square nonlinear wave propagation problem simulated on one domain Ω as a function of the basis size M. We calculate and report $\mathcal{E}_{\text{proj}}(\boldsymbol{u}, \boldsymbol{\Phi}_M)$, $\mathcal{E}_{\text{proj}}(\boldsymbol{v}, \boldsymbol{\Phi}_M)$ and $\mathcal{E}_{\text{proj}}(\boldsymbol{a}, \boldsymbol{\Phi}_M)$ for two kinds of POD bases: reproductive and predictive. The reader can observe that the projection error decreases with basis refinement for both basis types, indicating that obtaining a reasonably accurate solution should be possible even with the predictive POD basis, provided enough modes are retained. It is noted that the acceleration projection error is roughly two order of magnitude higher than the displacement projection error, and exhibits a much slower decay than the displacement and velocity projection errors.

As for the reproductive problem considered in Section 5.1, we evaluate the performance of the non-overlapping variant of the proposed Schwarz-based coupling method (Section 4.2) with the DDs, mesh resolutions, time-steps and Schwarz tolerance given in (5.5). Again, we set $\theta = 1$ in (4.3), as this value yielded the best performance. We consider predictive ROMs with a relatively large number of POD modes in each subdomain: 300 modes for a single-domain ROM, and 300/200 modes for a two subdomain ROM in Ω_1/Ω_2. This choice of basis size is motivated by the projection error results plotted in Figure 5.4, which demonstrate that a minimum of $O(100)$ modes are needed to reproduce our three solution fields to a sufficient accuracy in the predictive regime.

The main results are summarized in Table 5.2. All runs summarized in Table 5.2 were performed on the same Linux RHEL7 workstation, to ensure consistency of CPU times for an apples-to-apples comparison, as described earlier. First, we assess the single-domain

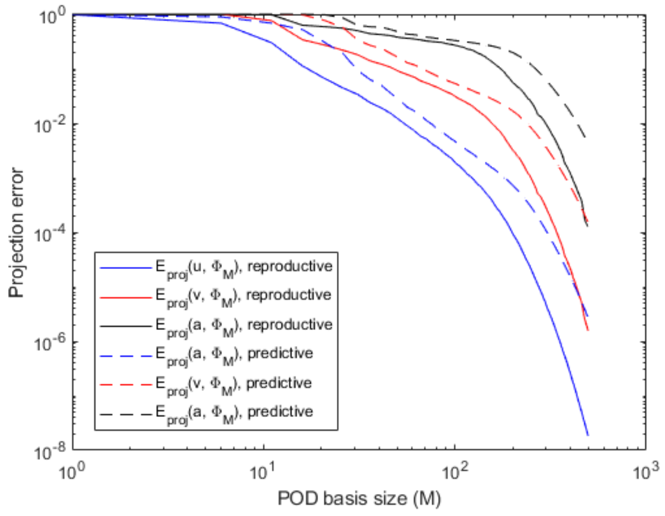


FIG. 5.4. *Projection errors for the nonlinear wave propagation problem with the Rounded Square initial condition.*

ROM and HROM results. The reader can observe that the MSEs for the ROM and HROM are almost identical. This is likely due to the fact that a very large number of mesh points (more than 60%) is being selected by the ECSW algorithm when constructing the HROM. It is interesting to remark that the ROM is actually slower than the FOM. This happens frequently when constructing projection-based ROMs for nonlinear problems without using hyper-reduction, and highlights the importance of hyper-reduction.

Turning our attention to the coupled results, we begin by discussing accuracy, as done in Section 5.1. It can be seen that the FOM-ROM coupling is remarkably accurate, achieving MSEs as low as $\mathcal{O}(10^{-8})$ for this *predictive* problem. This result supports our claim that a more accurate and robust model can be obtained via our Schwarz-based coupling, and suggests that no coupling errors have been introduced into the discretization. Although the FOM-HROM and ROM-ROM couplings are less accurate than the FOM-ROM coupling, one can see that they achieve MSEs lower than the single-domain ROM and single-domain HROM summarized in Table 5.2[2]. The MSEs for the HROM-HROM model in Table 5.2 are about one order of magnitude higher than its corresponding ROM-ROM, but nonetheless on par with the single-domain HROM we evaluated when it comes to MSE.

Next, we discuss the efficiency of the various models in Table 5.2. Interestingly, the FOM-ROM model takes slightly less CPU time to converge than its corresponding FOM-FOM model despite not having hyper-reduction, and requires approximately the same number of Schwarz iterations to converge. The same cannot be said for the FOM-HROM, ROM-ROM and HROM-HROM summarized in Table 5.2, which are slower than their analogous FOM-FOM coupled model. This is due to the number of Schwarz iterations, which is up to 30% higher than for the FOM-FOM and FOM-ROM couplings (with an average of as many as three Schwarz iterations per time-step needed to converge the HROM-HROM solution). As discussed in Section 5.1, the larger number of Schwarz iterations needed to reach convergence is most likely due to the lower accuracy of the individual models be-

[2]There is, of course, the caveat that the single-domain ROM has less total modes than our ROM-ROM coupling being evaluated.

ing coupled. In the case of the HROM-HROM case, the reader can observe that, like for the single-domain HROM, more than 60% of the mesh points are being sampled by the ECSW algorithm for the HROM-HROM coupling, which contributes to the higher CPU time attained by this model. The reader is reminded that, although the non-negative least squares solver `lsqnonneg` in MATLAB used in the ECSW procedure has a consistent early termination condition given by a fixed solution step size tolerance of 10^{-4}, the relative error tolerance $||\boldsymbol{C\xi} - \boldsymbol{d}||_2/||\boldsymbol{d}||_2$ from Section 3.3 will differ between reduced meshes depending on the choice of number of modes and number of degrees of freedom in a given subdomain. It may be possible to obtain a more accurate HROM solution by simply tweaking the `lsqnonneg` termination tolerance, but we chose not to do that here to ensure consistency between all methods being evaluated, as discussed in Section 3.3. Like in Section 5.1, we remark that all coupled models in Table 5.2 can be improved further through the use of the additive Schwarz variant, as discussed in Remark 3.

Table 5.2

Predictive non-overlapping Schwarz coupling results, with $\theta = 1$. The POD modes for the couplings involving ROMs and HROMs were constructed from snapshots of only the displacement field. The ROMs and HROMs summarized in this table had the following numbers of POD modes: $M_1 = 300$, $M_2 = 200$. Couplings outperforming the FOM-FOM model in terms of CPU-time and with reasonable errors are highlighted in green.

Model	CPU time (s)	$N_{e,1}/N_{e,2}$	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{u}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{u}}_2)$	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{v}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{v}}_2)$	$\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{a}}_1)/$ $\mathcal{E}_{\text{MSE}}(\tilde{\boldsymbol{a}}_2)$	N_S
FOM	1.288×10^3	–/–	–/–	–/–	–/–	–
ROM	1.358×10^3	–/–	3.451×10^{-3}/–	6.750×10^{-2}/–	3.021×10^{-1}/–	–
HROM	9.759×10^2	614/–	3.463×10^{-3}/–	6.750×10^{-2}/–	3.021×10^{-1}/–	–
FOM-FOM	2.133×10^3	–/–	–/–	–/–	–/–	23,280
FOM-ROM	2.084×10^3	–/–	1.907×10^{-8}/ 1.170×10^{-6}	1.461×10^{-6}/ 9.882×10^{-5}	3.973×10^{-5}/ 1.757×10^{-3}	23,288
FOM-HROM	2.219×10^3	–/253	1.967×10^{-4} 1.720×10^{-3}	4.986×10^{-3} 4.185×10^{-2}	2.768×10^{-2} 2.388×10^{-1}	29,700
ROM-ROM	2.502×10^3	–/–	5.592×10^{-4}/ 4.346×10^{-4}	1.575×10^{-2}/ 1.001×10^{-2}	9.197×10^{-2}/ 5.304×10^{-2}	26,220
HROM-HROM	2.200×10^3	405/253	4.802×10^{-3} 1.960×10^{-3}	8.500×10^{-2} 4.630×10^{-2}	3.744×10^{-1} 2.580×10^{-1}	30,067

As for our reproductive test case, we combine the accuracy and efficiency results summarized in Table 5.2 into a Pareto plot, which shows the CPU time in seconds versus the average displacement MSE over all subdomains being coupled (Figure 5.5). It is clear from this figure that, by coupling a ROM to a FOM, it is possible to reduce error of a given reduced model by several orders of magnitude. Figure 5.5 reinforces the need to investigate ways to improve FOM-HROM and HROM-HROM efficiency and accuracy in future work.

Figure 5.6 shows the displacement, velocity and acceleration solutions at the final time $T = 1.0 \times 10^{-3}$ for several of the predictive models being evaluated: the single-domain ROM and the coupled FOM-HROM. The coupled solutions in Ω_1 and Ω_2 are shown in red and green, respectively. These are plotted on top of a single-domain FOM solution, shown in black. The reader can observe that the predictive single-domain ROM solution (Figure 5.6(a)) exhibits some spurious oscillations in the velocity and acceleration fields. In contrast, the FOM-HROM solution (Figure 5.6(b)) is smooth and in good agreement with the single-domain FOM.

While the results summarized in Table 5.2 and Figure 5.6 are promising, some work still needs to be done in improving the accuracy of the coupled model when the Schwarz alternating method is used to stitch together HROMs. We will explore strategies to do this in the context of a two-dimensional (2D) problem in a subsequent publication.

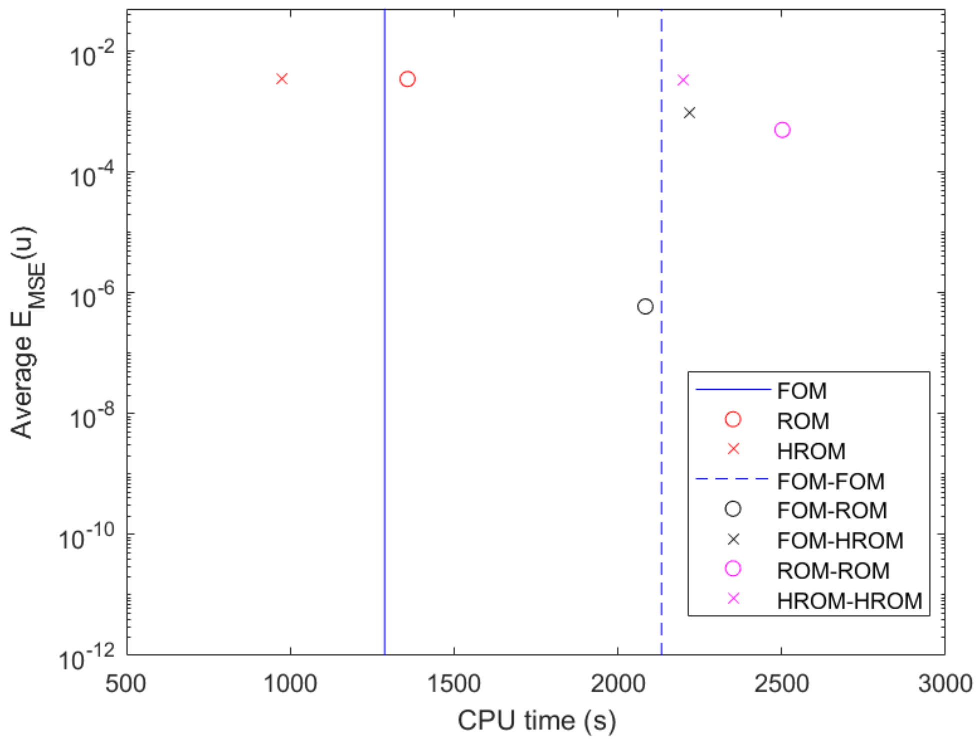
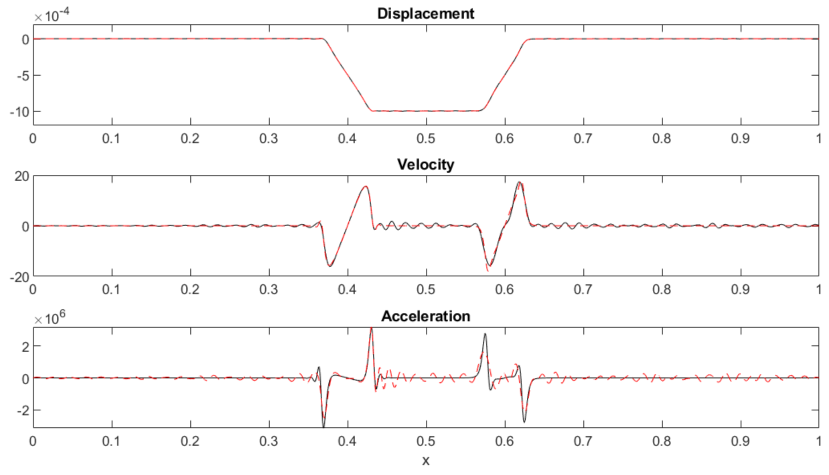


Fig. 5.5. *Pareto plot showing CPU time in seconds versus the average displacement MSE over all subdomains being coupled for the predictive couplings evaluated in Table 5.2.*

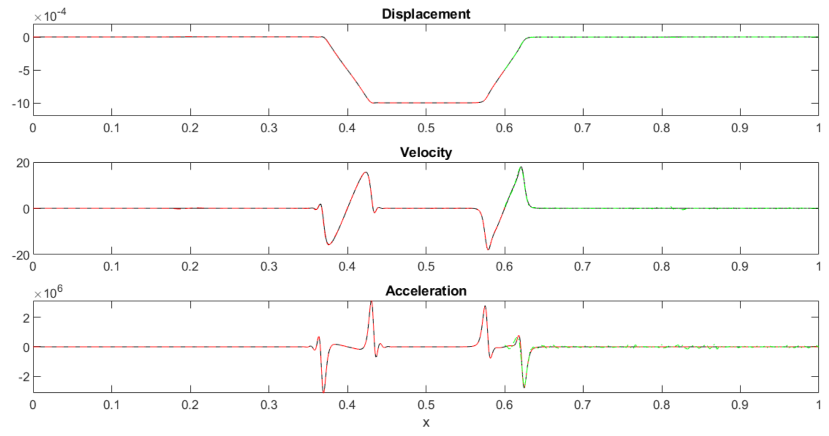
6. Summary and future work. In this paper, we described a methodology for coupling projection-based ROMs with each other and with conventional high-fidelity finite element models by means of the Schwarz alternating method. In this method, the physical domain is decomposed into two or more subdomains, and a sequence of subdomain-local problems is solved, with information propagating through carefully-constructed transmission boundary conditions posed on the subdomain boundaries. The method was formulated for both overlapping and non-overlapping domain decompositions in the context of a generic nonlinear solid dynamics problem. While our numerical experiments focused on couplings in which the same time-integrator and time-step is used in all subdomains being coupled, we emphasize that the method is capable of coupling not only disparate discretizations but also different time-integrators with disparate time-scales, as demonstrated in [36, 44].

The utility of the proposed coupling approach is two-fold. First, it provides a mechanism for enabling the "plug-and-play" integration of data-driven models into existing multi-scale and multi-physics coupling frameworks, with minimal intrusion (i.e., through the introduction of a simple outer loop around the two or more disparate models being coupled). Second, couplings such as those performed herein have the potential of improving the predictive viability of projection-based ROMs, by enabling the spatial localization of ROMs (via domain decomposition) and the online integration of high-fidelity information into these models (via FOM coupling).

Our numerical results for a 1D nonlinear wave propagation problem (a problem which poses a number of challenges for traditional POD-based model reduction approaches) are



(a) Predictive single-domain ROM



(b) Predictive FOM-HROM

FIG. 5.6. *Solutions to the predictive Rounded Square nonlinear wave propagation problem at the final time $T = 1.0 \times 10^{-3}$. The coupled solutions in Ω_1 and Ω_2 are shown in red and green, respectively; the single-domain FOM solution is shown in black. Whereas the velocity and acceleration components of the single-domain ROM is fraught with oscillations, the FOM-HROM coupled solutions agrees closely with the single-domain FOM solution.*

promising. These results demonstrate that the proposed methodology is capable of coupling disparate models without introducing numerical artifacts into the solution for both reproductive and predictive problems. Additionally, they show that the resulting couplings can be cost-effective when combined with hyper-reduction. Our conclusion that FOM-ROM couplings are particularly accurate for the problems considered is not surprising, given the sharp gradients present in the acceleration component of the solution. Features such as these are incredibly difficult to capture using POD modes alone, especially when they propagate dynamically in time. Our results suggest that some more work can be done in the future to try to improve the accuracy and efficiency of couplings involving HROMs. This task will be pursued in the context of 2D and three-dimensional (3D) problems with localized features in their solutions (e.g., shedding vortices behind a cylinder in a fluid simulation, failure and strain localization in a solid simulation), where more benefits are expected from both hyper-reduction as well as domain decomposition. Although attention herein was restricted to non-overlapping couplings, similar results were obtained with the overlapping version of our method, and are omitted simply for the sake of brevity; for a flavor of these results, the reader is referred to [44].

In future work, we plan to explore the following research directions:

- *Verifying that similar results can be obtained via explicit-explicit and implicit-explicit Schwarz couplings, and in the case when disparate time-steps are used to advance the solution in different subdomains.* Preliminary testing not reported here suggests that the same conclusions hold in the case of couplings involving disparate time-integrators/time-steps.
- *An extension of the proposed coupling framework to 2D and 3D problems.* A multi-dimensional implementation will require the development of transfer operators for defining the Schwarz transmission boundary conditions. We plan to investigate the usage of the Compadre toolkit [38] for this task. It is expected that the benefits of hyper-reduction will be far greater in 2D and 3D than in 1D.
- *Developing error indicator-based approaches for determining "optimal" domain decompositions and ROM/FOM placement.* We plan to build on the work of Bergmann *et al.* [2].
- *Implementing and testing the additive Schwarz variant of the proposed coupling method.* Please see Remark 3 and [11] for more information on this variant of the method. Preliminary studies suggest that additive Schwarz-based coupling has the potential to yield coupled models which can achieve speed-ups over a single-domain FOM when parallelized over the number of subdomains.
- *Examining snapshot collection strategies that do not require simulating the coupled high-fidelity model over the the entire domain.* Ideas such as oversampling [43] will be explored towards the goal of designing a workflow that can reuse existing modular codes for individual physics in an effort to perform minimally-intrusive modular couplings.
- *Extending the proposed approach to coupling scenarios that enable "on-the-fly" FOM-ROM switching and ROM adaptation.* A nice starting point for this research direction is the work of Corigliano *et al.* [5].
- *Performing an analysis of the method's theoretical convergence properties.* We plan to leverage some of our past work, which analyzed the method's convergence for FOM-FOM coupling in solid mechanics [35, 36].
- *Applying the proposed coupling methodology to problems other problems, including multi-physics problems.* We are particularly interested in problems involving FSI. We have begun to move in this direction by starting to explore multi-material coupling using Schwarz, as a proxy for a multi-physics problem,
- *Extending the proposed framework to enable the seamless coupling of other types of data-driven models, e.g., PINNs.* PINNs are well-suited for Schwarz-based couplings, as they have the notion of boundary conditions, which can be incorporated through the loss function being minimized [28, 29].

REFERENCES

[1] J. Baiges, R. Codina, and S. R. Idelsohn, *A domain decomposition strategy for reduced order models. Application to the incompressible Navier–Stokes equations*, Computer Methods in Applied Mechanics and Engineering, 267 (2013), pp. 23–42.

[2] M. Bergmann, A. Ferrero, A. Iollo, E. Lombardi, A. Scardigli, and H. Telib, *A zonal Galerkin-free POD model for incompressible flows*, Journal of Computational Physics, 352 (2018), pp. 301–325.

[3] M. Buffoni, H. Telib, and A. Iollo, *Iterative Methods for Model Reduction by Domain Decomposition*, Tech. Rep. 6383, INRIA Report, 2007.

[4] D. Cinquegrana, A. Viviani, and R. Donelli, *A hybrid method based on POD and domain decomposition to compute the 2-D aerodynamic flow field - incompressible validation*, 2011, pp. AIMETA 2011– XX Congresso dell'Associazione Italiana di Meccanica Teorica e Applicata, Bologna, ITA.

[5] A. Corigliano, M. Dossi, and S. Mariani, *Domain decomposition and model order reduction methods applied to the simulation of multi-physics problems in MEMS*, Computers & Structures, 122 (2013), pp. 113–127. Computational Fluid and Solid Mechanics 2013.

[6] A. Corigliano, M. Dossi, and S. Mariani, *Model order reduction and domain decomposition strategies for the solution of the dynamic elastic–plastic structural problem*, Computer Methods in Applied Mechanics and Engineering, 290 (2015), pp. 127–155.

[7] J. Côté, M. J. Gander, L. Laayouni, and S. Loisel, *Comparison of the Dirichlet-Neumann and Optimal Schwarz Method on the Sphere*, in Domain Decomposition Methods in Science and Engineering, T. J. Barth, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, T. Schlick, R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, eds., Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 235–242.

[8] A. de Castro, P. Kuberry, I. Tezaur, and P. Bochev, *a novel partitioned approach for reduced order model – finite element model (rom-fem) and rom-rom coupling.*

[9] C. Farhat, T. Chapman, and P. Avery, *Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models*, International journal for numerical methods in engineering, 102 (2015), pp. 1077–1110.

[10] D. Funaro, A. Quarteroni, and P. Zanolli, *An iterative procedure with interface relaxation for domain decomposition methods*, SIAM J. Numer. Anal., 25 (1988), pp. 1213–1236.

[11] M. J. Gander, *Schwarz methods over the course of time.*, ETNA. Electronic Transactions on Numerical Analysis [electronic only], 31 (2008), pp. 228–255.

[12] J.-C. Golaz, P. M. Caldwell, L. P. Van Roekel, M. R. Petersen, Q. Tang, J. D. Wolfe, G. Abeshu, V. Anantharaj, X. S. Asay-Davis, D. C. Bader, S. A. Baldwin, G. Bisht, P. A. Bogenschutz, M. Branstetter, M. A. Brunke, S. R. Brus, S. M. Burrows, P. J. Cameron-Smith, A. S. Donahue, M. Deakin, R. C. Easter, K. J. Evans, Y. Feng, M. Flanner, J. G. Foucar, J. G. Fyke, B. M. Griffin, C. Hannay, B. E. Harrop, M. J. Hoffman, E. C. Hunke, R. L. Jacob, D. W. Jacobsen, N. Jeffery, P. W. Jones, N. D. Keen, S. A. Klein, V. E. Larson, L. R. Leung, H.-Y. Li, W. Lin, W. H. Lipscomb, P.-L. Ma, S. Mahajan, M. E. Maltrud, A. Mametjanov, J. L. McClean, R. B. McCoy, R. B. Neale, S. F. Price, Y. Qian, P. J. Rasch, J. E. J. Reeves Eyre, W. J. Riley, T. D. Ringler, A. F. Roberts, E. L. Roesler, A. G. Salinger, Z. Shaheen, X. Shi, B. Singh, J. Tang, M. A. Taylor, P. E. Thornton, A. K. Turner, M. Veneziani, H. Wan, H. Wang, S. Wang, D. N. Williams, P. J. Wolfram, P. H. Worley, S. Xie, Y. Yang, J.-H. Yoon, M. D. Zelinka, C. S. Zender, X. Zeng, C. Zhang, K. Zhang, Y. Zhang, X. Zheng, T. Zhou, and Q. Zhu, *The doe e3sm coupled model version 1: Overview and evaluation at standard resolution*, Journal of Advances in Modeling Earth Systems, 11 (2019), pp. 2089–2129.

[13] M. Gunzburger, J. Peterson, and J. Shadid, *Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data*, Computer Methods in Applied Mechanics and Engineering, 196 (2007), pp. 1030–1047.

[14] H. Henky, *The law of elasticity for isotropic and quasi-isotropic substances by finite deformations*, J. Rheology, 2 (1931), pp. 169–176.

[15] C. Hoang, Y. Choi, and K. Carlberg, *Domain-decomposition least-squares Petrov–Galerkin (DD-LSPG) nonlinear model reduction*, Computer Methods in Applied Mechanics and Engineering, 384 (2021), p. 113997.

[16] P. Holmes, J. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, 1996.

[17] J. Hoy, I. Tezaur, and A. Mota, *The Schwarz alternating method for multiscale contact mechanics.* in Computer Science Research Institute Summer Proceedings 2021. E. Galvan and D. Smith, eds., Technical Report SAND2021-9886C, Sandia National Laboratories, 2021.

[18] C. Huang, K. Duraisamy, and C. Merkle, *Component-based reduced order modeling of large-scale complex systems*, Frontiers in Physics, 10 (2022).

[19] T. Hughes, *Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, Inc., Mineola, New York, 2000.

[20] L. Iaopichino, A. Quarteroni, and G. Rozza, *Reduced basis method and domain decomposition for elliptic problems in networks and complex parametrized geometries*, Computers & Mathematics

with Applications, 71 (2016), pp. 408–430.

[21] T. Iliescu and Z. Wang, *Are the snapshot difference quotients needed in the proper orthogonal decomposition?*, SIAM Journal on Scientific Computing, 36 (2014), pp. A1221–A1250.

[22] A. Iollo, G. Sambataro, and T. Taddei, *A one-shot overlapping schwarz method for component-based model reduction: application to nonlinear elasticity*, 2022.

[23] P. Kerfriden, O. Goury, T. Rabczuk, and S. Bordas, *A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics*, Computer Methods in Applied Mechanics and Engineering, 256 (2013), pp. 169–188.

[24] P. Kerfriden, J. C. Passieux, and S. P. A. Bordas, *Local/global model order reduction strategy for the simulation of quasi-brittle fracture*, International Journal for Numerical Methods in Engineering, 89 (2012), pp. 154–179.

[25] F. Kwok, *Neumann–Neumann Waveform Relaxation for the Time-Dependent Heat Equation*, in Domain Decomposition Methods in Science and Engineering XXI, J. Erhel, M. J. Gander, L. Halpern, G. Pichot, T. Sassi, and O. Widlund, eds., Cham, 2014, Springer International Publishing, pp. 189–198.

[26] S. Lall, P. Krysl, and J. E. Marsden, *Structure-preserving model reduction for mechanical systems*, Physica D: Nonlinear Phenomena, 184 (2003), pp. 304–318. Complexity and Nonlinearity in Physical Systems – A Special Issue to Honor Alan Newell.

[27] P. LeGresley, *Application of Proper Orthogonal Decomposition (POD) to Design Decomposition Methods*, PhD thesis, Stanford University, Stanford, CA, 2005.

[28] K. Li, K. Tang, T. Wu, and Q. Liao, *D3m: A deep domain decomposition method for partial differential equations*, IEEE Access, 8 (2020), pp. 5283–5294.

[29] W. Li, X. Xiang, and Y. Xu, *Deep Domain Decomposition Method: Elliptic Problems*, Proceedings of Machine Learning Research, 107 (2020), pp. 269–286.

[30] P. Lions, *On the Schwarz alternating method I.* in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1988.

[31] P. L. Lions, *On the Schwarz alternating method III: A variant for nonoverlapping subdomains*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, R. Glowinski, J. Périaux, and O. B. Widlund, eds., Society for Industrial and Applied Mathematics, 1990, pp. 202–223.

[32] D. J. Lucia, P. I. King, and P. S. Beran, *Reduced order modeling of a two-dimensional flow with moving shocks*, Computers & Fluids, 32 (2003), pp. 917–938.

[33] Y. Maday and E. Ronquist, *The reduced basis element method: application to a thermal fin problem*, SIAM J. Sci. Comput., 26 (2004), pp. 240–258.

[34] I. Maier and B. Haasdonk, *A Dirichlet-Neumann reduced basis method for homogeneous domain decomposition problems*, Applied Numerical Mathematics, 78 (2014), pp. 31–48.

[35] A. Mota, I. Tezaur, and C. Alleman, *The Schwarz alternating method in solid mechanics*, Computer Methods in Applied Mechanics and Engineering, 319 (2017), pp. 19–51.

[36] A. Mota, I. Tezaur, and G. Phlipot, *The Schwarz alternating method for dynamic solid mechanics*, Int. J. Numer. Meth. Engng, (2022), pp. 1–36.

[37] N. Newmark, *A method of computation for structural dynamics*, Journal of the Engineering Mechanics Division, 85 (1959), pp. 67–94.

[38] N. T. P. Kuberry, P. Bosler, *The Compadre Toolkit.* https://github.com/sandialabs/compadre, 2019.

[39] A. Radermacher and S. Reese, *Model reduction in elastoplasticity: proper orthogonal decomposition combined with adaptive sub-structuring*, Computational Mechanics, 54 (2014), pp. 677–687.

[40] S. Riffaud, M. Bergmann, C. Farhat, S. Grimberg, and A. Iollo, *The DGDD method for reduced-order modeling of conservation laws*, Journal of Computational Physics, 437 (2021), p. 110336.

[41] H. A. Schwarz, *Ueber einen Grenzübergang durch alternirendes Verfahren*, Zürcher u. Furrer, 1870.

[42] L. Sirovich, *Turbulence and the dynamics of coherent structures, part iii: dynamics and scaling*, Q. Appl. Math., 45 (1987), pp. 583–590.

[43] K. Smetana and T. Taddei, *Localized model reduction for nonlinear elliptic partial differential equations: localized training, partition of unity, and adaptive enrichment*, 2022.

[44] I. Tezaur, Y. Shimizu, A. Mota, and J. Barnett, *The Schwarz alternating method for ROM-ROM and ROM-FOM coupling*, 2022.

[45] P. Tiso and D. Rixen, *Discrete empirical interpolation method for finite element structural dynamics*, Topics in Nonlinear Dynamics, 1 (2013), pp. 203–212.

[46] M. Wicke, M. Stanton, and A. Treuille, *Modular bases for fluid dynamics*, ACM Trans. Graph., 28 (2009).

[47] P. Zanolli, *Domain decomposition algorithms for spectral methods*, CALCOLO, 24 (1987), pp. 201–240.

MULTI-FIDELITY TRAINING IN FEEDFORWARD SUPERVISED LEARNING NETWORKS

OWEN N. DAVIS [*], GIANLUCA GERACI [†], TIMOTHY M. WILDEY [‡], AND MOHAMMAD MOTAMED [§]

Abstract. In many scientific modeling applications large amounts of high fidelity data are unavailable or prohibitively expensive to generate. In many situations, a larger data set is assembled from disparate sources with varying accuracy and cost. The goal of this work is twofold. First, we want to explore strategies for training an accurate, precise, and cost effective neural network (NN), which exploits multifidelity (MF) data with any finite number of fidelity classes. For this task we develop novel extensions of two methods that have been previously studied for the case of two fidelity classes. The first method is a fusion of NNs that learns the direct (possibly non-linear) correlation between the low fidelity and high fidelity data, and the second method is a fusion of NNs which learns the (possibly non-linear) correlation between the low fidelity data and the residual of the two data types. For each of these methods, we consider two distinct extension strategies, which are formulated to favor two different types of MF data that often arise in MF modeling applications. Second, we want to obtain uncertainty estimates for the NN predictions and use these estimates to guide subsequent decisions, e.g. how to expand the MF data set to improve the reliability in the NN predictions. For this task we explore ensemble networks and dropouts. For our model problems, we consider a function approximation task with different fidelity data classes inherited from past multi-fidelity modeling work.

1. Introduction.

Using mathematical and numerical modeling to understand physical phenomena is a common practice in science and engineering. In situations where physical phenomena are hard to observe directly mathematical models can provide invaluable insight into the phenomena behavior. However, mathematical and numerical models, which capture the behavior of large scale multi-physics problems, often require considerable computational investment. This is especially problematic for tasks such as uncertainty quantification (UQ) that generally require many model evaluations. Fortunately, it is often possible to construct a cheap to evaluate high accuracy surrogate model by combining a small number expensive high fidelity (HF) approximations with a larger number of inexpensive, but low fidelity (LF), approximations. This practice is generally referred to as multifidelity (MF) surrogate modeling.

There are many existing strategies for MF surrogate modeling in the literature [4, 12, 16, 15, 17, 3, 22, 5, 9, 14]. Perhaps the most common approach is using a Bayesian auto-regressive Gaussian process to model the correlations between the different data fidelity classes [12, 5]. This method performs well for sparse HF data sets and provides built in uncertainty estimates, but can become relatively expensive as the problem dimension grows larger.

Recent work in MF surrogate modeling has focused on training a neural network (NN) as the surrogate model. When seeking to use a NN as a surrogate we have MF data for use in NN training, and the optimal way to leverage this MF training data has been the subject of considerable past research [4, 22, 17, 16, 15, 3, 9, 14]. The motivating principle behind the majority of this work has been to take advantage of the correlation between the LF and HF data. In [4] an approach known as the *comprehensive correction* is proposed. Here the correlation between the LF and HF data is assumed to be linear, and one seeks to use a NN to learn a multiplicative and or additive correction which transforms the LF data to the HF data. This method is limited by the assumption that the correlation between the LF and

[*]University of New Mexico, daviso@unm.edu
[†]Sandia National Laboratories, ggeraci@sandia.gov
[‡]Sandia National Laboratories, tmwilde@sandia.gov
[§]University of New Mexico, motamed@unm.edu

HF data is linear, and there have been several modifications of this method in an attempt to handle non-linear correlations. In [22] the multiplicative correction term is replaced by an unknown possibly non-linear function. In [17, 9] the multiplicative and additive corrections are absorbed into one unknown non-linear function and this general non-linear function is learned via a NN. We refer to this method as the Direct Non-linear NN (DNLNN). In [16, 15, 9] this general non-linear function is split into linear and non-linear components, and these are learned by separate NNs. In [3] it is pointed out that it is often computationally advantageous to reformulate the correlation between data types as a non-linear residual function and learn this instead. This method is referred to as the Residual Multifidelity NN (RMFNN). In [9] two additional methods are proposed. In the first, the LF outputs are considered latent variables in the NN, which learns the HF data. This is accomplished by inserting the LF outputs in an intermediate weight layer of the network. The second proposed method takes advantage of the connection between neural networks and Gaussian processes [8, 13, 18]. This method is referred to as GPmimic and seeks to mimic the action of a Gaussian process without incurring the same prohibitive computational cost. In [14] the authors use a transfer learning approach. A deep NN (DNN) is pre-trained on a data set that includes mostly LF data. This pre-trained DNN is then fine tuned on a target or HF data set. In order to enforce transfer learning during fine tuning the parameters in the first several layers of the pre-trained DNN are fixed and only the parameters in the final layers of the network are allowed to vary.

Once a strategy is chosen to harness the MF training data it is important to understand how this chosen strategy impacts the predictive uncertainty of the NN surrogate. Quantifying uncertainty in NN predictions, a task often called UQ for Machine Learning (ML), is an active research area, and several strategies have been proposed in the literature [10, 11, 1, 19]. Among these the most computationally inexpensive is called dropout [23], a procedure by which neurons are randomly dropped during training and or testing. Using dropout during training provides network regularization and helps prevent over-fitting, and using dropout during testing offers computationally inexpensive variance estimates. The interpretation of dropout as an ensemble of network architectures has also been investigated thoroughly in [2, 6].

In this work, we provide a direct comparison between the DNLNN and RMFNN methods using MF training data with exactly two fidelity classes. We compare the methods on the basis of predictive uncertainty (estimated via dropout), mean test error over an ensemble of network architectures, and cost of generating the MF training set. In addition, for each of the DNLNN and RMFNN methods, we motivate and provide two distinct strategies for extension to MF training sets with more than two fidelity classes.

The remainder of the article is organized as follows. In section 2, we formulate the MF surrogate modeling problem and discuss it in the case where the surrogate is taken to be an NN and the MF data set is assumed to have any finite number of fidelity classes. We provide mathematical descriptions of the DNLNN and RMFNN methods and discuss our proposed extensions of these methods for more than two fidelity classes. Moreover, we discuss our procedure for estimating the predictive uncertainty of the methods via dropout. In section 3, we compare the DNLNN and RMFNN methods for a function approximation task using MF data with both two and three fidelity classes.

2. Mathematical Background and Theoretical Contributions. In this section, we provide a formal mathematical description of MF surrogate modeling and specifically consider the case where a NN is used as the surrogate. Further, we provide rigorous explanations of the RMFNN and DNLNN methods and describe their generalizations to MF data sets with $M \geq 2$ fidelity classes. Finally, we discuss the way in which dropout is used in these methods to estimate their predictive uncertainty.

2.1. Multifidelity Surrogate Modeling. Suppose we want to create a surrogate model, S, for some quantity of interest, $Q : X \to Y$, where $X \subset \mathbb{R}^m$ and $Y \subset \mathbb{R}^n$. Further, suppose we have an HF approximation of Q called Q_{HF} and $M-1$ inexpensive LF approximations of Q called Q_{LF_i}, $i = 1 \cdots M-1$. When seeking to use a neural network as a surrogate for Q we view $\{(x, Q_{HF}(x))\}$, $\{(x, Q_{LF_1}(x))\}$, $\cdots$, $\{(x, Q_{LF_{M-1}}(x))\}$ as high fidelity and lower fidelity data sets for use in NN training. Our goal is to leverage this MF training data as to minimize $||Q(x) - S(x)||_{L_2}$ and $||Q(x) - S(x)||_{L_\infty}$ for all $x \in X$.

2.1.1. Hierarchical and Non-Hierarchical MF Data. Suppose we have a MF data set containing M different data fidelity classes, one high fidelity, HF, and $M-1$ lower fidelity, $\{LF_i\}_{i=1}^{M-1}$. In general, leveraging this MF training data means accurately learning the correlations between the various data fidelity classes. We argue that the procedure by which these correlations are learned should depend on the structure of the MF data set. To illustrate this we consider MF data which has either a hierarchical or non-hierarchical structure.

Hierarchical data is characterized by the different fidelity classes being ordered in their approximation quality of the true quantity of interest. This type of MF data often arises when all fidelity classes are generated by the same procedure. One example is MF data coming from a finite element approximation of a PDE where the fidelity of the data is controlled by the coarseness of the mesh. For data of this type we hypothesize that, for a fixed cost, we will achieve greater accuracy by learning the correlations between adjacent lower fidelity data classes in the hierarchy (as opposed to directly learning the correlations between each lower fidelity data class and the highest fidelity data class). It is often possible to learn these correlations between adjacent fidelity classes all at once, but from an implementation perspective it is generally easier to use a sequential procedure. Assume our M fidelity classes are hierarchical with the LF_i data being a better approximation to the true quantity of interest than the LF_j data whenever $i < j$. Given this set up we begin by learning the correlation between the LF_{M-1} and LF_{M-2} data. Next, we learn the correlation between the LF_{M-2} and LF_{M-3} data. We continue in this manner and terminate the process after learning the correlation between the LF_1 and HF data.

Dealing with non-hierarchical MF data in surrogate modeling applications is also gaining traction due to the abundance of cases in which this type of data often arises (the lower fidelity data classes relationships are often unknown). In this case, we do not want to use a sequential method which would impose a potentially incorrect hierarchy on the data fidelity classes. As an alternative, we leverage the MF data via what we refer to as a concatenation method. This method is defined by independently learning each correlation between the LF_i data and the HF data for all $i = 1, \cdots, M-1$. In learning these correlations independently we avoid potentially false assumptions concerning the relationships between the lower fidelity data classes.

2.1.2. Hyperparameter Tuning in NN Training. In addition to the importance of effectively leveraging the MF training data, there are many challenges associated with constructing an optimally performing NN. In the case of limited training data and finite computational resources, training a NN, which performs optimally, requires careful tuning of many hyperparameters including capacity, learning rate, initialization, and regularization. Optimal tuning of NN hyperparameters is often computationally prohibitive because it involves expensive continuous optimization over hyperparameter combinations. More often these hyperparameters are tuned manually or via some finite grid search over hyperparameter combinations. In addition to the hyperparameters contained in a given NN there are many different types of neural networks. In this work, we restrict ourselves to feedforward supervised learning and we point the reader to [7] for a comprehensive introduction to the theory of feedfoward NNs and their training procedures.

2.2. The RMFNN and DNLNN Methods. While the two fidelity class versions of the RMFNN and DNLNN methods have been well studied in [17, 3], no explicit method for extending the DNLNN and RMFNN methods for $M > 2$ fidelity classes or experimental results for these extensions are present in the literature. In this section, we provide general RMFNN and DNLNN formulations, which work for any number of fidelity classes and are capable of leveraging the MF training data using either a sequential or concatenation approach. In order to clearly explain these methods we (1) provide a detailed description of the structure of the training data, (2) point out the core difference between the RMFNN and DNLNN methods, (3) provide general training diagrams and pseudocode implementations of the methods, and (4) explain the dropout procedure used to quantify their predictive uncertainty.

2.2.1. The Training Data. In this work, we consider training data with the general structure pictured in Figure 2.1 Each fidelity class, LF_i, has its own unique collection of

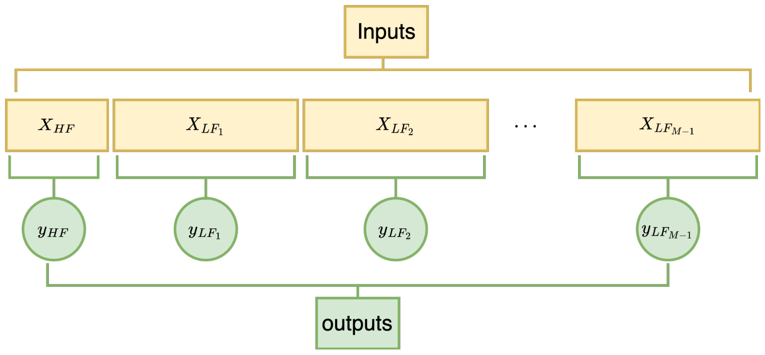


FIG. 2.1. *Training data for the DNLNN and RMFNN methods.*

samples, X_{LF_i}, and for each datapoint, $x \in X_{LF_i}$, we have an LF_i output, y_{LF_i}. We note here that the structure of the training data provided in Figure 2.1 is the most general form that can be used for all of our proposed methods. In certain experimental settings it may be possible or more convenient to take different forms of the training data. In particular, we point out the case in which lower fidelity data is readily generated at any point in the domain with negligible cost. In this case, a training data structure that has a nested sample input space across different fidelity classes can be implemented. However, this can be viewed as a special case of the training data in Figure 2.1.

2.2.2. Correlational Learning in the DNLNN and RMFNN Methods. The core difference between the DNLNN and RMFNN methods is the process by which they learn the correlation between different data fidelity classes. To illustrate this we consider learning the correlation between the LF_i and HF data.

The DNLNN learns this possibly non-linear correlation directly using the procedure outlined in Figure 2.2 and described in the following.

First, a neural network, NN_{LF_i}, is trained on the LF_i data. Next, a neural network, NN_{corr_i}, is trained to learn the map between LF_i and HF data. In particular, NN_{corr_i} takes inputs $x \in X_{HF}$ and an LF_i prediction at x, $NN_{LF_i}(x)$. It then attempts to learn the mapping between these inputs and the HF output, $y_{HF}(x)$. Notice that the training set

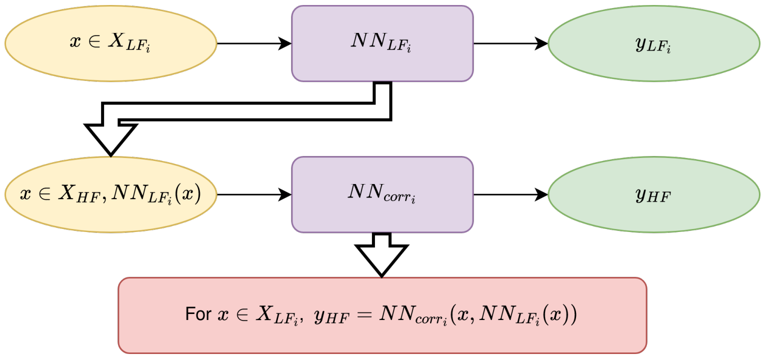

$x \in X_{LF_i}$
NN_{LF_i}
y_{LF_i}
$x \in X_{HF}, NN_{LF_i}(x)$
NN_{corr_i}
y_{HF}
For $x \in X_{LF_i}$, $y_{HF} = NN_{corr_i}(x, NN_{LF_i}(x))$

FIG. 2.2. *The procedure by which the DNLNN method learns the correlation between LF_i and HF data.*

for NN_{corr_i} has size equal to the number of available HF training samples, so in a general MF surrogate modeling situation NN_{corr_i} is trained on relatively sparse data.

The RMFNN learns the correlation between fidelity classes in a slightly different way. It casts this problem as learning the possible non-linear correlation between the LF_i data and the residual between the HF and LF_i data. This procedure is pictured in Figure 2.3. The first step is the same as it was in the DNLNN correlational learning procedure. The

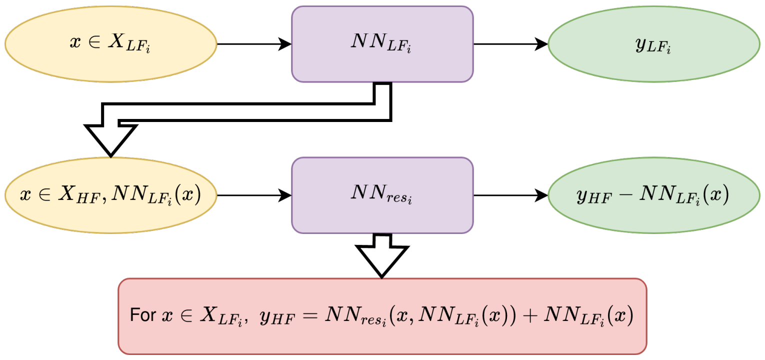

$x \in X_{LF_i}$
NN_{LF_i}
y_{LF_i}
$x \in X_{HF}, NN_{LF_i}(x)$
NN_{res_i}
$y_{HF} - NN_{LF_i}(x)$
For $x \in X_{LF_i}$, $y_{HF} = NN_{res_i}(x, NN_{LF_i}(x)) + NN_{LF_i}(x)$

FIG. 2.3. *The procedure by which the RMFNN method learns the correlation between LF_i and HF data.*

difference is evident in the second step where we train the neural network NN_{res_i}. The training inputs to this network are the same as they were for NN_{corr_i} in the DNLNN method, but we now train with respect to the difference between the HF output, $y_{HF}(x)$, and the LF_i prediction, $NN_{LF_i(x)}$. Once NN_{res_i} is trained, the HF model can be evaluated by summing the two contributions, as illustrated in the red box of Figure 2.3.

2.2.3. Training and Testing Procedures. In this section, we present training/testing diagrams for the DNLNN and RMFNN methods. For each of these methods we present one training procedure that leverages the MF training data using a sequential approach and one that leverages the MF data using a concatenation approach. The interested reader can also find full pseudo-code implementations of these methods in Appendix A.

DNLNN Sequential. In Figure 2.4 the network training and prediction diagram for the DNLNN sequential method is pictured. We note that, when we leverage the MF training data using a sequential approach we learn the correlations between the data fidelity in a sequential manner. In accordance with this goal, we begin by training a single neural

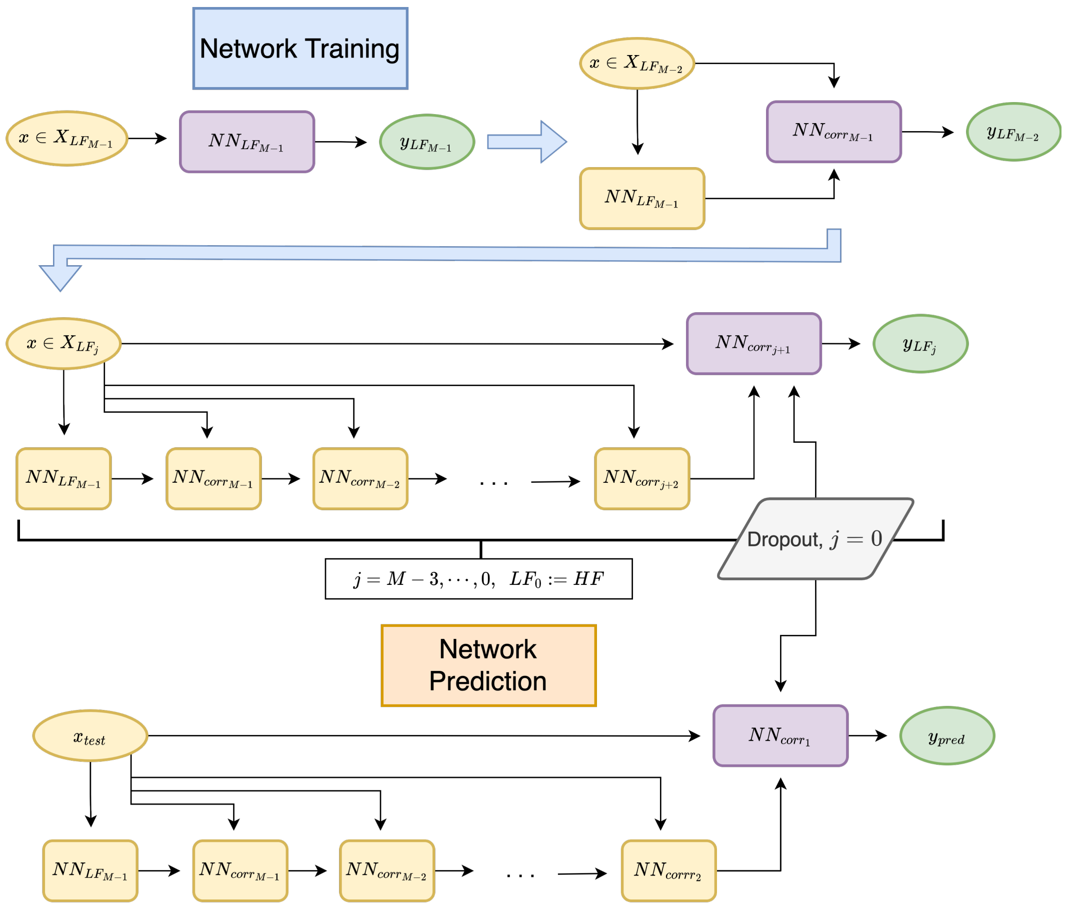


FIG. 2.4. *DNLNN training on a MF data set with M fidelity classes leveraged using a sequential approach.*

network, $NN_{LF_{M-1}}$, to learn the LF_{M-1} data. Next, we train a neural network, $NN_{corr_{M-1}}$, to learn the correlation between the LF_{M-1} and LF_{M-2} data. This correlational learning is accomplished in the manner previously explained for the DNLNN method. From here, we continue to train neural networks that allow us to sequentially move up the fidelity class hierarchy. The general step pictured in the diagram is for the training of $NN_{corr_{j+1}}$, which learns the correlation between the LF_{j+1} and LF_j data. Notice that the direct inputs are a native LF_j input, $x \in X_{LF_j}$, and an LF_{j+1} prediction at x. This LF_{j+1} prediction is handed down from the previously trained $NN_{corr_{j+2}}$, which in turn requires the prediction of $NN_{corr_{j+3}}$, and so on and so forth. The training outputs for $NN_{corr_{j+1}}$ are $y_{LF_j}(x)$ for each $x \in X_{LF_j}$. This general training process is terminated with the training of NN_{corr_1}, which learns the correlation between the LF_1 and HF data. From here NN_{corr_1} issues network predictions. Notice though that to issue a network prediction all networks trained during the training procedure need to be evaluated at the test input. We refer to this as a compound network prediction. In Figure 2.4, we also indicate where we apply dropout to quantify predictive uncertainty, which happens in the training and testing of NN_{corr_1}.

DNLNN Concatenation. The network training and prediction for the DNLNN concatenation method is pictured in Figure 2.5. The method proceeds in a slightly different way

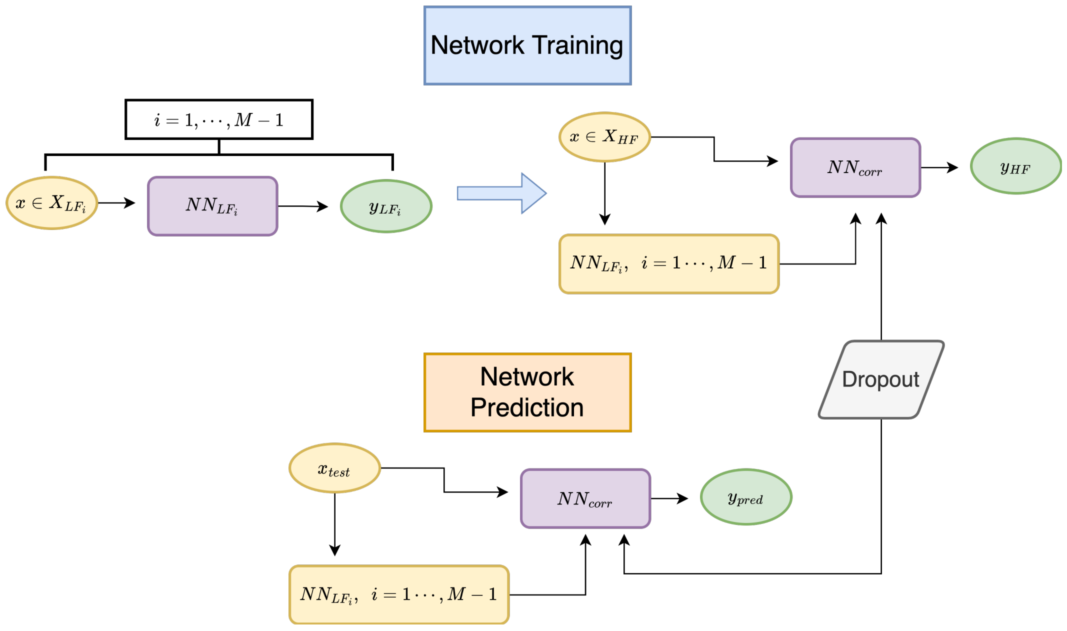


FIG. 2.5. *DNLNN training on a MF training set with M fidelity classes leveraged using a concatenation approach.*

than it did in the sequential approach. Recall that when leveraging the MF training data using a concatenation approach we independently learn all the correlations between the LF_i, $i = 1, \cdots, M-1$, and HF data. Given this goal, we begin by training NN_{LF_i} to learn the LF_i data for all $i = 1, \cdots, M-1$. Once these $M-1$ neural networks are trained, we train a single neural network, NN_{corr}. Notice that the training inputs are a native HF input, $x \in X_{HF}$, and an LF_i prediction at x, $NN_{LF_i}(x)$ for all $i = 1, \cdots, M-1$. The training outputs are the HF outputs, $y_{HF}(x)$ for each $x \in X_{HF}$. Given this training data, NN_{corr} can be understood as learning the general correlation between all lower fidelity data and the HF data. Once NN_{corr} is trained it is used to issue predictions on our test inputs. Notice that evaluating NN_{corr} requires an evaluation of NN_{LF_i} for all $i = 1, \cdots, M-1$, so once again we use a compound network prediction. Take note as well that dropout is being applied to NN_{corr} in both training and testing.

RMFNN Concatenation. In Figure 2.6 the training testing procedure for the RMFNN leveraging data using a concatenation approach is pictured. Following the training diagram, we start by training a neural network, NN_{LF_i}, to learn the LF_i data for all $i = 1, \cdots, M-1$. Next, we train a neural network, NN_{res_i}, to learn the correlation between the LF_i data and the residual between the HF and LF_i data for all $i = 1, \cdots, M-1$. This correlational learning is done via the RMFNN strategy previously explained. From here the trained NN_{res_i} is used to generate synthetic HF outputs for all $x \in X_{LF_i}$. After all synthetic data generation is complete, we have an HF output for every input in the MF training set. We finish by training NN_{HF} to learn this enlarged HF training set, and then use NN_{HF} to issue network predictions. Notice the difference in workflow from the DNLNN methods. Here a single neural network trained on an enlarged synthetic HF data set is used to issue final network predictions. We refer to this as synthetic data generation network prediction. As indicated in Figure 2.6, dropout is used in both the training and testing of NN_{HF}.

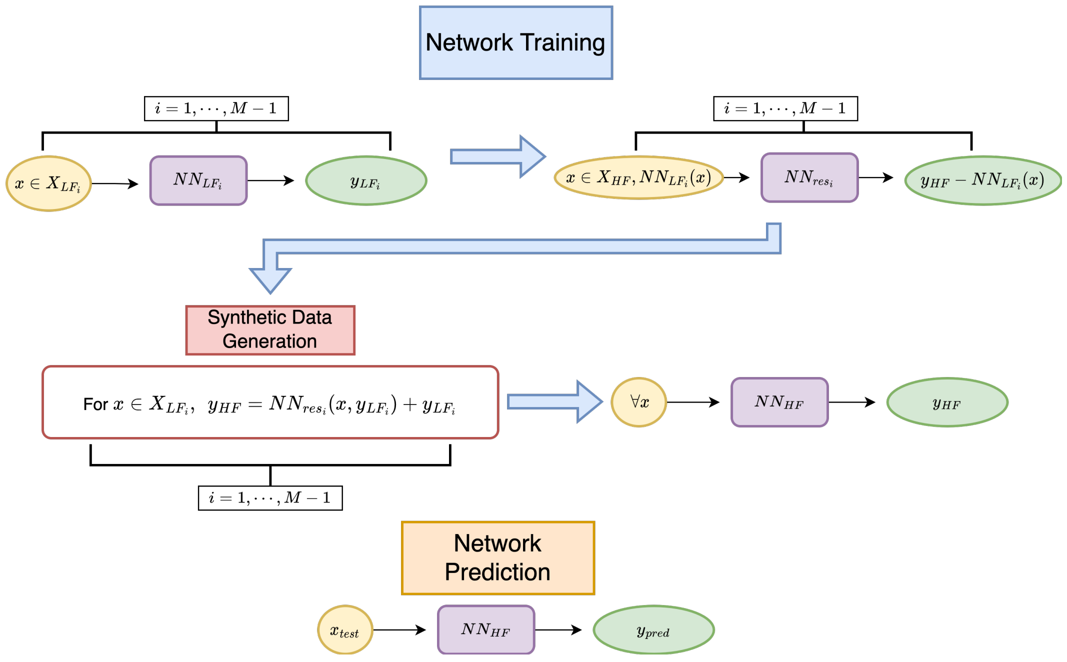


FIG. 2.6. *RMFNN training procedure for a MF data set with M fidelity classes leveraged using a concatenation approach.*

RMFNN Sequential. In Figure 2.7, the training procedure used for the RMFNN method with training data leveraged via a sequential approach is pictured. Following the training diagram we begin by training a neural network, NN_{LF_i}, to learn the LF_i data for all $i = 1, \cdots, M-1$. Next, for $i = M-1, \cdots, 1$ we alternate between (1) training a neural network, NN_{res_i}, which learns the correlation between the LF_i data and the residual between the LF_{i-1} and LF_i data, and (2) using the trained NN_{res_i} to generate synthetic LF_{i-1} outputs for all $x \in X_{LF_i}$. Once this sequential procedure is complete we have an HF output for all inputs in the original MF training set. We finish by training a neural network, NN_{HF}, on this enlarged HF training set, and we use the trained NN_{HF} to issue network predictions. Notice again that we have made use of a synthetic data generation network prediction. Just as in the RMFNN concatenation method dropout is applied during the training and testing of NN_{HF}

RMFNN Compound Approach. In [3], an alternative workflow for the RMFNN method is presented for the case of two fidelity classes. This alternative workflow uses a compound network prediction instead of the synthetic data generation network prediction used in the previously presented RMFNN methods. For our explanation of this two-fidelity method we label our fidelity classes LF_1 and HF. When the MF training set contains only two fidelity classes the concatenation and sequential methods are the same, so to understand the RMFNN compound workflow the reader can refer to the $M = 2$ version of either Figure 2.6 or Figure 2.7. In the standard $M = 2$ version of the RMFNN method, we start by training a neural network, NN_{LF_1}, to learn the LF_1 data. Then we train a neural network, NN_{res_1}, to learn the correlation between the LF_1 data and the residual between the HF and LF_1 data. Once trained, NN_{res_1} is used to generate synthetic HF outputs for all $x \in LF_1$. In the alternate workflow this synthetic data generation step is removed. Instead we use a combination of NN_{res_1} and NN_{LF_1} to directly issue network predictions,

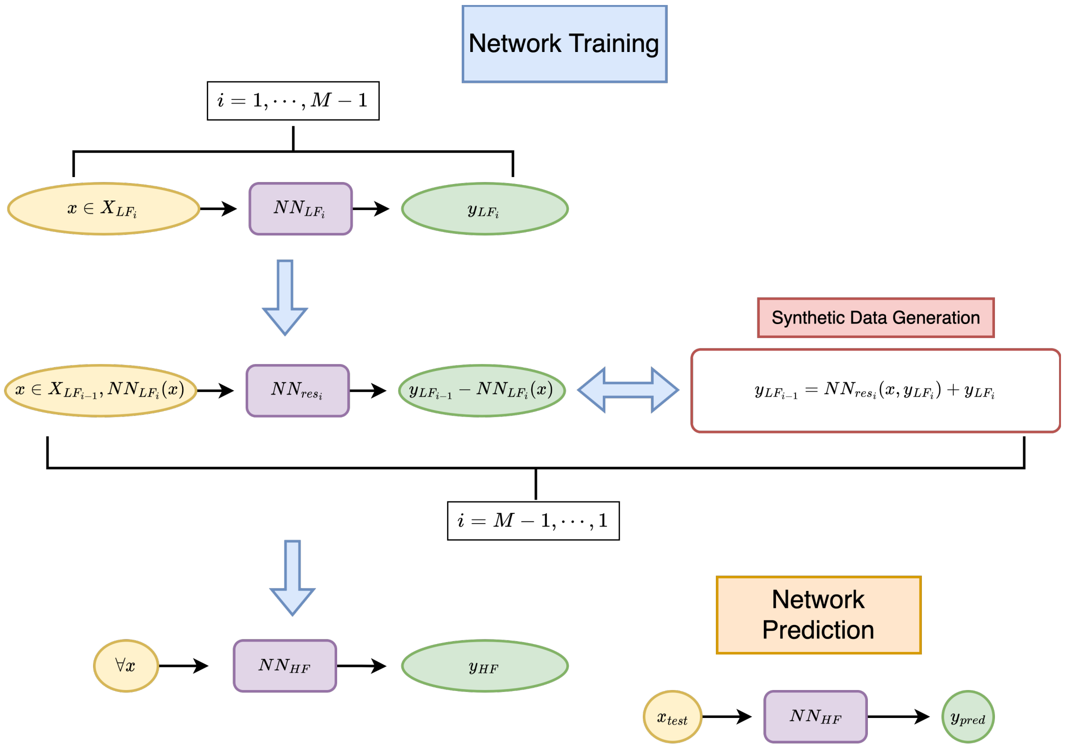


FIG. 2.7. *RMFNN training procedure for a MF data set with M fidelity classes leveraged using a sequential approach.*

$y_{pred} = NN_{res_1}(x_{test}, NN_{LF_1}(x_{test})) + NN_{LF_1}(x_{test})$. In this alternative workflow we apply dropout to the training and testing of NN_{res_1}.

For clarity we provide Table 2.1, which contains a compact summary of all proposed methods from this section.

TABLE 2.1
Proposed NN Methods

Num. Fidelity Classes (M)	Method	Data Leveraging: sequential (S) or concatenation (C)	Prediction Workflow
$M = 2$	RMFNN	S and C equivalent	Synthetic Data Gen.
			Compound
	DNLNN		Compound
$M > 2$	RMFNN	S	Synthetic Data Gen.
		C	
	DNLNN	S	Compound
		C	

2.3. Estimating Predictive Uncertainty. We quantify the predictive uncertainty in our NN methods via dropout. A general description of dropout as both a regularization and UQ strategy is presented in section 1 along with the references therein. In this section we offer a description of our dropout scheme used for the previously presented methods. In each method's network diagram it was indicated to which neural networks dropout would be applied. In each instance dropout was applied during both training and testing. The use of dropout during both training and testing was inspired by the findings in [21] where it was pointed out that to promote accurate individual test predictions (as opposed to only an accurate mean prediction over many model evaluations) required that dropout be used in training as well as testing. All neural networks in our methods are composed of linear layers and ReLU activation functions. We apply dropout after each linear layer (except the output layer), and we assign all neurons a fixed dropout probability of 1/20. This probability is a hyperparameter that was manually tuned. From our tests, we observed that the value 1/20 provided a favorable balance of network regularization during training and sufficient perturbation of the network architecture during testing.

3. Numerical Results.

3.1. Training Data Structure. As mentioned in section 2.2.1, there are variety of training data structures that are applicable to our proposed methods. For our numerical

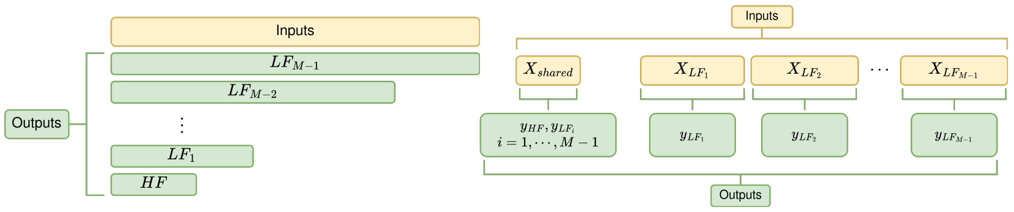


FIG. 3.1. *On the left is the training data structure for RMFNN compound and sequential methods, and on the right is the training data structure for RMFNN concatenation method.*

results the DNLNN sequential and concatenation methods use the training data structure outlined in section 2.2.1. For the RMFNN compound, concatenation, and sequential methods we use the training data structure outlined in Figure 3.1. This data structure is the one used in [3] and provides a simplification of the RMFNN training procedures outlined in Figure 2.6 and Figure 2.7. The left graphic in Figure 3.1 is the training data structure used for the RMFNN sequential method. Recall that in the RMFNN sequential training procedure, Figure 2.7, the network(s) NN_{res_i}, $i = 1, \cdots, M-1$, learn the mapping $(x \in X_{LF_{i-1}}, NN_{LF_i}(x)) \mapsto (y_{LF_{i-1}} - NN_{LF_i}(x))$. Now, since the data fidelity classes are nested in the input space we can replace the network prediction, $NN_{LF_i}(x)$, with the available lower fidelity sample, $y_{LF_i}(x)$. The right graphic in Figure 3.1 is the training data structure for the RMFNN concatenation method. Recalling the training procedure, Figure 2.6, the network(s) NN_{res_i}, $i = 1, \cdots, M-1$, learn the mapping $(x \in X_{HF}, NN_{LF_i}(x)) \mapsto (y_{HF} - NN_{LF_i}(x))$. In this new training structure there is a sample input space, X_{shared}, common to all fidelity classes which numbers the amount of available HF samples. The training of the network(s), NN_{res_i}, are now done on this shared input space, and the network prediction, $NN_{LF_i}(x)$, can be replaced by the available lower fidelity output, $y_{LF_i}(x)$. For the RMFNN compound numerical results either of the the training data structures in Figure 3.1 can be used as they are equivalent in the case of two fidelity classes.

3.2. Test Problem: 4D Function Approximation, M = 2.

In order to compare the $M = 2$ DNLNN and composite RMFNN methods we consider the task of approximating the following function using MF training data.

$$f_{HF}(x_1,x_2,x_3,x_4) = \frac{x_1}{2}\left[\sqrt{1+(x_2+x_3^2)\frac{x_4}{x_1^2}}-1\right] + (x_1+3x_4)\exp(1+\sin(x_3)) \tag{3.1}$$

where $(x_1,x_2,x_3,x_4) \in (0,1)^4$. This function has been used in past multifidelity modeling work [20]. Since we are considering only two fidelity classes we call them HF and LF. In order to generate HF training data we sample directly from Equation (3.1).

We hypothesize that the nature of the correlation between the LF and HF data in the MF training set will heavily impact the performance of the RMFNN and DNLNN methods, so we choose to conduct this approximation task on two different MF training sets. In one of them the LF data is linearly correlated with the HF data. To generate this linearly correlated LF data we sample from the following function:

$$f_{LF,lin}(x_1,x_2,x_3,x_4) = \left[1+\frac{\sin(x_1)}{10}\right] f_{HF}(x_1,x_2,x_3,x_4) - 2x_1 + x_2^2 + x_3^2 + 0.5 \tag{3.2}$$

which is linear in f_{HF} and appeared in past MF modeling work [24]. In the other MF training set the LF data is non-linearly correlated with the HF data. To generate this LF training data we sample from

$$f_{LF,nonlin}(x_1,x_2,x_3,x_4) = \left[1+\frac{\sin(\tilde{x_1})}{10}\right] \left[f_{HF}(\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_4})\right]^{3/2} - 2\tilde{x_1} + \tilde{x_2}^2 + \tilde{x_3}^2 + 0.5$$

where

$$\begin{pmatrix}\tilde{x_1} & \tilde{x_2} & \tilde{x_3} & \tilde{x_4}\end{pmatrix} = \begin{pmatrix}\cos(3\pi/2) & 0 & -\sin(3\pi/2) & 0\\ 0 & 1 & 0 & 0\\ \sin(3\pi/2) & 0 & \cos(3\pi/2) & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\\ x_4\end{pmatrix} \tag{3.3}$$

where $f_{LF,nonlin}$ is a non-linear function of f_{HF}.

All the network training data is generated through random uniform sampling of the functions above. In particular, our sampling workflow is as follows. For a given fidelity class we generate random uniform samples of each coordinate numbering the desired amount of data points in that fidelity class. Using these random samples we generate a random grid on which we evaluate the desired function.

Our goal is to compare the best possible DNLNN and the best possible RMFNN with respect to the cost of generating the MF training data. To verify that the DNLNN and RMFNN methods are leveraging the MF data effectively we also consider a neural network, NN_{HF}, trained on a data set comprised of only HF data with the same cost as the MF training set used for the DNLNN and RMFNN methods.

To generate these data sets, we fix a cost ratio, $C_{HF}/C_{LF} = 20/1$, where C_{HF} and C_{LF} are respectively the cost to generate HF and LF samples, and a fidelity ratio, $N_{HF}/N_{LF} = 1/5$, where N_{HF} and N_{LF} are respectively the number of HF and LF samples present in the training set. This cost and fidelity ratio simulate a common MF surrogate modeling situation where we have a small set of expensive HF data and a much larger set of inexpensive LF data. With these cost and fidelity ratios fixed we measure the cost of an MF training set in HF equivalent evaluations.

For each numerical experiment, (1) linearly correlated LF data and (2) non-linearly correlated LF data, we have four different test cases, A, B, C, and D, each of which considers a training set of different cost. The composition of the training sets for each of these test cases and for each of the $RMFNN$, $DNLNN$, and NN_{HF} methods are presented in Table 3.1.

TABLE 3.1
Training set composition for DNLNN, RMFNN, and NN_{HF}.

Test Case	Cost	$RMFNN$, $DNLNN$ Training Set	NN_{HF} Training Set
A	10	$N_{HF} = 8$, $N_{LF} = 40$	$N_{HF} = 10$, $N_{LF} = 0$
B	50	$N_{HF} = 40$, $N_{LF} = 200$	$N_{HF} = 50$, $N_{LF} = 0$
C	100	$N_{HF} = 80$, $N_{LF} = 400$	$N_{HF} = 100$, $N_{LF} = 0$
D	500	$N_{HF} = 400$, $N_{LF} = 2000$	$N_{HF} = 500$, $N_{LF} = 0$

In comparing the RMFNN and DNLNN methods we desire estimates of their predictive uncertainty. In order to obtain these estimates we use dropout as outlined in section 2.3. Dropout is also applied to NN_{HF} in the same manner with the same drop probability of 1/20. Furthermore, in section 2 the necessity of tuning the hyperparameters of NNs was discussed. Since for a given MF training set we want to compare the best possible RMFNN, DNLNN, and NN_{HF} we manually tune their hyper parameters independently to provide close to optimal performance. Now with dropout active, as described in section 2.3, we evaluate each model 1000 times on an independent test set. From these 1000 model evaluations we collect the mean L_∞ test error, standard deviation in the L_∞ test error, mean L_2 test error, and the standard deviation in the L_2 test error. The results for both the linearly correlated LF training data and the non-linearly correlated LF training data are displayed in Figure 3.2 and Figure 3.3.

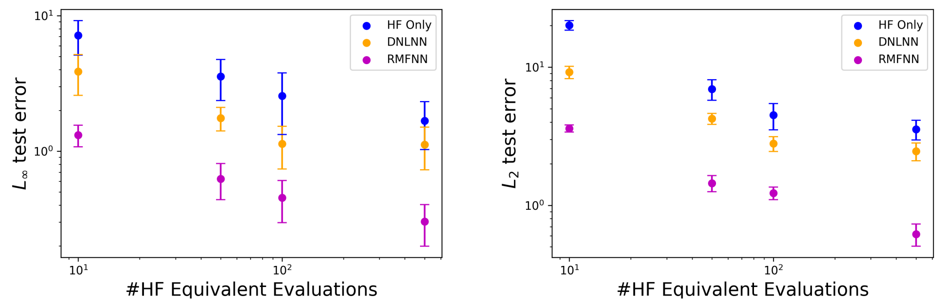


FIG. 3.2. *L_∞ test error and standard deviaition (left) and L_2 test error and standard deviation (right) for HF only (blue), DNLNN (orange), and RMFNN (magenta) surrogates on an MF training set where the LF data is linearly correlated with the HF data.*

Examining Figure 3.2 we see that both the RMFNN and DNLNN methods provide improved mean L_∞ and mean L_2 test error when compared to NN_{HF}. We also notice that for a given MF training set the standard deviation in predictions issued by DNLNN and RMFNN are marginally lower than those for NN_{HF}. In comparing the RMFNN and DNLNN methods, we see that RMFNN is considerably better in terms of mean test error, and comparable in terms of standard deviation. The relatively high performance of the RMFNN method on this test problem most likely stems from the residual between the HF

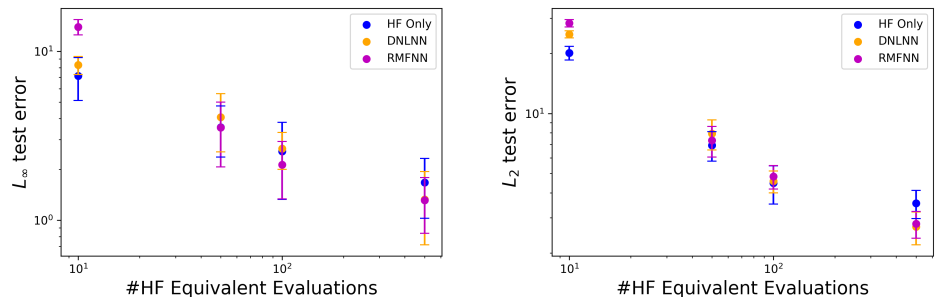


FIG. 3.3. *L_∞ test error and standard deviaition (left) and L_2 test error and standard deviation (right) for HF only (blue), DNLNN (orange), and RMFNN (magenta) surrogates on an MF training set where the LF data is non-linearly correlated with the HF data.*

and LF data being uniformly small in magnitude over the full domain. Because of this RMFNN is able to approximate this residual very accurately with sparse training data [3].

Examining Figure 3.3, we see very little benefit to the RMFNN or DNLNN methods in terms of mean test error or standard deviation when compared to NN_{HF}. The reason for this is most likely due to the fact that (1) the direct correlation between the LF and HF data and (2) the correlation between the LF data and the residual of the HF and LF data are both very non-linear. The performance of DNLNN and RMFNN depends on accurately learning (1) and (2) respectively using a small number of training samples equal to the amount of HF data available in the MF training set. In the case of a very difficult mapping and extreme data sparsity these methods suffer. This assessment is supported by the results presented in Figure 3.3. The only MF data set for which we see a benefit to the DNLNN and RMFNN methods is that which costs 500 HF equivalent evaluations and includes 400 samples of strictly HF training data. With this larger set of HF data the DNLNN and RMFNN methods are able to learn their respective non-linear correlation maps and issue predictions which improve upon the predictions of NN_{HF}.

3.3. Test Problem: 4D function approximation, M = 3. Here we consider the same four dimensional function approximation task as in section 3.2, but instead of using MF training data with two fidelity classes we consider three fidelity classes. We call these HF, LF_1, and LF_2. Moreover, we consider two distinct MF training sets. We hypothesize that one of them is structured to favor sequential data harnessing while the other is structured to favor the concatenation data harnessing. Our goal in this numerical experiment is to compare the best possible DNLNN sequential, RMFNN sequential, DNLNN concatenation, and RMFNN concatenation methods with respect to the structure of the MF training data. Moreover, to verify that the MF methods are effectively leveraging the MF training data we also compare to a single fidelity neural network, NN_{HF}, trained on an exclusively HF training set of cost equal to that of the MF training sets.

Sequential Favoring MF Training Set. In this first MF training set we take the HF data to be random samples of the function f_{HF} in Equation (3.1). For the LF_1 data we sample from the function $f_{LF,lin}$ in Equation (3.2). For the LF_2 data we sample from the function:

$$f_{LF_2,seq}(x_1,x_2,x_3,x_4) = \left[1+\frac{\sin(\tilde{x_1})}{10}\right]\left[f_{HF}(\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_4})\right]^{3/2} - 2\tilde{x_1} + \tilde{x_2}^2 + \tilde{x_3}^2 + \left[0.5+\frac{\sin(x_1)}{20}\right] f_{LF}(x_1,x_2,x_3,x_4) + 0.5$$

where $(\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_4})$ is the same as in Equation (3.3). Notice now that the LF_1 data is linearly correlated with the HF data whereas the LF_2 data is non-linearly correlated with the HF data, but linearly correlated with the LF_1 data. Because of this we expect it to be easiest to first learn the correlation between the LF_2 and LF_1 data, and then learn the correlation between the LF_1 and HF data. This is a procedure that theoretically favors the sequential versions of the RMFNN and DNLNN methods.

Concatenation Favoring MF Training Set. In this MF training set we take the HF data to be random samples of the function f_{HF} in Equation (3.1). For the LF_1 data we sample from the function $f_{LF,lin}$ in Equation (3.2). For the LF_2 data we sample from the function:

$$f_{LF_2,concat}(x_1,x_2,x_3,x_4) = \left[1+\frac{\sin(\tilde{x_1})}{10}\right]\left[f_{LF}(\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_4})\right]^{3/2} - 2\tilde{x_1} + \tilde{x_2}^2 + \tilde{x_3}^2 + \left[0.5+\frac{\sin(x_1)}{20}\right] f_{HF}(x_1,x_2,x_3,x_4) + 0.5$$

where $(\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_4})$ is the same as in Equation (3.3). Notice now that the LF_1 data is linearly correlated with the HF data whereas the LF_2 data is non-linearly correlated with the LF_1 data, but linearly correlated with the HF data. Because of this we expect it to be easier to learn direct correlations between the LF_1 and HF data and between the LF_2 and HF data than it is to learn the correlation between the LF_1 and LF_2 data. This is a procedure that theoretically favors the concatenation versions of the RMFNN and DNLNN methods.

For each type of MF data, (1) sequential favoring and (2) concatenation favoring, we consider a single test case with training data which costs 48 high fidelity equivalent evaluations. The composition of our training set (in terms of the number of samples from each fidelity class) will be based on (1) the cost to generate a sample from each fidelity class and (2) the ratio of number of HF to number of LF_1 to number of LF_2 samples in the training set. The costs of generating a data sample (measured in high fidelity equivalent evaluations) for each fidelity type are $C_{HF} = 1$, $C_{LF_1} = 1/10$, and $C_{LF_2} = 1/20$ where C_{HF}, C_{LF_1}, and C_{LF_2} are respectively the cost to generate a single sample of HF, LF_1, and LF_2 data. The ratio between number of HF samples (N_{HF}) and number of LF_1 samples (N_{LF_1}) is $N_{HF}/N_{LF_1} = 1/5$, and the ratio between the number of HF samples and the number of LF_2 samples (N_{LF_2}) is $N_{HF}/N_{LF_2} = 1/10$. The specific composition of the training sets used for DNLNN sequential and concatenation, RMFNN sequential and concatenation, and NN_{HF} are pictured in Table 3.2.

In comparing the RMFNN and DNLNN $M = 3$ sequential and concatenation methods we desire estimates for their predictive uncertainty. In order to obtain these estimates we use dropout as outlined in section 2.3. Dropout is also applied to NN_{HF} in the same manner with the same drop probability of 1/20. Furthermore, in section 2 the necessity of

Table 3.2
Training set composition for DNLNN (sequential and concatenation), RMFNN (sequential and concatenation), and NN_{HF}.

Cost	RMFNN and DNLNN (seq. and concat.) Training Set	NN_{HF} Training Set
48	$N_{HF} = 30$, $N_{LF_1} = 150$, $N_{LF_2} = 300$	$N_{HF} = 48$, $N_{LF_1} = 0$, $N_{LF_2} = 0$

tuning the hyperparameters of NNs was discussed. Since we want to train the best possible RMFNN, DNLNN, and NN_{HF} we manually tune their hyperparameters independently to provide close to optimal performance. Once each model is trained it is evaluated 1000 times on an independent test set with dropout activated. From these 1000 model evaluations we collect mean L_2 test error, standard deviation in L_2 test error, mean L_∞ test error, and standard deviation in L_∞ test error. The results of this numerical experiment are pictured in Figure 3.4 and Figure 3.5 below.

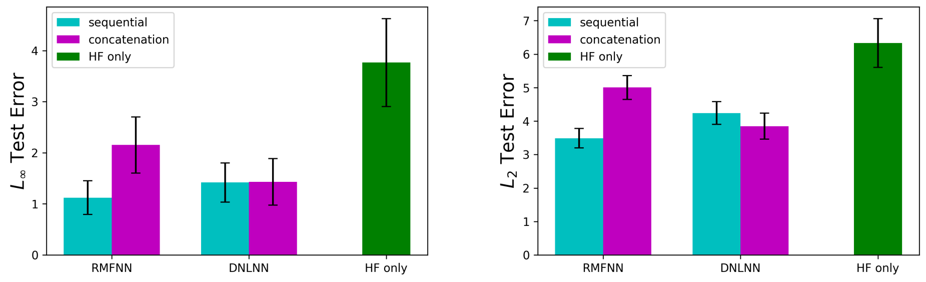


Fig. 3.4. *Mean L_2 test error and standard deviation (right) and mean L_∞ test error and standard deviation (left) for the RMFNN and DNLNN sequential methods (cyan) and concatenation methods (magenta) trained on the sequential favoring MF training set. The same statistics for a NN trained on only HF data is also displayed (green).*

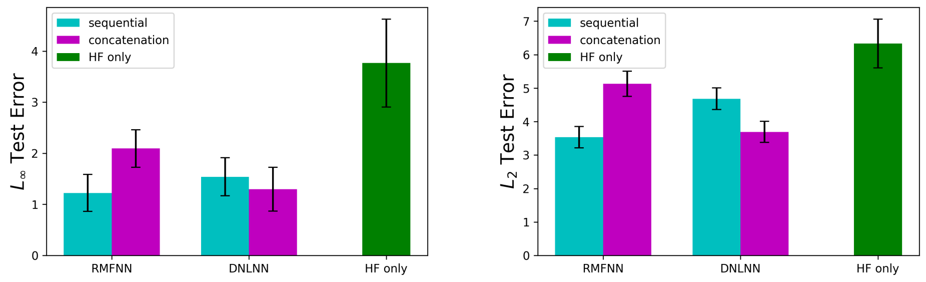


Fig. 3.5. *Mean L_2 test error and standard deviation (right) and mean L_∞ test error and standard deviation (left) for the RMFNN and DNLNN sequential methods (cyan) and concatenation methods (magenta) trained on the concatenation favoring MF training set. The same statistics for a NN trained on only HF data is also displayed (green).*

Examining Figure 3.4 we see that the results are very similar across both the L_∞ and L_2 performance measures. For the RMFNN methods we see that the sequential extension far outperforms the the concatenation extension. This is inline with our prediction based on the structure of the MF training set.

For the DNLNN methods the difference in performance between the sequential and concatenation extensions is negligible.

Comparing the results across all MF methods we see that RMFNN sequential yields both the lowest mean L_2 test error and lowest mean L_∞ test error. At the same time RMFNN concatenation is the worst for both measures. Also note that all MF methods outperform NN_{HF} in terms of both mean test error and predictive uncertainty. Additional numerical experimentation and a more robust UQ strategy is needed before we can say anything conclusive about the predictive uncertainty of these methods.

Examining Figure 3.5 we again see that the results are very similar across both the L_∞ and L_2 performance measures. For the RMFNN results we see that the sequential method far outperforms the concatenation method. This is opposite of what was predicted, and it suggests a potential pitfall in RMFNN concatenation. As the method currently functions, it attempts to independently learn the correlations between (1) the LF_1 and HF data and (2) the LF_2 and HF data. The training sets for these correlational maps match the number of HF data available. In the case where the HF data is especially sparse or does not spread evenly over the full domain these correlational maps can be extremely difficult to learn. In certain cases this data sparsity issue may negate any inherent benefit the concatenation method has stemming from the overall structure of the MF training set. Further experimentation is necessary before any conclusion can be drawn about this hypothesis.

For the DNLNN method we see the results we expected. The concatenation extension outperforms the sequential by a small margin.

It is also illuminating to compare DNLNN concatenation with the RMFNN concatenation. Recalling Figure 2.5 the DNLNN concatenation method learns a single general correlation between (1) LF_1 and LF_2 data, and (2) HF data. We hypothesize that if one of the correlational maps ($LF_1 \mapsto HF$ or $LF_2 \mapsto HF$) is considerably more difficult than the other the DNLNN concatenation method will choose to ignore the lower fidelity data associated with the harder correlational map. This could be useful in a situation where some portion of the lower fidelity data is too poor to be informative of the true quantity of interest, but in other situations it may prevent the method from optimally harnessing all lower fidelity information. Compare this with RMFNN concatenation, Figure 2.6, where each of the correlations ($LF_1 \mapsto (HF - LF_1)$ and $LF_2 \mapsto (HF - LF_2)$) are learned independently, and where the synthetic data generation workflow enforces that each of these learned maps is used to generate training data for NN_{HF}, which issues final network predictions. This strategy makes sure that all lower fidelity data is harnessed, but, in the case that one of the correlational maps ($LF_1 \mapsto HF - HF - LF_1$ or $LF_2 \mapsto HF - LF2$) is considerably harder than the other, we hypothesize that the overall generalization error of the method will trend with the accuracy with which the more difficult correlational map is learned. More experimentation is necessary to reject or support these hypotheses.

Comparing the results across all the methods we again see that RMFNN sequential is best in all collected performance measures while RMFNN concatenation is worst in all collected measures. We also see that all the MF methods outperform NN_{HF} in terms of both test error and predictive uncertainty. Additional experimentation and more robust dropout procedures are needed before we can say anything conclusive concerning the predictive uncertainty of the methods.

4. Conclusions and Future Work. In this work we considered two distinct strategies for leveraging MF training data in feed-forward supervised learning networks. These were the DNLNN and RMFNN methods which have been independently studied for the case of two fidelity classes in [17] and [3]. In section 2.1, we pointed out that MF data with greater than two fidelity classes can be both hierarchical and non-hierarchical, and argued that it is important to consider this structure when trying to leverage MF data in network training. For hierarchical data we argued that learning the correlations between fidelity classes in a sequential manner is preferable, and for non-hierarchical data we argued that the correlations should be learned via the concatenation method. Moreover, in section 2.2, we provided two distinct strategies for extending the DNLNN and RMFNN methods for any number of fidelity classes based on the sequential and concatenation correlational learning frameworks.

In section 3.2, we saw that for a function approximation task the performance of the two fidelity composite RMFNN and two fidelity DNLNN methods in terms of mean test error and predictive uncertainty was dependent on the nature of the correlation between the LF and HF data. When the LF data was linearly correlated with the HF data we saw both MF methods far outperform a single fidelity neural network, NN_{HF}, trained on exclusively HF data with cost equivalent to MF training sets used for the MF methods. In this case where we had linear correlation between the fidelity classes the RMFNN composite method was particularly successful. It far outperformed the DNLNN method on the basis of mean test error. When the LF data was non-linearly correlated with the HF data we saw little benefit to using the MF methods. Both DNLNN and RMFNN require that the correlation between the LF and HF data be learned on a training set with size equal to the number of HF training samples. When the correlation between the data fidelity classes is very non-linear and HF training data is sparse these methods can falter.

In section 3.3, we found inconclusive results concerning whether or not the concatenation and sequential MF methods perform differently depending on the structure of the MF data. We considered two separate MF data sets. One of them was theoretically structured to favor the sequential methods and the other to favor the concatenation methods. In terms of mean test error across all MF methods and both MF data sets RMFNN sequential performed best and RMFNN concatenation performed worst. The results for DNLNN were more in line with our hypothesis. For the MF data set favoring the sequential extension both DNLNN methods performed similarly in terms of mean test error, and for the MF data set favoring the concatenation extensions the DNLNN concatenation method outperformed the DNLNN sequential method in terms of mean test error. When compared to a single fidelity neural network, NN_{HF}, trained on HF data of cost equivalent to the MF training data, all MF methods were superior in terms of both mean test error and predictive uncertainty.

Future work will consider adding an active learning component to our proposed RMFNN and DNLNN methods where initial predictions inform future sampling in targeted areas of the domain where the method suffers from epistemic error or high predictive uncertainty.

REFERENCES

[1] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al., *A review of uncertainty quantification in deep learning: Techniques, applications and challenges*, Information Fusion, 76 (2021), pp. 243–297.

[2] P. Baldi and P. J. Sadowski, *Understanding dropout*, Advances in neural information processing systems, 26 (2013).

[3] O. Davis, M. Motamed, and R. Tempone, *Residual multi-fidelity neural network computing.* Preprint, 2022.

[4] M. G. Fernández-Godino, C. Park, N.-H. Kim, and R. T. Haftka, *Review of multi-fidelity models*, arXiv preprint arXiv:1609.07196, (2016).

[5] A. I. Forrester, A. Sóbester, and A. J. Keane, *Multi-fidelity optimization via surrogate modelling*, Proceedings of the royal society a: mathematical, physical and engineering sciences, 463 (2007),

pp. 3251–3269.

[6] Y. Gal and Z. Ghahramani, *Dropout as a bayesian approximation: Representing model uncertainty in deep learning*, in international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[8] M. Guo, *A brief note on understanding neural networks as gaussian processes*, arXiv preprint arXiv:2107.11892, (2021).

[9] M. Guo, A. Manzoni, M. Amendt, P. Conti, and J. S. Hesthaven, *Multi-fidelity regression using artificial neural networks: efficient approximation of parameter-dependent output quantities*, Computer methods in applied mechanics and engineering, 389 (2022), p. 114378.

[10] E. Hüllermeier and W. Waegeman, *Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods*, Machine Learning, 110 (2021), pp. 457–506.

[11] A. Kendall and Y. Gal, *What uncertainties do we need in bayesian deep learning for computer vision?*, Advances in neural information processing systems, 30 (2017).

[12] M. C. Kennedy and A. O'Hagan, *Predicting the output from a complex computer code when fast approximations are available*, Biometrika, 87 (2000), pp. 1–13.

[13] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, *Deep neural networks as gaussian processes*, arXiv preprint arXiv:1711.00165, (2017).

[14] Z. Li, S. Zhang, H. Li, K. Tian, Z. Cheng, Y. Chen, and B. Wang, *On-line transfer learning for multi-fidelity data fusion with ensemble of deep neural networks*, Advanced Engineering Informatics, 53 (2022), p. 101689.

[15] X. Meng, H. Babaee, and G. E. Karniadakis, *Multi-fidelity bayesian neural networks: Algorithms and applications*, Journal of Computational Physics, 438 (2021), p. 110361.

[16] X. Meng and G. E. Karniadakis, *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems*, Journal of Computational Physics, 401 (2020), p. 109020.

[17] M. Motamed, *A multi-fidelity neural network surrogate sampling method for uncertainty quantification*, International Journal for Uncertainty Quantification, 10 (2020).

[18] R. M. Neal, *Bayesian learning for neural networks*, vol. 118, Springer Science & Business Media, 2012.

[19] J. Nitzler, J. Biehler, N. Fehn, P.-S. Koutsourelakis, and W. A. Wall, *A generalized probabilistic learning approach for multi-fidelity uncertainty quantification in complex physical simulations*, Computer Methods in Applied Mechanics and Engineering, 400 (2022), p. 115600.

[20] J. S. Park, *Tuning complex computer codes to data and optimal designs*, University of Illinois at Urbana-Champaign, 1991.

[21] L. Partin, G. Geraci, A. A. Rushdi, M. S. Eldred, and D. E. Schiavazzi, *Multifidelity data fusion in convolutional encoder/decoder networks*, Journal of Computational Physics, (2022), p. 111666.

[22] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis, *Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 473 (2017), p. 20160751.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The journal of machine learning research, 15 (2014), pp. 1929–1958.

[24] S. Xiong, P. Z. Qian, and C. J. Wu, *Sequential design and analysis of high-accuracy and low-accuracy computer codes*, Technometrics, 55 (2013), pp. 37–46.

Appendix A. Pseudocode.

A.1. DNLNN Pseudocode Implementation.

```
1:  procedure Training Data Generation
2:      $N = N_{LF_{M-1}} + N_{LF_{M-2}} + \cdots + N_{LF_1} + N_{HF}$ = number of training samples
3:      procedure Generate Input Values
4:          $X_{HF} := \{x^{(1)}, \cdots, x^{(N_{HF})}\}$
5:          $X_{LF_1} := \{x^{(N_{HF}+1)}, \cdots, x^{(N_{HF}+N_{LF_1})}\}$
6:          $X_{LF_2} := \{x^{(N_{HF}+N_{LF_1}+1)}, \cdots, x^{(N_{HF}+N_{LF_1}+N_{LF_2}+1)}\}$
7:          $\vdots$
8:          $X_{LF_{M-1}} := \{x^{(N-N_{LF_{M-1}})}, \cdots, x^{(N)}\}$
9:          $X = X_{LF_1} \cup X_{LF_2} \cup \cdots \cup X_{LF_{M-1}} \cup X_{HF}$
10:     end procedure
11:     procedure Generate Output Values
```

```
12:         for $x^{(i)} \in X_{HF}$ do
13:             compute: $y^{(i)}_{HF}$
14:         end for
15:         for $j = 1 \cdots, M-1$ do
16:             for $x^{(i)} \in X_{LF_j}$ do
17:                 compute: $y^{(i)}_{LF_j}$
18:             end for
19:         end for
20:     end procedure
21: end procedure
22: if Sequential then
23:     procedure Network Training
24:         for $x^{(i)} \in X_{LF_{M-1}}$ do
25:             $NN_{LF_{M-1}}$ learns $x^{(i)} \mapsto y^{(i)}_{LF_{M-1}}$
26:         end for
27:         for $j = M-1, \cdots, 1$ do
28:             for $x^{(i)} \in X_{LF_{j-1}}$ do
29:                 Define: $X_{LF_0} := X_{HF}$
30:                 $NN_{corr_j}$ learns:
31:                 $(x^{(i)}, NN_{corr_{j+1}}(\cdots NN_{corr_{M-1}}(x^{(i)}, NN_{LF_{M-1}}(x)))) \mapsto y^{(i)}_{LF_{j-1}}$
32:             end for
33:         end for
34:     end procedure
35:     procedure Network Prediction
36:         Test Input: $x$
37:         $y = NN_{corr_1}(x, NN_{corr_2}(x, NN_{corr_3}(\cdots NN_{corr_{M-1}}(x, NN_{LF_{M-1}}(x)))))$
38:     end procedure
39: end if
40: if Concatenation then
41:     procedure Network Training
42:         for $j = 1, \cdots, M-1$ do
43:             for $x^{(i)} \in X_{LF_j}$ do
44:                 $NN_{LF_j}$ learns $x^{(i)} \mapsto y^{(i)}_{LF_j}$
45:             end for
46:         end for
47:         for $x^{(i)} \in X_{HF}$ do
48:             $NN_{corr}$ learns $(x^{(i)}, NN_{LF_1}(x^{(i)}), \cdots, NN_{LF_{M-1}}(x^{(i)})) \mapsto y^{(i)}_{HF}$
49:         end for
50:     end procedure
51:     procedure Network Prediction
52:         Test Input: $x$
53:         $y = NN_{corr}(x, NN_{LF_1}(x), NN_{LF_2}(x), \cdots, NN_{LF_{M-1}}(x))$
54:     end procedure
55: end if
```

A.2. RMFNN Pseudocode Implementation.

```
1: procedure Training Data Generation
2:     $N = N_{LF_{M-1}} + N_{LF_{M-2}} + \cdots + N_{LF_1} + N_{HF}$ = number of training samples
```

```
    procedure Generate Input Values
        $X_{HF} := \{x^{(1)}, \cdots, x^{(N_{HF})}\}$
        $X_{LF_1} := \{x^{(N_{HF}+1)}, \cdots, x^{(N_{HF}+N_{LF_1})}\}$
        $X_{LF_2} := \{x^{(N_{HF}+N_{LF_1}+1)}, \cdots, x^{(N_{HF}+N_{LF_1}+N_{LF_2}+1)}\}$
        $\vdots$
        $X_{LF_{M-1}} := \{x^{(N-N_{LF_{M-1}})}, \cdots, x^{(N)}\}$
        $X = X_{LF_1} \cup X_{LF_2} \cup \cdots \cup X_{LF_{M-1}} \cup X_{HF}$
    end procedure
    procedure Generate Output Values
        for $x^{(i)} \in X_{HF}$ do
            compute: $y_{HF}^{(i)}$
        end for
        for $j = 1 \cdots, M-1$ do
            for $x^{(i)} \in X_{LF_j}$ do
                compute: $y_{LF_j}^{(i)}$
            end for
        end for
    end procedure
end procedure
if Sequential then
    procedure Network Training
        if Sequential then
            Define $LF_0 := HF$
            for $j = 1, \cdots, M-1$ do
                for $x^{(i)} \in X_{LF_j}$ do
                    $NN_{LF_j}$ learns $x^{(i)} \mapsto y_{LF_j}^{(i)}$
                end for
            end for
            for $j = M-1, \cdots, 1$ do
                for $x^{(i)} \in X_{LF_{j-1}}$ do
                    $NN_{res_j}$ learns $(x^{(i)}, NN_{LF_j}(x^{(i)})) \mapsto (y_{LF_{j-1}}^{(i)} - NN_{LF_j}(x^{(i)}))$
                end for
                Generate synthetic $LF_{j-1}$ outputs
                for $x^{(i)} \in X_{LF_j}$ do
                    $y_{LF_{j-1}}^{(i)} = NN_{res_j}(x^{(i)}, NN_{LF_j}(x^{(i)})) + NN_{LF_j}(x^{(i)})$
                    $X_{LF_{j-1}} \leftarrow X_{LF_{j-1}} \cup X_{LF_j}$
                end for
            end for
            for $x^{(i)} \in X_{HF}$ do
                $NN_{HF}$ learns $x^{(i)} \mapsto y_{HF}^{(i)}$
            end for
        end if
    end procedure
end if
if Concatenation then
    procedure Network Training
        for $j = 1, \cdots, M-1$ do
            for $x^{(i)} \in X_{LF_j}$ do
```

```
51:                 $NN_{LF_j}$ learns $x^{(i)} \mapsto y^{(i)}_{LF_j}$
52:             end for
53:             for $x^{(i)} \in X_{HF}$ do
54:                 $NN_{res_j}$ learns $(x^{(i)}, NN_{LF_j}(x^{(i)})) \mapsto (y^{(i)}_{HF} - NN_{LF_j}(x^{(i)}))$
55:             end for
56:         end for
57:         Generate synthetic $HF$ outputs
58:         for $j = 1, \cdots, M-1$ do
59:             for $x^{(i)} \in X_{LF_j}$ do
60:                 $y^{(i)}_{HF} = NN_{res_j}(x^{(i)}, NN_{LF_j}(x^{(i)})) + NN_{LF_j}(x^{(i)})$
61:                 $X_{HF} \leftarrow X_{HF} \cup X_{LF_j}$
62:             end for
63:         end for
64:         for $x^{(i)} \in X_{HF}$ do
65:             $NN_{HF}$ learns $x^{(i)} \mapsto y^{(i)}_{HF}$
66:         end for
67:     end procedure
68: end if
69: procedure Network Prediction
70:     Test Input: $x$
71:     Prediction: $y = NN_{HF}(x)$
72: end procedure
```

A.3. RMFNN 2-Fidelity Composite Approach Pseudocode Implementation.

```
1: procedure Training Data Generation
2:     $N = N_{LF} + N_{HF}$ = number of training samples
3:     procedure Generate Input Values
4:         $X_{LF} := \{x^{(1)}, \cdots, x^{(N_{LF})}\}$, $X_{HF} := \{x^{(N_{LF}+1)}, \cdots, x^{(N)}\}$
5:         Require: $X_{LF} \cap X_{HF} = \emptyset$
6:         $X = X_{LF} \cup X_{HF}$
7:     end procedure
8:     procedure Generate Output Values
9:         for $x^{(i)} \in X_{LF}$ do
10:            compute $y^{(i)}_{LF}$
11:        end for
12:        for $x^{(i)} \in X_{HF}$ do
13:            compute $y^{(i)}_{HF}$
14:        end for
15:    end procedure
16: end procedure
17: procedure Network Training
18:     for $x^{(i)} \in X_{LF}$ do
19:         $NN_{LF}$ learns $x^{(i)} \mapsto y^{(i)}_{LF}$
20:     end for
21:     for $x^{(i)} \in X_{HF}$ do
22:         $NN_{res}$ learns $(x^{(i)}, NN_{LF}(x^{(i)})) \mapsto (y^{(i)}_{HF} - NN_{LF}(x^{(i)})$
23:     end for
24: end procedure
25: procedure Network Prediction
```

```
26:     Test Input: $x$
27:     Prediction: $y = NN_{res}(x, NN_{LF}(x)) + NN_{LF}(x)$
28: **end procedure**
```

A SYNCHRONOUS PARTITIONED SCHEME FOR COUPLED REDUCED ORDER MODELS BASED ON SEPARATE REDUCED ORDER BASES FOR THE INTERIOR AND INTERFACE VARIABLES.

AMY DE CASTRO*, PAUL KUBERRY†, IRINA TEZAUR‡, AND PAVEL BOCHEV§

Abstract.
We present further development of the schemes proposed in [3], which couple a projection-based reduced order model (ROM) to either a finite element model (FEM) or another ROM. The non-singularity of the dual Schur complement is an essential requirement for these coupling schemes, because it defines a system for the interface flux that is used to decouple the subdomain equations. The approach in [3] utilized full subdomain bases containing all subdomain degrees of freedom (DOFs) and did not provide rigorous guarantees for unique solvability of the ROM-ROM/FEM systems. To address this issue we reformulate our coupling scheme by (i) employing separate reduced bases (RBs) for the interior and interface DOFs, and (ii) using one of the interface RBs for the Lagrange multipliers. These modifications ensure that the dual Schur complement is provably non-singular.

1. Introduction. Partitioned methods are an attractive alternative to monolithic approaches for multiphysics applications, because they increase concurrency and enable reuse of existing codes for the constituent physics components; see, e.g., [7] for an expository survey. Furthermore, because each individual component is solved independently, the codes can run at their "sweet spots" utilizing, e.g., multi-rate time integrators [8]. Performance of partitioned schemes can be further enhanced by replacing the full-fidelity models in one or more subdomains by computationally efficient projection-based reduced order models (ROMs).

This work continues our efforts in [3] to extend the partitioned schemes in [17] and [21] to the coupling of projection-based ROM to either a finite element model (FEM) or another ROM. In [3], we defined the subdomain ROMs by utilizing full subdomain bases obtained by performing proper orthogonal decomposition (POD) [19, 12] on a collection of snapshots representing both the interior and interface DOFs. While this strategy is common in approaches combining domain decomposition and ROM (see, e.g., [15]), it does not guarantee that the dual Schur complement system for the Lagrange multiplier is non-singular. Unique solvability of this system is essential for the extension of the partitioned schemes in [17] and [21] because the Lagrange multiplier defines the interface flux, which allows us to define well-posed subdomain equations that can be solved independently.

In this paper, we propose modifications of the coupling scheme in [3] that address this issue. First, instead of using the conventional full subdomain bases, we construct separate ROM bases for the interfacial and interior DOFs. Second, the dimension of the Lagrange multiplier space is also reduced by utilizing the reduced interfacial basis from a particular subdomain. By combining these two ideas, we are able to obtain a provably non-singular, symmetric and positive definite Schur complement matrix. We provide numerical results that demonstrate the ability of these modifications to improve the conditioning of the dual Schur complement system and to obtain accurate solutions while retaining fewer reduced basis modes than the original formulation. Results are shown on a two-dimensional (2D) time-dependent advection-diffusion problem in the advection-dominated (high Péclet) regime.

Conceptually, our work is related to approaches combining reduced order models and

*Clemson University, agmurda@clemson.edu
†Sandia National Laboratories, pakuber@sandia.gov
‡Sandia National Laboratories, ikalash@sandia.gov
§Sandia National Laboratories, pbboche@sandia.gov

domain decomposition (DD) ideas such as the Reduced Basis Element (RBE) method [15, 16], the Reduced Basis DD method [13], and the static condensation RBE [18], among others. However, these papers focus primarily on using DD as a vehicle to improve the efficiency of model order reduction for extreme scale problems and decomposable problems. In the former case the offline phase can be computationally intractable if a global state space is used, whereas in the latter case the problem domain can be decomposed into a relatively small number of *archetypes* [18], each depending on a separate set of parameters. A change in an archetype thus changes the global space, making a monolithic problem ill-suited for these types of problems. Typically, application of DD in the above context leads to subdomains separated by artificial rather than physical interfaces. In contrast, we consider multiphysics problems where the interface is physical and our main goal is to enable partitioned solution of the coupled system. Thus, our focus is on schemes enabling the independent solution of the subdomain equations using possibly different time integrators running at different time steps (see, e.g., [8, 9, 2]), whereas the combinations of ROM and DD cited earlier use the same time integrator and time step for all subdomains.

2. Model problem and coupling formulation. In this section we define the model transmission problem and summarize the Implicit Value Recovery (IVR) coupling [17] that forms the basis for our ROM-ROM and ROM-FEM coupling schemes. Let $\Omega \in \mathbb{R}^d$, $d = 2, 3$ denote a bounded region divided into two non-overlapping subdomains Ω_1 and Ω_2 by an interface γ. Without a loss of generality, we assume that the unit normal $\boldsymbol{n}_\gamma$ points towards Ω_2, and set $\Gamma_i := \partial\Omega_i \backslash \gamma$, $i = 1, 2$. We then consider the following advection-diffusion transmission problem

$$\begin{aligned} \dot{\varphi}_i - \nabla \cdot F_i(\varphi_i) = f_i \qquad & \text{on } \Omega_i \times [0, T] \\ \varphi_i = g_i \qquad & \text{on } \Gamma_i \times [0, T], \quad i = 1, 2, \end{aligned} \tag{2.1}$$

where the over-dot notation denotes differentiation in time, the unknown φ_i is a scalar field, $F_i(\varphi_i) = \kappa_i \nabla \varphi_i - \mathbf{u}\varphi_i$ is the total flux function, $\kappa_i > 0$ is the diffusion coefficient in Ω_i, and $\mathbf{u}$ the velocity field. We augment (2.1) with initial conditions:

$$\varphi_i(\mathbf{x}, 0) = \varphi_{i,0}(\mathbf{x}) \qquad \text{in } \Omega_i, \quad i = 1, 2\,. \tag{2.2}$$

Along the interface γ, we enforce continuity of the states and continuity of the total flux, giving rise to the following interface conditions:

$$\varphi_1(\mathbf{x}, t) - \varphi_2(\mathbf{x}, t) = 0 \quad \text{and } F_1(\mathbf{x}, t) \cdot \mathbf{n}_\gamma = F_2(\mathbf{x}, t) \cdot \mathbf{n}_\gamma \quad \text{on } \gamma \times [0, T]. \tag{2.3}$$

To define the IVR scheme we differentiate the first interface condition in (2.3) in time[1] and use Lagrange multipliers to enforce it in a weak sense. We then discretize the weak monolithic problem over a finite element subspace $V^h \subset V := H^1_\Gamma(\Omega_1) \times H^1_\Gamma(\Omega_2) \times H^{-1/2}(\gamma)$. The resulting semi-discrete monolithic problem can be written in the following compact matrix form:

$$\begin{bmatrix} M_1 & 0 & G_1^T \\ 0 & M_2 & -G_2^T \\ G_1 & -G_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{\Phi}}_1 \\ \dot{\mathbf{\Phi}}_2 \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 - K_1 \mathbf{\Phi}_1 \\ \boldsymbol{f}_2 - K_2 \mathbf{\Phi}_2 \\ \mathbf{0} \end{bmatrix}, \tag{2.4}$$

where, for $i = 1, 2$, M_i is a mass matrix, K_i is a stiffness matrix, $\boldsymbol{f}_i$ is a source term vector, and $\mathbf{\Phi}_i$ is the vector containing the unknown nodal values of the solution. The matrices G_i

[1]As long as the initial conditions are continuous over the interface, this differentiated constraint is equivalent to the original one.

define the algebraic form of the first constraint in (2.3) and $\boldsymbol{\lambda}$ is the coefficient vector of the discrete Lagrange multiplier.

One can show that if the Lagrange multiplier space is taken to be the trace of the finite element space on either of Ω_1 or Ω_2, the dual Schur complement of the matrix in (2.4) is symmetric and positive definite; see [17] for details. This fact is at the core of the IVR scheme because it allows us to solve for the Lagrange multiplier in terms of the subdomain states, which reduces (2.4) to a system of two ODEs:

$$\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\Phi}}_1 \\ \dot{\boldsymbol{\Phi}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_1 - K_1 \boldsymbol{\Phi}_1 - G_1^T \boldsymbol{\lambda}(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2) \\ \boldsymbol{f}_2 - K_2 \boldsymbol{\Phi}_2 + G_2^T \boldsymbol{\lambda}(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2) \end{bmatrix}. \tag{2.5}$$

It is easy to see that application of either a *fully explicit* or an *implicit-explicit* (IMEX) scheme that treats $\boldsymbol{\lambda}(\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2)$ explicitly, to each equation in (2.5) decouples the subdomain equations and allows their independent solution; see [17]. We note that these time integrators are not required to be the same and they can also use different time steps over shared synchronization time intervals.

3. Extension of IVR to ROM-ROM/FEM couplings using full subdomain bases. In this section we briefly review the extension of IVR to ROM-ROM and ROM-FEM couplings, developed in [3]. For brevity we only present the ROM-ROM case, as the ROM-FEM coupling is very similar.

To obtain the coupled ROM-ROM problem, we apply the POD/Galerkin method to the coupled FEM-FEM problem (2.4). During the offline stage of the model reduction procedure, we solve a monolithic FEM problem to collect a series of solution snapshots over the entire domain Ω and arrange them into a snapshot matrix $\overline{X}$. We then partition $\overline{X}$ into subdomain snapshot matrices $\overline{X}_j$ containing the finite element DOFs on Ω_j, $j = 1, 2$. After zeroing all entries of $\overline{X}_j$ corresponding to the Dirichlet nodes in the finite element mesh we obtain the adjusted snapshot matrices X_j and compute their SVD, i.e., $X_j = U_j \Sigma_j V_j^T$. The reduced basis on each subdomain is defined as the first d left singular vectors of the adjusted snapshot matrix, where $d > 0$ is an integer that depends on the desired ROM accuracy. We arrange these vectors in reduced basis matrices $\widetilde{U}_j$, $j = 1, 2$. Because the columns of these matrices contain both the interior and interface DOFs on Ω_j, in the literature they are usually referred to as the *full subdomain bases*. To effect the projection of the coupled FEM-FEM problem (2.4) onto the reduced basis we set

$$\boldsymbol{\Phi}_i := \widetilde{U}_i \boldsymbol{\varphi}_i + \boldsymbol{\beta}_i; \quad i = 1, 2. \tag{3.1}$$

The rows of $\boldsymbol{\beta}_i$ corresponding to non-Dirichlet nodes are set to 0, while the rows corresponding to Dirichlet nodes on Γ_i contain the values of the Dirichlet boundary data g_i at these nodes; see [10]. Note that the FEM coefficient vectors $\boldsymbol{\Phi}_i$, the reduced order coefficient vectors $\boldsymbol{\varphi}_i$, and the boundary vectors $\boldsymbol{\beta}_i$ are functions of the time.

Inserting (3.1) into (2.4) and then multiplying the resulting subdomain equations on the left by $\widetilde{U}_j^T$ yields the full subdomain basis monolithic ROM-ROM system:

$$\begin{bmatrix} \widetilde{M}_1 & 0 & \widetilde{G}_1^T \\ 0 & \widetilde{M}_2 & -\widetilde{G}_2^T \\ \widetilde{G}_1 & -\widetilde{G}_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\varphi}}_1 \\ \dot{\boldsymbol{\varphi}}_2 \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{f}}_1 - \widetilde{K}_1 \boldsymbol{\varphi}_1 - \widetilde{U}_1^T K_1 \boldsymbol{\beta}_1 - \widetilde{U}_1^T M_1 \dot{\boldsymbol{\beta}}_1 \\ \widetilde{\boldsymbol{f}}_2 - \widetilde{K}_2 \boldsymbol{\varphi}_2 - \widetilde{U}_2^T K_2 \boldsymbol{\beta}_2 - \widetilde{U}_2^T M_2 \dot{\boldsymbol{\beta}}_2 \\ -G_1 \dot{\boldsymbol{\beta}}_1 + G_2 \dot{\boldsymbol{\beta}}_2 \end{bmatrix} =: \begin{bmatrix} \boldsymbol{s}_1 \\ \boldsymbol{s}_2 \\ 0 \end{bmatrix} \tag{3.2}$$

where $\widetilde{M}_i = \widetilde{U}_i^T M_i \widetilde{U}_i$, $\widetilde{K}_i = \widetilde{U}_i^T K_i \widetilde{U}_i$, $\widetilde{G}_i = G_i \widetilde{U}_i$, and $\widetilde{\boldsymbol{f}}_i = \widetilde{U}_i^T \boldsymbol{f}_i$. We note that the right-hand side of the last equation, $-G_1 \dot{\boldsymbol{\beta}}_1 + G_2 \dot{\boldsymbol{\beta}}_2$, is precisely zero, because the matrices G_i act only on the interface nodes where the coefficients of $\boldsymbol{\beta}_i$ are zero by construction.

Similar to the monolithic FEM-FEM problem (2.4), which is the basis for the IVR scheme, the monolithic problem (3.2) is the basis for the extension of IVR to ROM-ROM couplings, which reads as follows.

Offline: Computation of POD bases and and pre-computation of matrices for the full subdomain basis ROMs

1. Using a suitable monolithic FOM solve the model problem (2.1) on Ω, collect solution snapshots into a snapshot matrix $\overline{X}$ and partition $\overline{X}$ into subdomain snapshot matrices $\overline{X}_j$, $j = 1, 2$. Compute the SVD of the adjusted snapshot matrix $X_j = U_j \Sigma_j V_j^T$.
2. For $j = 1, 2$, given a threshold $\delta_j > 0$, define the reduced basis dimension $N_{R,j}$ as the smallest integers such that $\sum_{k=1}^{N_{R,j}} \sigma_{j,k}^2 \geq \Big(1 - \delta_j\Big) \sum_{k=1}^{N} \sigma_{j,k}^2$, where $\sigma_{j,k}$ are the singular values from the diagonal of Σ_j, and N is the total number of singular values. Define the reduced basis matrices $\widetilde{U}_j$ by retaining the first $N_{R,j}$ columns of U_j.
3. For $j = 1, 2$, pre-compute the ROM matrices, where N_γ is the size of the LM space:

$$\widetilde{M_j} := \widetilde{U}_j^T M_j \widetilde{U}_j \in \mathbb{R}^{N_{R,j} \times N_{R,j}}; \quad \widetilde{K}_j := \widetilde{U}_j^T K_j \widetilde{U}_j \in \mathbb{R}^{N_{R,j} \times N_{R,j}}; \quad \text{and} \quad \widetilde{G}_j := G_j \widetilde{U}_j \in \mathbb{R}^{N_\gamma \times N_{R,j}}. \tag{3.3}$$

Online: Solution of the full subdomain basis ROM-ROM system

1. Given a simulation time interval $[0, T]$ choose an explicit time integration scheme, or an IMEX scheme with explicit treatment of $\boldsymbol{\lambda}$, for each subdomain, i.e., an operator $D_{j,t}^n(\boldsymbol{\varphi})$, $j = 1, 2$.
2. For $n = 0, 1, \ldots$ use $\boldsymbol{\varphi}_1^n$ to compute the vector

$$\boldsymbol{s}_1^n := \widetilde{U}_1^T \mathbf{f}_1^n - \widetilde{K}_1 \boldsymbol{\varphi}_1^n - \widetilde{U}_1^T K_1 \boldsymbol{\beta}_1^n - \widetilde{U}_1^T M_1 \dot{\boldsymbol{\beta}}_1^n$$

3. For $n = 0, 1, \ldots$ use $\boldsymbol{\varphi}_2^n$ to compute the vector

$$\boldsymbol{s}_2^n := \widetilde{U}_2^T \mathbf{f}_2^n - \widetilde{K}_2 \boldsymbol{\varphi}_2^n - \widetilde{U}_2^T K_2 \boldsymbol{\beta}_2^n - \widetilde{U}_2^T M_2 \dot{\boldsymbol{\beta}}_2^n.$$

4. Solve the Schur complement system

$$\big(\widetilde{G}_1 \widetilde{M_1}^{-1} \widetilde{G}_1^T + \widetilde{G}_2 \widetilde{M_2}^{-1} \widetilde{G}_2^T\big) \boldsymbol{\lambda}^n = \widetilde{G}_1 \widetilde{M_1}^{-1} \boldsymbol{s}_1^n - \widetilde{G}_2 \widetilde{M_2}^{-1} \boldsymbol{s}_2^n$$

   for $\boldsymbol{\lambda}^n$. Compute $\widetilde{G}_1^T \boldsymbol{\lambda}^n$ and $\widetilde{G}_2^T \boldsymbol{\lambda}^n$.
5. Solve the system $\widetilde{M_1} D_{1,t}^n(\boldsymbol{\varphi}_1) = \boldsymbol{s}_1^n - \widetilde{G}_1^T \boldsymbol{\lambda}^n$.
6. Solve the system $\widetilde{M_2} D_{2,t}^n(\boldsymbol{\varphi}_2) = \boldsymbol{s}_2^n + \widetilde{G}_2^T \boldsymbol{\lambda}^n$.
7. Project the ROM solutions $\boldsymbol{\varphi}_i$ onto the full order state spaces on Ω_i, $i = 1, 2$:

$$\boldsymbol{\Phi}_1 := \widetilde{U}_1 \boldsymbol{\varphi}_1 + \boldsymbol{\beta}_1; \quad \boldsymbol{\Phi}_2 := \widetilde{U}_2 \boldsymbol{\varphi}_2 + \boldsymbol{\beta}_2.$$

It is clear that success of the above extension of IVR to a ROM-ROM coupling hinges on the unique solvability of the Schur complement system in Step 4 of the *online* portion of the scheme. For the original IVR scheme, one can show that under some conditions on the Lagrange multiplier space the dual Schur complement of the upper left 2×2 submatrix from (2.4) is symmetric and positive definite; see [17] for details. In particular, choosing the trace of the finite element space from either one of the subdomains as a Lagrange multiplier space always satisfies these conditions.

At the same time, one can easily construct an example for which the matrix $\widetilde{S} := \widetilde{G}_1\widetilde{M}_1^{-1}\widetilde{G}_1^T + \widetilde{G}_2\widetilde{M}_2^{-1}\widetilde{G}_2^T$ cannot even be defined. To that end, it suffices to consider any manufactured solution whose restriction to Ω_i is supported on a compact subset $\omega_i \subset \Omega_i$ such that ω_i does not intersect any elements having nodes on the interface γ. In this case, all the rows of the snapshot matrix corresponding to interfacial degrees of freedom would be exactly 0, which would result in $\widetilde{U}_i$ being 0 on those same rows. It is easy to see that in this case the projected mass matrices $\widetilde{M}_i$ will be singular, and so the Schur complement will be undefined.

The example problem used to perform reproductive tests of the full subdomain basis extensions of IVR to ROM-ROM and ROM-FEM in [3] did not have the pathology described in the above scenario and we did not observe failures of these formulations. At the same time, these extensions of IVR obviously cannot offer any guarantees that the ROM-ROM and ROM-FEM couplings will be well-posed for all possible configurations of the model transmission problem (2.1). In the next section we formulate a modified version of the IVR extension that resolves this issue.

4. A split reduced basis ROM-ROM formulation. In this section we formulate an alternative extension of the IVR scheme to a ROM-ROM coupling that has a provably non-singular dual Schur complement. To achieve this property we need to ensure the following two conditions: first, the projected mass matrices $\widetilde{M}_i$, $i = 1, 2$ are symmetric and positive definite, and second, the constraint matrix $(\widetilde{G}_1, \widetilde{G}_2)^T$ has a full column rank. To that end we consider a combination of two ideas targeting each one of these conditions.

4.1. Projection-based ROM using split reduced bases. To prevent the projected mass matrices $\widetilde{M}_i$ from becoming singular we must ensure that the sub-columns of $\widetilde{U}_i$ corresponding to the interfacial DOFs have full column rank on their own. It is clear that such a property cannot be achieved when using the full subdomain basis, in which case, one can only guarantee that the *entire* columns of $\widetilde{U}_i$, comprising both interior and interface DOFs, are full column rank.[2].

To regain control over the properties of the sub-columns containing interfacial DOFs we propose to construct separate reduced order bases $\widetilde{U}_{i,\gamma}$ and $\widetilde{U}_{i,0}$, performing POD for the interface and interior DOFs, respectively. The construction of these matrices is described in Steps 1-2 of the offline stage of the algorithm presented in Section 4.3, and is similar in flavor to the approach of Eftang *et al.* [4] and Hoang *et al.* [11]. To effect the projection of the FEM-FEM monolithic problem, instead of (3.1) we now consider two separate expansions for the interior and interface DOFs given by

$$\boldsymbol{\Phi}_{i,0} = \widetilde{U}_{i,0}\boldsymbol{\varphi}_{i,0} + \boldsymbol{\beta}_{i,0} \quad \text{and} \quad \boldsymbol{\Phi}_{i,\gamma} = \widetilde{U}_{i,\gamma}\boldsymbol{\varphi}_{i,\gamma} + \boldsymbol{\beta}_{i,\gamma} \tag{4.1}$$

respectively. We then insert (4.1) into (2.4) and multiply the blocks corresponding to the interior and interface DOFs by $\widetilde{U}_{i,0}^T$ and $\widetilde{U}_{i,\gamma}^T$, respectively. These steps yield the following

[2]We note that the "method of snapshots" in which one computes the SVD of X^TX instead of of X, is a cost-effective alternative to compute the reduced order basis when the number of FOM DOFs is significantly larger than the number of collected snapshots. This, however, may lead to a loss of orthonormality in the basis vectors. In this work we always use the SVD of the snapshot matrix X and, for the examples considered in the tests, the columns of $\widetilde{U}_i$ have remained orthonormal to machine precision.

split reduced basis formulation:

$$\begin{bmatrix} \widetilde{M}_{1,\gamma} & \widetilde{M}_{1,\gamma 0} & 0 & 0 & \widetilde{G}_1^T \\ \widetilde{M}_{1,0\gamma} & \widetilde{M}_{1,0} & 0 & 0 & 0 \\ 0 & 0 & \widetilde{M}_{2,\gamma} & \widetilde{M}_{2,\gamma 0} & -\widetilde{G}_2^T \\ 0 & 0 & \widetilde{M}_{2,0\gamma} & \widetilde{M}_{2,0} & 0 \\ \widetilde{G}_1 & 0 & -\widetilde{G}_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\varphi}}_{1,\gamma} \\ \dot{\boldsymbol{\varphi}}_{1,0} \\ \dot{\boldsymbol{\varphi}}_{2,\gamma} \\ \dot{\boldsymbol{\varphi}}_{2,0} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_{1,\gamma} \\ \boldsymbol{s}_{1,0} \\ \boldsymbol{s}_{2,\gamma} \\ \boldsymbol{s}_{2,0} \\ -G_1\dot{\boldsymbol{\beta}}_{1,\gamma} + G_2\dot{\boldsymbol{\beta}}_{2,\gamma} \end{bmatrix} \tag{4.2}$$

where for $i \in \{1,2\}$ and $\{j,k\} \in \{\gamma, 0\}$,

$$\widetilde{M}_{i,jk} := \widetilde{U}_{i,j}^T M_{i,jk} \widetilde{U}_{i,k}, \quad \widetilde{K}_{i,jk} := \widetilde{U}_{i,j}^T K_{i,jk} \widetilde{U}_{i,k}, \quad \widetilde{G}_i := G_i \widetilde{U}_{i,\gamma}, \quad \widetilde{\boldsymbol{f}}_{i,j} := \widetilde{U}_{i,j}^T \boldsymbol{f}_{i,j} \tag{4.3}$$

and

$$\begin{aligned} \begin{bmatrix} \boldsymbol{s}_{i,\gamma} \\ \boldsymbol{s}_{i,0} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{f}}_{i,\gamma} \\ \widetilde{\boldsymbol{f}}_{i,0} \end{bmatrix} - \begin{bmatrix} \widetilde{K}_{i,\gamma} & \widetilde{K}_{i,\gamma 0} \\ \widetilde{K}_{i,0\gamma} & \widetilde{K}_{i,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varphi}_{i,\gamma} \\ \boldsymbol{\varphi}_{i,0} \end{bmatrix} \\ - \begin{bmatrix} \widetilde{U}_{i,\gamma}^T K_{i,\gamma} & \widetilde{U}_{i,\gamma}^T K_{i,\gamma 0} \\ \widetilde{U}_{i,0}^T K_{i,0\gamma} & \widetilde{U}_{i,0}^T K_{i,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_{i,\gamma} \\ \boldsymbol{\beta}_{i,0} \end{bmatrix} - \begin{bmatrix} \widetilde{U}_{i,\gamma}^T M_{i,\gamma} & \widetilde{U}_{i,\gamma} M_{i,\gamma 0} \\ \widetilde{U}_{i,0}^T M_{i,0\gamma} & \widetilde{U}_{i,0} M_{i,0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\beta}}_{i,\gamma} \\ \dot{\boldsymbol{\beta}}_{i,0} \end{bmatrix} \end{aligned}$$

Let $\widetilde{Q}_i := \widetilde{M}_{i,\gamma 0} \widetilde{M}_{i,0}^{-1} \widetilde{M}_{i,0\gamma}$ and $\widetilde{P}_i = \widetilde{M}_{i,\gamma} - \widetilde{Q}_i$, $i = 1, 2$. Elimination of the interior degrees of freedom from (4.2) then yields the following linear system:

$$\begin{bmatrix} \widetilde{P}_1 & 0 & 0 & 0 & \widetilde{G}_1^T \\ \widetilde{Q}_1 & \widetilde{M}_{1,\gamma 0} & 0 & 0 & 0 \\ 0 & 0 & \widetilde{P}_2 & 0 & -\widetilde{G}_2^T \\ 0 & 0 & \widetilde{Q}_2 & \widetilde{M}_{2,\gamma 0} & 0 \\ \widetilde{G}_1 & 0 & -\widetilde{G}_2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\varphi}}_{1,\gamma} \\ \dot{\boldsymbol{\varphi}}_{1,0} \\ \dot{\boldsymbol{\varphi}}_{2,\gamma} \\ \dot{\boldsymbol{\varphi}}_{2,0} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_{1,\gamma} - \widetilde{M}_{1,\gamma 0} \widetilde{M}_{1,0}^{-1} \boldsymbol{s}_{1,0} \\ \widetilde{M}_{1,\gamma 0} \widetilde{M}_{1,0}^{-1} \boldsymbol{s}_{1,0} \\ \boldsymbol{s}_{2,\gamma} - \widetilde{M}_{2,\gamma 0} \widetilde{M}_{2,0}^{-1} \boldsymbol{s}_{2,0} \\ \widetilde{M}_{2,\gamma 0} \widetilde{M}_{2,0}^{-1} \boldsymbol{s}_{2,0} \\ \boldsymbol{0} \end{bmatrix} \tag{4.4}$$

Because the columns of $\widetilde{U}_{i,\gamma}$ and $\widetilde{U}_{i,0}$ are orthonormal, one can show that the matrices $\widetilde{P}_i$ are invertible and so, one can formally define the Schur complement matrix $\widetilde{S} := \widetilde{G}_1 \widetilde{P}_1^{-1} \widetilde{G}_1^T + \widetilde{G}_2 \widetilde{P}_2^{-1} \widetilde{G}_2^T$ and the following linear system

$$\widetilde{S}\boldsymbol{\lambda} = \widetilde{G}_1 \widetilde{P}_1^{-1} (\boldsymbol{s}_{1,\gamma} - \widetilde{M}_{1,\gamma 0} \widetilde{M}_{1,0}^{-1} \boldsymbol{s}_{1,0}) - \widetilde{G}_2 \widetilde{P}_2^{-1} (\boldsymbol{s}_{2,\gamma} - \widetilde{M}_{2,\gamma 0} \widetilde{M}_{2,0}^{-1} \boldsymbol{s}_{2,0})$$

for the Lagrange multiplier in (4.4).

4.2. Reduced Lagrange multiplier space. We now focus on the second condition necessary for the well-posedness of the Schur complement matrix, i.e., ensuring that the constraint matrix $(\widetilde{G}_1, \widetilde{G}_2)^T$ has a full column rank. To that end, assume that the FEM matrices G_i are defined using the trace of one of the subdomain FEM spaces on γ for the Lagrange multiplier. We recall that with this choice the monolithic FEM-FEM problem (2.4) is well-posed [17]. Let N_γ be the dimension of this Lagrange multiplier space. Then, the matrices $\widetilde{G}_i$ in the split reduced basis formulation (4.4) have dimension $N_\gamma \times N_{R,i\gamma}$ where $N_{R,i\gamma}$ the number of reduced basis modes on the interface of Ω_i.

It is clear that $(\widetilde{G}_1, \widetilde{G}_2)^T$ cannot have a full column rank unless $N_{R,1\gamma} + N_{R,2\gamma} > N_\gamma$. Since effective ROM requires reduced bases with the smallest possible dimensions $N_{R,i\gamma}$, this inequality will likely be violated unless one artificially keeps the size of the reduced basis large enough. Our second modification aims to resolve this problem by reducing the size of the Lagrange multiplier (LM) space to some $N_{R,\gamma} < N_\gamma$ such that

$$N_{R,1\gamma} + N_{R,2\gamma} > N_{R,\gamma}. \tag{4.5}$$

By assumption, the original Lagrange multiplier space was chosen to be the trace space of the FEM space on Ω_i for $i = 1$ or $i = 2$. Thus, we can choose the associated matrix $\widetilde{U}_{i,\gamma}$ as the reduced basis matrix for the LM space. Specifically, we define $\widetilde{U}_{LM} := \widetilde{U}_{1,\gamma}$ if the LM space was taken to be the trace space of Ω_1 and $\widetilde{U}_{LM} := \widetilde{U}_{2,\gamma}$ if the LM space was the trace space of Ω_2. With these choices $N_{R,\gamma} = N_{R,1\gamma}$ or $N_{R,\gamma} = N_{R,2\gamma}$ and (4.5) is trivially satisfied.

To incorporate this modification into (4.4) we insert the ansatz $\boldsymbol{\lambda} = \widetilde{U}_{LM}\boldsymbol{\ell}$, where $\boldsymbol{\ell}$ is the reduced basis Lagrange multiplier, into (4.2) and multiply the constraint equation by $\widetilde{U}_{LM}^T$. The new problem has the same form as (4.2), but with $\widetilde{G}_i$ now defined as $\widetilde{G}_i =: \widetilde{U}_{LM}^T G_i \widetilde{U}_{i,\gamma}$. As a result, the linear system for the reduced basis Lagrange multiplier $\boldsymbol{\ell}$ is now given by

$$\widetilde{S}\boldsymbol{\ell} = \widetilde{G}_1\widetilde{P}_1^{-1}(\boldsymbol{s}_{1,\gamma} - \widetilde{M}_{1,\gamma 0}\widetilde{M}_{1,0}^{-1}\boldsymbol{s}_{1,0}) - \widetilde{G}_2\widetilde{P}_2^{-1}(\boldsymbol{s}_{2,\gamma} - \widetilde{M}_{2,\gamma 0}\widetilde{M}_{2,0}^{-1}\boldsymbol{s}_{2,0}) \tag{4.6}$$

where the Schur complement $\widetilde{S} := \widetilde{G}_1\widetilde{P}_1^{-1}\widetilde{G}_1^T + \widetilde{G}_2\widetilde{P}_2^{-1}\widetilde{G}_2^T$ is provably invertible. Indeed, assuming the conditions in [17], the FOM matrix $(G_1, G_2)^T$ is guaranteed to have a full column rank. The columns of $\widetilde{U}_{LM}^T$ and $\widetilde{U}_{i,\gamma}$ are orthonormal and so, they have full column rank by construction. As a result, pre and post multiplication of G_i by these matrices preserves the column rank of the FOM matrix, i.e., $(\widetilde{G}_1, \widetilde{G}_2)^T$ also has full column rank.

Remark 1. *Although the full subdomain basis system* (3.2) *cannot be guaranteed to have a well-defined Schur complement, a use of a reduced basis LM space in this system can help avoid its over constraining. We implemented such a version of* (3.2) *using the reduced basis LM space outlined in this section, i.e., setting $\boldsymbol{\lambda} = \widetilde{U}_{LM}\boldsymbol{\ell}$ in* (3.2)*, where $\widetilde{U}_{LM}$ is defined as described above. Numerical results in Section 5 confirm that the full subdomain basis scheme implemented with this reduced Lagrange multiplier space performs very well, although it still lacks the theoretical guarantees of the split basis scheme.*

4.3. Split basis ROM-ROM algorithm. Together, the modifications in Sections 4.1-4.2 lead to the following split reduced basis extension of the IVR scheme to ROM-ROM couplings.

Offline: Computation of the split basis ROMs.

1. Using a suitable monolithic FOM solve the model problem (2.1) on Ω, collect solution snapshots into a snapshot matrix $\overline{X}$ and partition $\overline{X}$ into subdomain snapshot matrices $\overline{X}_i$, $i = 1, 2$. Compute two SVDs from the adjusted snapshot matrix: $X_{i,0} = U_{i,0}\Sigma_{i,0}V_{i,0}^T$ for the interior DOFs and $X_{i,\gamma} = U_{i,\gamma}\Sigma_{i,\gamma}V_{i,\gamma}^T$ for the interface DOFs.

2. Given thresholds $\delta_{i,0}, \delta_{i,\gamma} > 0$ for $i = 1, 2$, define the reduced bases dimensions $N_{R,i0}$ and $N_{R,i\gamma}$ as the smallest integers such that

$$\sum_{k=1}^{N_{R,i0}} \sigma_{i,k0}^2 \geq \Big(1 - \delta_{i,0}\Big)\sum_{k=1}^{N_0}\sigma_{i,k0}^2 \quad \text{and} \quad \sum_{k=1}^{N_{R,i\gamma}} \sigma_{i,k\gamma}^2 \geq \Big(1 - \delta_{i,\gamma}\Big)\sum_{k=1}^{N_\gamma}\sigma_{i,k\gamma}^2$$

   Define the reduced basis matrices $\widetilde{U}_{i,0}, \widetilde{U}_{i,\gamma}$ by retaining the first $N_{R,i0}$ columns of $U_{i,0}$ and the first $N_{R,i\gamma}$ columns of $U_{i,\gamma}$.

3. Choose $\widetilde{U}_{LM}$ to be $\widetilde{U}_{1,\gamma}$ or $\widetilde{U}_{2,\gamma}$. For $i = 1, 2$ and $\{j, k\} \in \{0, \gamma\}$, precompute the ROM matrices:

$$\widetilde{M}_{i,jk} := \widetilde{U}_{i,j}^T M_{i,jk}\widetilde{U}_{i,k} \in \mathbb{R}^{N_{R,ij}\times N_{R,ik}}; \quad \widetilde{K}_{i,jk} := \widetilde{U}_{i,j}^T K_{i,jk}\widetilde{U}_{i,k} \in \mathbb{R}^{N_{R,ij}\times N_{R,ik}};$$
$$\widetilde{G}_i := \widetilde{U}_{LM}^T G_i \widetilde{U}_{i,\gamma} \in \mathbb{R}^{N_{R,\gamma}\times N_{R,i\gamma}}; \quad \widetilde{P}_i := \widetilde{M}_{i,\gamma} - \widetilde{M}_{i,\gamma 0}\widetilde{M}_{i,0}^{-1}\widetilde{M}_{i,0\gamma} \in \mathbb{R}^{N_{R,i\gamma}\times N_{R,i\gamma}}$$

Online: Solution of the split basis ROM-ROM system

1. Given a simulation time interval $[0, T]$ choose an explicit time integration scheme, or an IMEX scheme with explicit treatment of $\boldsymbol{\lambda}$, for each subdomain, i.e., an operator $D^n_{j,t}(\boldsymbol{\varphi})$, $j = 1, 2$.
2. For $i = 1, 2$ and $n = 0, 1, \dots$ use $\boldsymbol{\varphi}^n_{i,0}$ and $\boldsymbol{\varphi}^n_{i,\gamma}$ to compute the vectors:

$$\begin{bmatrix} \boldsymbol{s}^n_{i,\gamma} \\ \boldsymbol{s}^n_{i,0} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{f}}^n_{i,\gamma} \\ \widetilde{\boldsymbol{f}}^n_{i,0} \end{bmatrix} - \begin{bmatrix} \widetilde{K}_{i,\gamma} & \widetilde{K}_{i,\gamma 0} \\ \widetilde{K}_{i,0\gamma} & \widetilde{K}_{i,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varphi}^n_{i,\gamma} \\ \boldsymbol{\varphi}^n_{i,0} \end{bmatrix} - \begin{bmatrix} \widetilde{U}^T_{i,\gamma} K_{i,\gamma} & \widetilde{U}^T_{i,\gamma} K_{i,\gamma 0} \\ \widetilde{U}^T_{i,0} K_{i,0\gamma} & \widetilde{U}^T_{i,0} K_{i,0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}^n_{i,\gamma} \\ \boldsymbol{\beta}^n_{i,0} \end{bmatrix}$$
$$- \begin{bmatrix} \widetilde{U}^T_{i,\gamma} M_{i,\gamma} & \widetilde{U}_{i,\gamma} M_{i,\gamma 0} \\ \widetilde{U}^T_{i,0} M_{i,0\gamma} & \widetilde{U}_{i,0} M_{i,0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\beta}}^n_{i,\gamma} \\ \dot{\boldsymbol{\beta}}^n_{i,0} \end{bmatrix}$$

3. Solve the Schur complement system

$$\left(\widetilde{G}_1 \widetilde{P}_1^{-1} \widetilde{G}_1^T + \widetilde{G}_2 \widetilde{P}_2^{-1} \widetilde{G}_2^T\right) \boldsymbol{\ell}^n = \widetilde{G}_1 \widetilde{P}_1^{-1} (\boldsymbol{s}^n_{1,\gamma} - \widetilde{M}_{1,\gamma 0} \widetilde{M}^{-1}_{1,0} \boldsymbol{s}^n_{1,0})$$
$$- \widetilde{G}_2 \widetilde{P}_2^{-1} (\boldsymbol{s}_{2,\gamma} - \widetilde{M}^n_{2,\gamma 0} \widetilde{M}^{-1}_{2,0} \boldsymbol{s}^n_{2,0})$$

   for $\boldsymbol{\ell}^n$. Compute $\widetilde{G}_1^T \boldsymbol{\ell}^n$ and $\widetilde{G}_2^T \boldsymbol{\ell}^n$.
4. Solve the systems

$$\begin{bmatrix} \widetilde{M}_{i,\gamma} & \widetilde{M}_{i,\gamma 0} \\ \widetilde{M}_{0\gamma} & \widetilde{M}_0 \end{bmatrix} D^n_{i,t}\left(\begin{bmatrix} \boldsymbol{\varphi}_{i,\gamma} \\ \boldsymbol{\varphi}_{i,0} \end{bmatrix} \right) = \begin{bmatrix} \boldsymbol{s}^n_{i,\gamma} + (-1)^i \widetilde{G}_i^T \boldsymbol{\ell}^n \\ \boldsymbol{s}^n_{i,0} \end{bmatrix}.$$

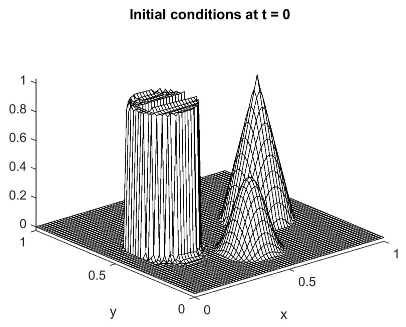
5. Project the ROM solutions to the state spaces of the full order models on Ω_i:

$$\boldsymbol{\Phi}_{i,\gamma} := \widetilde{U}_{i,\gamma} \boldsymbol{\varphi}_{i,\gamma} + \boldsymbol{\beta}_{i,\gamma}; \quad \boldsymbol{\Phi}_{i,0} := \widetilde{U}_{i,0} \boldsymbol{\varphi}_{i,0} + \boldsymbol{\beta}_{i,0}.$$

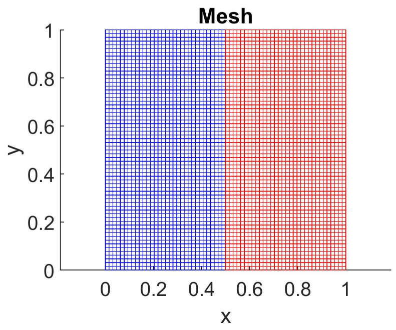
5. Numerical Results. To demonstrate the efficacy of separating the interior and interface DOFs as well as reducing the size of the Lagrange multiplier space, we show results for a solid body rotation test for (2.1) from [14], as in our previous work [3]. The initial conditions for this test problem comprise a cone, a cylinder, and a smooth hump (Figure 5.1(a)). Using the domain $\Omega := (0, 1) \times (0, 1)$, with $\Omega_1 := (0, 0.5) \times (0, 1)$ and $\Omega_2 := (0.5, 1) \times (0, 1)$, we define the advection field $\boldsymbol{u} := (0.5 - y, x - 0.5)$ and diffusion coefficients $\kappa_i := 10^{-5}$, $i = 1, 2$. We impose homogeneous Dirichlet boundaries on all non-interface boundaries $\Gamma_i, i = 1, 2$ and set the final time to be 2π, representing one full rotation. All results in this section were obtained by using an IMEX version of the Crank-Nicholson time-stepping scheme in which the Lagrange multiplier terms in Step 3 of the Online phase of the algorithm are treated explicitly.

For the FEM discretizations, we spatially discretize Ω by setting $\Delta x = \Delta y = \frac{1}{64}$, yielding 4225 DOFs in Ω, and 2145 DOFs in Ω_i for $i = 1, 2$ as seen in Figure 5.1(b). Snapshots are collected from a monolithic FEM solution using the snapshot time step $\Delta t_s = 6.734 \times 10^{-3}$. All coupled problems are solved with $\Delta t = 3.367 \times 10^{-3}$, which is determined from the Courant-Friedrichs-Lewy (CFL) condition. We present results for ROM-ROM couplings using the full subdomain basis and the split basis, each with a full size or a reduced size Lagrange multiplier space. For comparison, the FEM-FEM coupling is shown as well. All cases shown here are reproductive, meaning that we solve the ROM-ROM problems to the same final time 2π with the same diffusion coefficient and initial conditions.

With the snapshot time step set to $\Delta t_s = 6.734 \times 10^{-3}$, 933 snapshots are collected. A prerequisite for an effective ROM is the rapid decay of the singular values. We first



(a) Initial conditions



(b) Meshes used to discretize Ω_1 (blue) and Ω_2 (red)

Fig. 5.1. *Initial conditions, domain partitioning, and mesh for the model 2D transmission problem.*

confirm that this is indeed the case and that most of the energy is contained within a much smaller subset of the snapshots. Let d represent the number of reduced basis modes retained from the singular value decomposition of the snapshot matrix. Then the snapshot energy is defined as

$$\mathcal{E} := \frac{\sum_{i=1}^{d} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2}, \tag{5.1}$$

where N is the number of singular values.

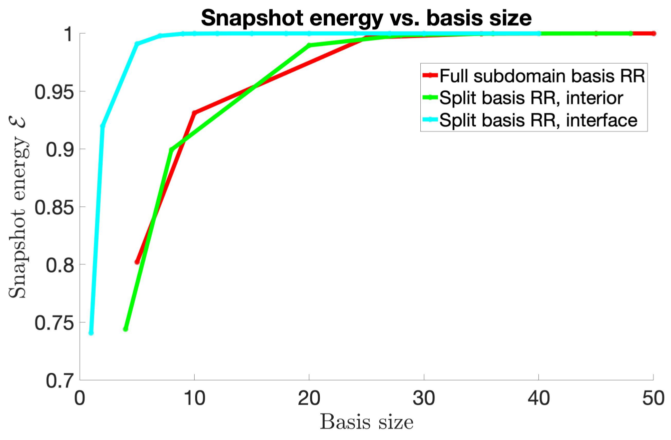


Fig. 5.2. *Snapshot energy* (5.1) *as a function of the POD basis size for each ROM-ROM coupling type.*

In Figure 5.2, we see that for each of the ROM couplings, no more than 50 modes are needed to capture 99.999% of the snapshot energy. In fact, for the full subdomain basis ROM-ROM coupling, about 25 modes are needed to capture 99% of the energy, whereas for the split basis ROM-ROM coupling, about 20 interior modes are needed and only 5 interface nodes are needed to capture 99% of their respective snapshot energies. For this reason, we only show results up until a basis size of about 120, as all the significant energy is contained in these modes.

First, we examine the relative errors, measured against the monolithic FEM solution. We define these errors as

$$\epsilon := \frac{||X_{2\pi} - F_{2\pi}||_2}{||F_{2\pi}||_2} \tag{5.2}$$

where $F_{2\pi}$ represents the monolithic FEM solution at final time $T = 2\pi$, and $X_{2\pi}$ represents one of the coupled solutions (either ROM-ROM, or FEM-FEM) at final time $T = 2\pi$. For the split basis ROM-ROM coupling, using a full or reduced size LM space produces the same relative errors up to machine precision, so we only display errors for the reduced LM space.

Note that in Figure 5.3 and Figure 5.4, the basis size shown on the x-axis is the total size of the reduced basis employed on each subdomain. For the full subdomain basis ROM-ROM coupling, this is just the number of modes retained for each subdomain. For the split basis ROM-ROM, the total reduced basis size equals the sum of the interior and interfacial modes retained on each subdomain, i.e., $N_{R,10} + N_{R,1\gamma}$.

In each split basis ROM-ROM simulation, we set the size of the interfacial basis to be one-fifth of the total subdomain basis size. Since the size of the reduced basis for the LM is defined in terms of one of the interfacial bases, the size for the reduced LM space is also one-fifth of the total subdomain basis size. Thus, we have

$$N_{R,i\gamma} = \frac{1}{5}\Big(N_{R,i\gamma} + N_{R,i0}\Big) \implies N_{R,i\gamma} = \frac{1}{4}N_{R,i0}.$$

The interfacial basis size (and thus the reduced LM size) does have a fixed maximum, however, and so $N_{R,i\gamma}$ cannot be greater than 63 for our particular mesh discretization. This yields

$$N_{R,i\gamma} = \min\Big\{\frac{1}{4}N_{R,i0}, 63\Big\}.$$

In Figure 5.3, the notation "fLM" refers to the full size Lagrange multiplier space, which is of dimension 63, and the notation "rLM" refers to the reduced size Lagrange multiplier space. We observe that using the reduced Lagrange multiplier space in the full subdomain basis formulation (3.2) eliminates the large spike in errors observed around a basis size of 50. Also, as expected for a reproductive test, the error in all couplings approaches the FEM-FEM coupling error as the total size of the reduced basis is increased. The full-subdomain basis formulations[3] are actually able to very slightly outperform the FEM-FEM formulation, with an error of about 0.0032 versus 0.0036. The split basis ROM-ROM formulations both asymptotically approach a relative error of about 0.0045. Using the split basis ROM-ROM formulation produces errors on the same order as the full subdomain basis ROM-ROM coupling with the reduced LM space. This supports the hypothesis that our new formulation, namely, separating the DOFs in the ROM construction, does not introduce extra errors to the coupling.

Next we examine the condition number of the Schur complement in each one of our coupling formulations. This number is a measure of the Schur complement's "well-posedness" and can be used to assess the success of our modifications. Recall that our proposed solutions to ensuring a well-conditioned Schur complement matrix were to both separate out

[3]We note that as the size of the reduced basis increases past the dimension of the full size Lagrange multiplier space, the full subdomain basis formulation (3.2) recovers its accuracy because it ceases to be over-constrained anymore.

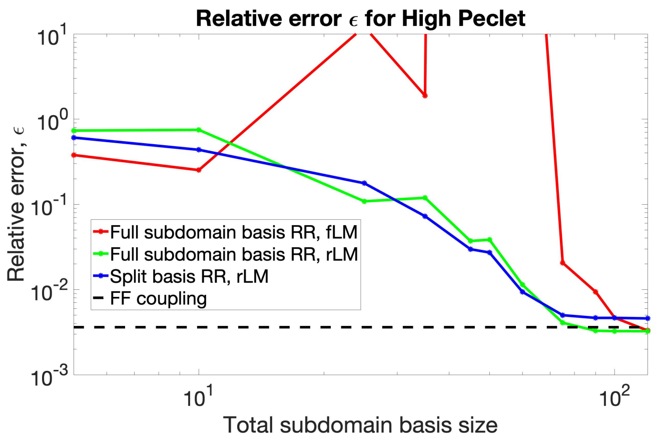


Fig. 5.3. *Relative error* (5.2) *as a function of the POD basis size for each coupling type. Note: errors for "split basis RR, fLM" and "split basis RR, rLM" are identical to machine precision, so only "split basis RR, rLM" is shown.*

the interfacial and interior DOFs in constructing ROM bases, as well as to reduce the size of the Lagrange multiplier space. As seen in Figure 5.4, taken together, these two modifications do indeed empirically fix the condition number. First, observe that using the full order Lagrange multiplier space, with either the full subdomain or split basis ROM-ROM coupling, results in very high condition numbers. While the condition number does decrease as basis size increases for the full subdomain basis ROM-ROM coupling, it is still high compared to the FEM-FEM coupling, and it only reaches a reasonable scale when the reduced basis size is larger than we wish to implement. Using a reduced size Lagrange multiplier space dramatically decreases the condition number. With this tweak, the split basis ROM-ROM formulation ($1.0 \leq \text{cond(S)} \leq 5.7$), is able to outperform the full subdomain basis ROM-ROM formulation ($1.0 \leq \text{cond(S)} \leq 1.8 \times 10^4$). In particular, the conditioning of the Schur complement for the split basis ROM-ROM formulation is essentially the same as for the FEM-FEM coupling, for which the Schur complement matrix was proven to be well-conditioned in [17].

In the Pareto plot in Figure 5.5, we see again that each coupling variant is capable of obtaining an error on the order of the relative error for the FEM-FEM coupling. The reduced LM space formulations are able to achieve this error in less time than it takes the FEM-FEM coupling, while the split basis ROM-ROM with full LM space takes longer to achieve this optimal error. It can be observed from Figure 5.5 that the full subdomain basis ROM-ROM with the reduced Lagrange multiplier space is the method of choice for this problem. Again, we note that the full subdomain basis ROM-ROM with full LM space experiences a sharp jump in errors (likely due to the ill-conditioning of the Schur complement system for smaller basis sizes), but eventually it is able to also obtain the optimal error for one particular choice of reduced basis size.

Next, in Figure 5.6, we show the ability of the ROM formulations to enforce the interface condition. Although these are only for selected basis sizes, we note that the split basis formulations are actually able to capture the behavior on the interface accurately by a basis size of about 45, while the full basis formulations take a larger basis size to perform this well. However, all four ROM-ROM formulations are able to enforce alignment on the interface as well as the FEM-FEM coupling, provided the basis size is large enough.

Lastly, we show a visual comparison of the simulation results by the four different ROM-

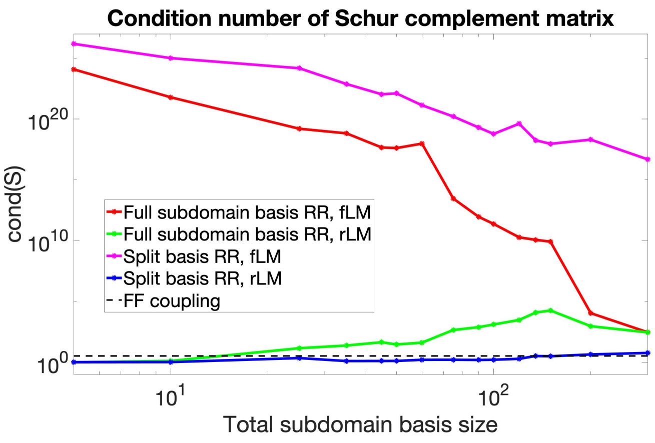


Fig. 5.4. *Condition number of Schur complement matrix as a function of the POD basis size for each coupling type.*

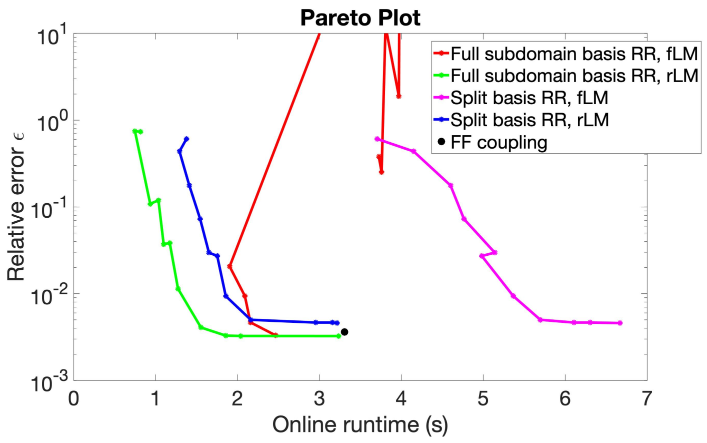
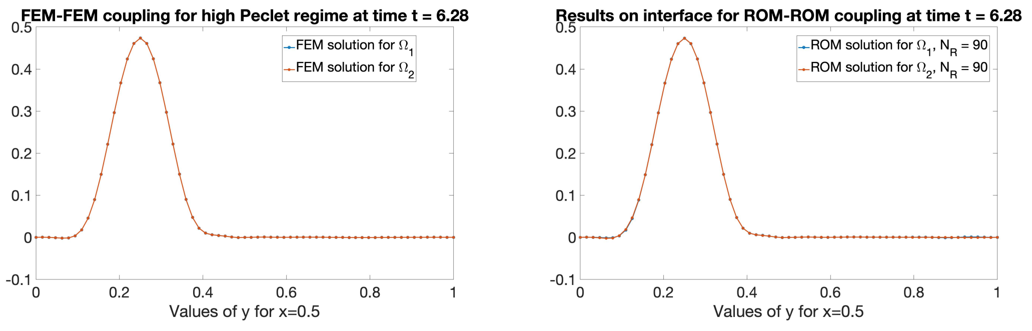


Fig. 5.5. *Pareto plot showing relative error* (5.2) *as a function of online run time for each coupling formulation. Each point on the plot represents a different subdomain basis size.*

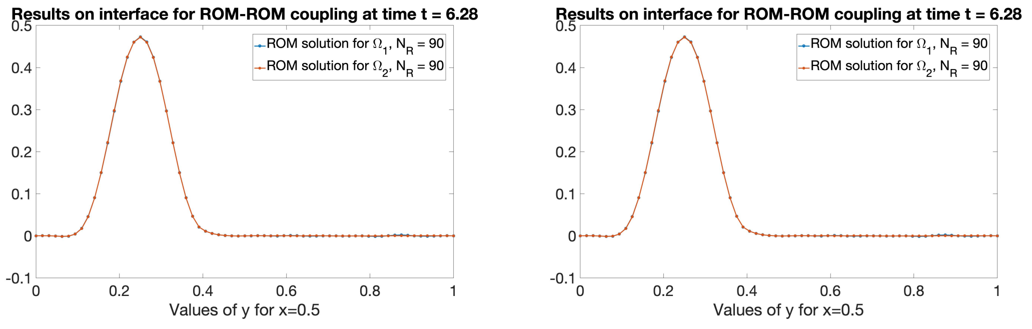
ROM couplings at the final time. Each of the following results uses a total of 50 reduced basis modes in each subdomain. For the split ROM-ROM formulation, this amounts to 40 interior and 10 interfacial basis modes. This is one of the cases that we saw the spike in relative error for the full subdomain basis ROM-ROM with full LM space in Figure 5.3. As seen in Figure 5.7, using either the split basis or the reduced LM space provide drastic improvements over the full subdomain basis with full LM space, for this particular basis size. Both of the split basis ROM-ROM couplings (Figures 5.7(c)-5.7(d)) perform better in the eye ball norm than the full subdomain basis ones (Figures 5.7(a)-5.7(b)).

6. Conclusions.

While implementing ROMs on individual subdomains of partitioned problems provides valuable savings in computational time, care needs to be taken in extending existing coupling formulations designed for FOMs to work with ROMs. For our partitioned approach in particular, the dual Schur complement expressing the Lagrange multiplier is the key component, as it provides the interface flux needed to independently solve the subdomains at each time step. This system must be well-conditioned for the



(a) FEM-FEM formulation (b) Full subdomain basis ROM-ROM; reduced LM



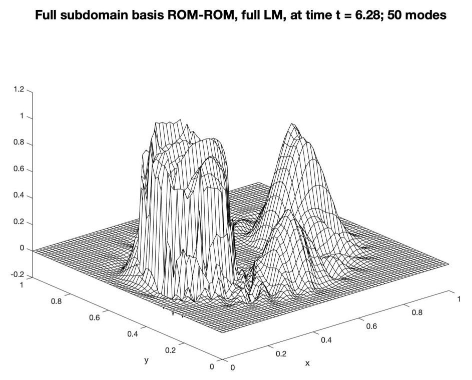
(c) Split basis ROM-ROM; full LM (d) Split basis ROM-ROM; reduced LM

FIG. 5.6. *Results on the interface at T_f for $N_{R,i} = 90$ for FEM-FEM and selected ROM-ROM coupling formulations*

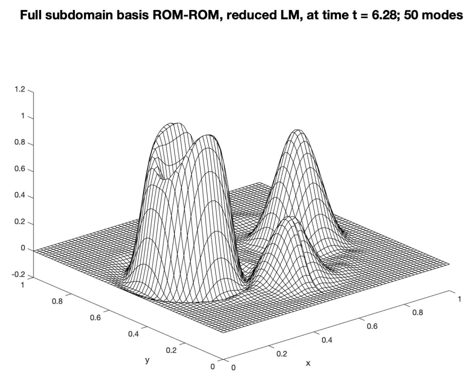
partitioned scheme to be reliable. Our previous numerical studies revealed that the Schur complement of the full subdomain basis ROM-ROM formulation proposed in [3] can become severely ill-conditioned, compromising the accuracy of the coupling. In this work we considered two modifications of our earlier scheme that target two specific conditions necessary to obtain well-posed Schur complements. To ensure invertibility of the projected interfacial mass matrices we deployed a dedicated interfacial reduced basis obtained from snapshots containing only the interface DOFs. To guarantee that the transpose constraint matrix has a full column rank we replaced the standard finite element Lagrange multiplier space by a reduced basis one taken to be one of the two interfacial reduced bases.

Our future work will focus on rigorous analysis of the Schur complement conditioning for the split basis formulation, demonstration of the scheme in the predictive regime and examining other methods for constructing the reduced Lagrange multiplier space. Additionally, towards our goal of designing a workflow that can reuse existing codes for individual physics that may not be coupled with each other, we plan to examine snapshot collection strategies that do not require simulating the coupled high-fidelity model over the the entire domain, e.g., oversampling [20].

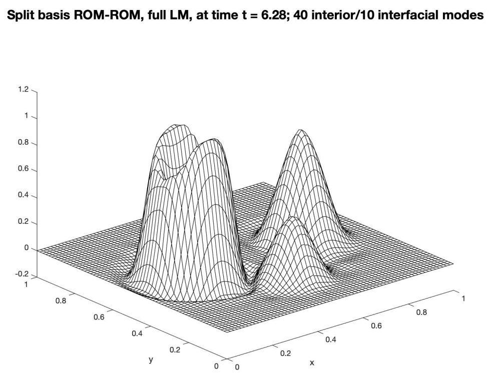
Finally, we plan to consider extensions of our ROM-ROM and ROM-FEM partitioned schemes to nonlinear models. Indeed, the reduced order operators defining our ROM systems, such as (3.3) and (4.3), can only be precomputed in the case of a linear partial differential equation (PDE). For nonlinear problems, to circumvent the computational cost of computing the projection of the nonlinear terms in the PDEs onto the reduced basis online, hyper-reduction approaches such as the Discrete Empirical Interpolation Method



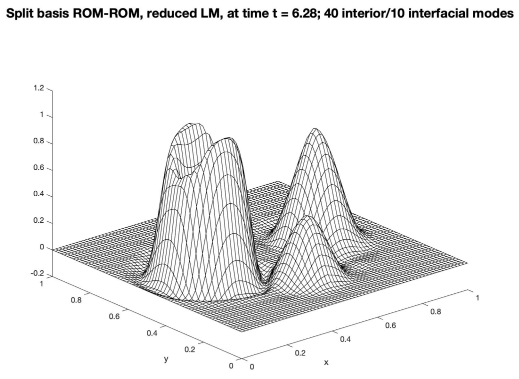
(a) Full subdomain basis ROM-ROM; full LM



(b) Full subdomain basis ROM-ROM; reduced LM



(c) Split basis ROM-ROM; full LM



(d) Split basis ROM-ROM; reduced LM

FIG. 5.7. *Results at T_f for $N_{R,i} = 50$ for each ROM-ROM coupling formulation*

(DEIM) [1], gappy POD [5] and the Energy-Conserving Sampling and Weighting (ECSW) method [6] can be employed and will be considered in our work.

REFERENCES

[1] S. Chaturantabut and D. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput., 32 (2010), pp. 2737–2764.

[2] J. M. Connors and K. C. Sockwell, *(2022): A multirate discontinuous-galerkin-in-time framework for interface-coupled problems.*, SIAM Journal on Numerical Analysis., To appear (2022).

[3] A. de Castro, P. Kuberry, I. Tezaur, and P. Bochev, *A Novel Partitioned Approach for Reduced Order Model – Finite Element Model (ROM-FEM) and ROM-ROM Coupling*, ArXiv pre-print, to appear in Proceedings of the 2022 Earth and Space conference. https://arxiv.org/abs/2206.04736.

[4] J. L. Eftang and A. T. Patera, *Port reduction in parametrized component static condensation: approximation and a posteriori error estimation*, International Journal for Numerical Methods in Engineering, 96 (2013), pp. 269–302.

[5] R. Everson and L. Sirovich, *Karhunen-Loeve procedure for gappy data*, J. Optical Society of America A, (1995), pp. 1657–1664.

[6] C. Farhat, T. Chapman, and P. Avery, *Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models*, International journal for numerical methods in engineering, 102 (2015),

pp. 1077–1110.

[7] C. A. Felippa, K. Park, and C. Farhat, *Partitioned analysis of coupled mechanical systems*, Computer Methods in Applied Mechanics and Engineering, 190 (2001), pp. 3247–3270. Advances in Computational Methods for Fluid-Structure Interaction.

[8] A. Gravouil and A. Combescure, *Multi-time-step explicit–implicit method for non-linear structural dynamics*, International Journal for Numerical Methods in Engineering, 50 (2001), pp. 199–225.

[9] A. Gravouil, A. Combescure, and M. Brun, *Heterogeneous asynchronous time integrators for computational structural dynamics*, International Journal for Numerical Methods in Engineering, 102 (2014), pp. 202–232.

[10] M. D. Gunzburger, J. S. Peterson, and J. N. Shadid, *Reduced-order modeling of time-dependent pdes with multiple parameters in the boundary data*, Computer Methods in Applied Mechanics and Engineering, 196 (2007), pp. 1030–1047.

[11] C. Hoang, Y. Choi, and K. Carlberg, *Domain-decomposition least-squares petrov–galerkin (dd-lspg) nonlinear model reduction*, Computer Methods in Applied Mechanics and Engineering, 384 (2021), p. 113997.

[12] P. Holmes, J. Lumley, and G. Berkooz, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*, Cambridge University Press, 1996.

[13] L. Iapichino, A. Quarteroni, and G. Rozza, *Reduced basis method and domain decomposition for elliptic problems in networks and complex parametrized geometries*, Computers & Mathematics with Applications, 71 (2016), pp. 408–430.

[14] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 627–665.

[15] Y. Maday and E. M. Ronquist, *A reduced-basis element method*, Comptes Rendus Mathematique, 335 (2002), pp. 195–200.

[16] Y. Maday and E. M. Ronquist, *The reduced basis element method: Application to a thermal fin problem*, SIAM Journal on Scientific Computing, 26 (2004), pp. 240–258.

[17] K. Peterson, P. Bochev, and P. Kuberry, *Explicit synchronous partitioned algorithms for interface problems based on lagrange multipliers*, Computers & Mathematics with Applications, 78 (2019), pp. 459–482.

[18] Phuong Huynh, Dinh Bao, Knezevic, David J., and Patera, Anthony T., *A static condensation reduced basis element method : approximation and a posteriori error estimation*, ESAIM: M2AN, 47 (2013), pp. 213–251.

[19] L. Sirovich, *Turbulence and the dynamics of coherent structures, part iii: dynamics and scaling*, Q. Appl. Math., 45 (1987), pp. 583–590.

[20] K. Smetana and T. Taddei, *Localized model reduction for nonlinear elliptic partial differential equations: localized training, partition of unity, and adaptive enrichment*, 2022.

[21] K. C. Sockwell, K. Peterson, P. Kuberry, P. Bochev, and N. Trask, *Interface flux recovery coupling method for the ocean–atmosphere system*, Results in Applied Mathematics, 8 (2020), pp. 100–110.

BAYESIAN FUNCTION APPROXIMATION USING FUNCTIONAL TENSOR-TRAINS WITH INVESTIGATIONS INTO UNCERTAINTY QUANTIFICATION AND ESTIMATION OF DYNAMICAL SYSTEMS

NICHOLAS GALIOTO*, COSMIN SAFTA†, JOHN D. JAKEMAN‡, AND ALEX A. GORODETSKY§

Abstract. Spatio-temporal systems are ubiquitous in many engineering disciplines such as climate modeling, materials engineering, and fluid dynamics. However, learning models of such systems is challenging because the high-dimensionality of their state-space necessitates learning models with large numbers of unknown coefficients, often from limited data. To address these challenges, we investigate the use of functional tensor-trains, which are extremely expressive parsimonious model forms whose number of coefficients scale linearly with dimension, with computationally efficient variational inference algorithms for learning dynamical systems with quantified uncertainty.

1. Introduction.

Models of physical systems are useful for a variety of applications such as ice sheet and climate modeling. Many physical systems of interest are spatio-temporal and thus high-dimensional, which makes generating high-fidelity simulations of the system extremely expensive. Therefore, inexpensive and less accurate models known as surrogates are typically required to explore the range of model behaviors over the range of parameter choices. The surrogate construction requires a rigorous control on the accuracy of the model over the range of dynamics for which the model is used.

In order to tackle high-dimensional complex systems, one requires seemingly conflicting properties, i.e., surrogate model forms need to be sufficiently expressive and yet sufficiently cheap to enable uncertainty quantification (UQ) studies. Typically, creating an inexpensive surrogate model of one's system of interest can address the issue of model expense, but the costs of many of the most popular surrogate modeling techniques scale poorly with dimension. For example, the dynamic mode decomposition [38] technique models a dynamical system in discrete time with a state-transition matrix. With this method, the number of parameters scales quadratically with dimension, and the cost scales cubically. The sparse identification of nonlinear dynamics [7] approach fits a linear combination of basis functions to a system's estimated time derivatives. If a tensor product basis is used for the basis expansion, the number of parameters in this method scales exponentially with the state dimension. Other methods have UQ built into the estimation procedure and can perform UQ without requiring a large number of model evaluations, such as Gaussian process regression [35, 23], polynomial chaos expansions [43, 26], and relevance vector machines [3]. Unfortunately, these approaches also tend to scale poorly with dimension.

One of the most common approaches when working with high-dimensional data is to project the data onto a low-dimensional manifold. These projections can be linear such as principal component analysis (PCA) [42, 36] or active subspaces [10], or they can be nonlinear such as diffusion maps [9, 31], autoencoders [41], or kernel PCA [30, 25]. The drawback of these dimension reduction methods is a loss of interpretability since the data are projected onto an eigenspace or some other mathematical abstraction where the physical meaning is unclear. In applications where the model is intended to provide physical insight into the actual system, this loss of interpretability is especially undesirable. Additionally, these techniques have the added cost during prediction of mapping input data to the low-

*University of Michigan-Ann Arbor, ngalioto@umich.edu
†Sandia National Laboratories, csafta@sandia.gov
‡Sandia National Laboratories, jdjakem@sandia.gov
§University of Michigan-Ann Arbor, goroda@umich.edu

dimensional space and/or outputs to the high-dimensional space.

A promising alternative to dimension reduction is representing functions as functional tensor-trains (FTT). FTTs scale linearly rather than exponentially with dimension and can use the original coordinates of the data, potentially offering better interpretability. Quantifying the uncertainty in FTTs, however, is not straightforward. In general, UQ is oftentimes achieved by drawing samples from an inferred posterior distribution. Using Markov chain Monte Carlo (MCMC) methods to sample the parameter posterior of an FTT, however, can take extremely long to converge due to the fact that FTT parameterizations are non-unique. Thus, more efficient methods of UQ for FTTs are needed. A common alternative to MCMC sampling is variational inference (VI), which approximates a target distribution by solving an optimization problem and can therefore lead to faster convergence. In this report, we investigate the use of VI to sample posterior distributions of FTT model parameters. Specifically, this report contributes the following:

- an investigation on how the choice of FTT structure affects the estimation error on unseen data,
- a comparison of MCMC and VI algorithms for sampling the posterior distribution over the FTT parameters,
- a proof of concept for using FTTs for estimating dynamical systems.

The rest of the report is organized as follows: Section 2 describes methods for function approximation based on tensor decompositions and details FTTs. Next, Section 3 explains Bayesian inference and touches on two of the most powerful inference algorithms: MCMC and VI. Then, Section 4 explains our approach to estimating dynamical systems. In Section 5, numerical results are presented on two test problems. The report ends with conclusions and possible directions for future work in Section 6.

2. Function approximation. In this section, methods for approximating functions with parsimonious representations are described. First, some notation is introduced. Let $\mathbb{R}$ denote the real space, and let $\mathbb{Z}_+$ denote the space of positive integers. In this report, tensors are denoted with bold calligraphic font, e.g., $\boldsymbol{\mathcal{A}}$, matrices with boldface and uppercase font, e.g., $\mathbf{A}$, vectors with boldface and lowercase font, e.g., $\mathbf{a}$, and scalars with italicized and lowercase font, e.g., a. Element $(i_1, i_2, \ldots, i_d)$ of a d-way tensor $\boldsymbol{\mathcal{A}}$ is denoted as $\boldsymbol{\mathcal{A}}_{i_1 i_2 \ldots i_d}$. If the full mode is being indexed, a colon (:) will be used in place of an index. For example, a two-dimensional slice of a three-way tensor $\boldsymbol{\mathcal{X}}$ would be written as $\boldsymbol{\mathcal{X}}_{::i_3}$. Lastly, a normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ is denoted as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and a uniform distribution with lower bound a and upper bound b is denoted as $\mathcal{U}[a, b]$.

2.1. Tensor-train decomposition. There are a number of different methods for representing tensors in compressed forms [28], but we focus here on tensor-train (TT) representations [32]. We choose this representation for its scalability, which allows for inexpensive modeling of high-dimensional/partial differential equation (PDE) systems. The memory and computational requirements of TTs scale linearly with dimension, and as a result, TTs have shown potential as a method for low-dimensional approximation of solutions to PDEs [2, 14, 37]. Furthermore, TTs belong to a broader class of tensor decompositions known as tensor networks. Therefore, they can be generalized to other tensor network topologies that may be more appropriate for certain problems. For example, different tensor network topologies are used to model quantum many-body problems depending on the structure of the particles, e.g., the lattice model geometry and holographic geometry are generated by different tensor network topologies [15]. Thus, any methods developed using the TT decomposition potentially have the flexibility of being applied to a wide range of structured problems. In this section, the TT basics are described.

Let $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ be a d-way tensor with $n_i \in \mathbb{Z}_+$ elements along the ith mode. A

TT representation approximates $\boldsymbol{\mathcal{B}}$ with a tensor $\boldsymbol{\mathcal{A}} \approx \boldsymbol{\mathcal{B}}$ whose elements can be evaluated as

$$\boldsymbol{\mathcal{A}}_{i_1 i_2,\dots i_d} = \boldsymbol{\mathcal{A}}^{(1)}_{1i_1:}\boldsymbol{\mathcal{A}}^{(2)}_{:i_2:}\dots\boldsymbol{\mathcal{A}}^{(d)}_{:i_d1}, \tag{2.1}$$

where each $\boldsymbol{\mathcal{A}}^{(k)} \in \mathbb{R}^{r_{k-1}\times n_k \times r_k}$ is a three-way tensor known as a *TT-core.* The sizes $r_k \in \mathbb{Z}_+$ are known as *TT-ranks*, and for proper dimensions, the condition $r_0 = r_d = 1$ must be satisfied. The ranks can be chosen either by the user or selected adaptively.

The primary advantage of the TT decomposition is that it avoids storage requirements that scale exponentially with dimension. If $r = \max_k r_k$ and $n = \max_k n_k$, then the memory complexity is $\mathcal{O}(dnr^2)$, which scales linearly with dimension d. Moreover, evaluating an element of the tensor can also be completed in linear (in d) time with computational complexity $\mathcal{O}(dr^2)$.

2.2. Tensor-train representation of functions. Tensor-train representations can be used to compactly approximate functions. Here we discuss the advantages and disadvantages of each tensor-train representation, relative to each other and to other approximation methods. In the following subsections, we will consider scalar functions of the form

$$f : X \mapsto \mathbb{R} \tag{2.2}$$

that map a d-dimensional space $X = X_1 \times X_2 \times \dots X_d$ to the real line.

2.2.1. Tensor grid. One of the simplest ways to approximate the function f is to discretize each X_k into $n_k \in \mathbb{Z}_+$ points and store the function value at each point of the finite subset in a d-way tensor $\boldsymbol{\mathcal{A}}$ [27, 12]. With this method, X becomes a tensor grid denoted as $\boldsymbol{\mathcal{X}}$, and $\boldsymbol{\mathcal{A}}$ is viewed as a restriction of f to $\boldsymbol{\mathcal{X}}$, i.e., $f|_{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{A}} : \boldsymbol{\mathcal{X}} \mapsto \mathbb{R}$.

Unfortunately using tensor grid approximations presents a number of challenges:

1. Without compression, this method quickly becomes unwieldy since the number of values that need to be stored is $\prod_{k=1}^{d} n_k$, which scales exponentially with the dimension.
2. Capturing multi-scale features on a given grid requires prior knowledge. In order to represent small-scale changes in the function value, finer grids would be needed in the regions of these features. Prior knowledge of where these regions are is oftentimes unavailable.
3. Function values are not available at arbitrary points in the domain which limits the accuracy of certain operations, e.g., numerically computing gradients and integrals. The accuracy of these numerical operations depends on the grid discretization, but even with fine grids, these methods still only return approximations.
4. Operations with multiple functions, e.g., additions or multiplication, require all functions be defined on the same grid. If the underlying grids are different, performing these operations requires interpolation, which can be inaccurate and adds the expense of decompressing the low-rank representation.

2.2.2. Tensor of coefficients. Some of the aforementioned challenges can be overcome by representing a function f as a linear expansion of a tensor product basis [8, 13]. Let $\psi_{ki_k} : X_k \mapsto \mathbb{R}$ be a univariate basis function for $k = 1, \dots, d$ and $i_k = 1, \dots, n_k$. Then the function f can be approximated by a function $\hat{f}$ defined as

$$\hat{f}(x_1, x_2, \dots, x_d) = \sum_{i_1}^{n_1}\sum_{i_2}^{n_2}\dots\sum_{i_d}^{n_d} \boldsymbol{\mathcal{A}}_{i_1 i_2 \dots i_d}\psi_{1i_1}(x_1)\psi_{2i_2}(x_2)\dots\psi_{di_d}(x_d). \tag{2.3}$$

This representation requires storage of the basis coefficients $\boldsymbol{\mathcal{A}}_{i_1 i_2 \ldots i_d}$. Again, there are $\prod_{k=1}^{d} n_k$ values that are stored in the tensor $\boldsymbol{\mathcal{A}}$, which is then compressed using a tensor decomposition.

Because this function approximation has an analytical expression, derivatives and integrals are analytically tractable, unlike for functions defined numerically on tensor grids. However, the issues with resolving local features and multilinear algebra still persist. If the local features of discontinuities are not known *a priori*, one cannot ensure that the choice of basis functions properly resolves these features. Performing multilinear algebra operations with the compressed representation of a tensor of coefficients requires that all functions share the same parameterization. Otherwise, projection or interpolation techniques must be used, likely adding to the computational expense and error incurred in the computation.

2.2.3. Functional tensor-train. Functional tensor-trains (FTTs) [33, 19] are continuous analogues of the TT decomposition [32] and can be used to address the aforementioned issues of tensor-grid and -coefficient methods. The FTT representation decomposes a function into d core tensors whose sizes depend on the specified or learned ranks r_k. An FTT approximation of the function f is written as

$$\hat{f}(x) = \mathcal{F}^{(1)}(x_1)\mathcal{F}^{(2)}(x_2)\ldots\mathcal{F}^{(d)}(x_d), \tag{2.4}$$

where $\mathcal{F}^{(k)} : X_k \mapsto \mathbb{R}^{r_{k-1}\times r_k}$ are the *FTT cores* and $r_k \in \mathbb{Z}_+$ are the *FTT ranks.* The condition $r_0 = r_d = 1$ is still enforced to ensure proper dimensions. Each FTT core is a collection of matrix-valued univariate functions defined as

$$\mathcal{F}^{(k)}(x_k) = \begin{bmatrix} f^{(k)}_{11}(x_k) & \cdots & f^{(k)}_{1r_k}(x_k) \\ \vdots & & \vdots \\ f^{(k)}_{r_{k-1}1}(x_k) & \cdots & f^{(k)}_{r_{k-1}r_k}(x_k) \end{bmatrix}. \tag{2.5}$$

Proper selection of the FTT ranks is important to avoid overfitting. One way to specify the FTT ranks without prior knowledge is to use a procedure known as rounding combined with cross-validation [20]. The idea of rounding is to decompose the FTT using an SVD-like algorithm, and then truncate as much as possible while keeping the relative error below a specified threshold. A detailed description of rounding can be found in Ref. [19].

The FTT representation addresses all four challenges described in Sec. 2.2.1:

1. A high-dimensional representation of the FTT is not required prior to compression, so the initial exponential memory requirement is avoided.
2. There exist algorithms that can adapt FTTs to properly resolve local features and discontinuities, such as the one described in Ref. [19].
3. Since FTTs are composed of continuous functions, taking derivatives and integrals is straightforward.
4. Lastly, the original coordinates are preserved in the low-rank form, so there is no need for projection or interpolation routines before performing multilinear algebraic operations.

The challenge of using FTTs, however, is that the nonlinear interactions of the parameters can make optimization quite challenging. One strategy for optimization is an algorithm known as alternating least squares (ALS). In ALS, the parameters for all FTT cores except for one are held constant, turning the optimization problem into a linear least squares problem for the parameters of one FTT core at a time. The algorithm loops over each FTT core until the improvement in the cost function falls below a specified threshold. It was shown in Ref. [20] that the gradient descent algorithm can find better estimates than ALS, especially when the number of training data is low. Motivated by these findings, Ref. [21]

developed an automatic differentiation framework for arbitrary tensor networks, which was used in this work to conduct the numerical experiments in Section 5.

3. Bayesian inference.

Using tensor-trains, or any other approximation, with confidence in place of the model it approximates necessitates quantifying the error in the approximation. The uncertainty of a parametric model can be represented as a distribution over the model parameters. Using Bayesian inference, the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ that represents the uncertainty of the parameters after data $\mathcal{D}$ have been collected is given by

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}. \tag{3.1}$$

The distribution $p(\mathcal{D}|\boldsymbol{\theta})$ is known as the likelihood, $p(\boldsymbol{\theta})$ as the prior, and $p(\mathcal{D})$ as the evidence.

3.1. Markov chain Monte Carlo.

In many cases, it is the uncertainty in the output of a model rather than the uncertainty of the model parameters that is of interest. This is especially true with black-box models where the parameters do not represent physical quantities. One way of obtaining the output uncertainty from the parameter uncertainty is to draw samples from the parameter posterior and push them through the model to generate samples of the output. The main challenge in this approach is drawing samples from the parameter posterior. When the posterior distribution is non-Gaussian and analytically intractable, which is almost always the case, drawing independent samples with methods like inverse transform sampling is not possible. A less efficient alternative to drawing independent samples is to instead draw autocorrelated samples using a Markov chain Monte Carlo (MCMC) algorithm [24]. This approach, however, typically requires hundreds of thousands, if not millions, of samples to reach an appropriate effective sample size. The cost of each sample includes at least one evaluation of the posterior, which makes this option infeasible when the posterior evaluation is expensive. Moreover, convergence of the MCMC sampler is especially tricky when the posterior is multi-modal, i.e., regions of high probability are separated by regions of low probability, or there are complicated correlations between the parameters.

3.2. Variational inference.

Variational inference (VI) [5] is an alternative to MCMC sampling that approximates the posterior using a distribution that is easy and computationally inexpensive to sample from. Furthermore, the approximation takes the form of an optimization problem, which can converge much faster than an MCMC sampler with modern optimization techniques. Using VI requires specifying a family of variational distributions $q_{\boldsymbol{\phi}}$ parameterized by variational parameters $\boldsymbol{\phi}$. Optimization is then used to vary $\boldsymbol{\phi}$ until some notion of distance between the variational distribution and the posterior is minimized. A common choice is the Kullback-Leibler (KL) divergence, defined as:

$$\mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{\theta}), p(\boldsymbol{\theta}|\mathcal{D})) = \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log\left(\frac{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right) = -L(\boldsymbol{\phi}) + \log p(\mathcal{D}), \tag{3.2}$$

$$\text{where } L(\boldsymbol{\phi}) := \int_{\boldsymbol{\theta}} q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) \log\left(\frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\boldsymbol{\theta})}\right). \tag{3.3}$$

The KL divergence is always nonnegative, so the log evidence $\log p(\mathcal{D})$ must always be greater than or equal to $L(\boldsymbol{\phi})$. For this reason, $L(\boldsymbol{\phi})$ is often referred to as the evidence lower bound (ELBO). Moreover, since the evidence is independent of the variational parameters, minimizing the KL divergence is equivalent to maximizing the ELBO. Therefore, the optimal variational distribution $q_{\boldsymbol{\phi}^*}$ is found by solving $\boldsymbol{\phi}^* = \max_{\boldsymbol{\phi}} L(\boldsymbol{\phi})$.

The choice of variational distribution plays an important role in how effective a given VI algorithm is. The family of distributions should be broad enough such that it can properly approximate the target distribution, but it should not be so broad that optimization becomes cumbersome. A reliable approach to choosing the variational distribution is to start simple and slowly add complexity until the desired approximation capabilities are achieved. One of the most popular methods to simplify the variational distribution is to use the mean-field approximation [40, 16, 6], in which each element of the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$ is assumed to be independent. Under this assumption, the variational distribution can be decomposed as $q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \prod_{i=1}^{p} q_{\boldsymbol{\phi}}^{i}(\theta_i)$. Here, we use Gaussian marginal variational distributions $q_{\boldsymbol{\phi}}^{i}$ [5]. This leads to the variational distribution

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = \prod_{i=1}^{p} \frac{1}{\sigma_i(\boldsymbol{\phi})\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\theta_i - \mu_i(\boldsymbol{\phi})}{\sigma_i(\boldsymbol{\phi})}\right)^2\right). \tag{3.4}$$

In Section 5, the mean-field Gaussian (3.4) and multivariate Gaussian (3.5) variational distributions

$$q_{\boldsymbol{\phi}}(\boldsymbol{\theta}) = |2\pi\boldsymbol{\Sigma}(\boldsymbol{\phi})|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}(\boldsymbol{\phi}))^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}(\boldsymbol{\phi}))\right). \tag{3.5}$$

will be compared for approximating the posterior distribution of FTT parameters using the Pyro software package [4, 34].

4. Dynamical system estimation. Now, we will investigate using tensor-trains to learn models of dynamical systems. First, consider a system with continuous-time dynamics and discrete-time outputs

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(\mathbf{x}), \qquad \mathbf{y}_k = \mathbf{x}_k, \tag{4.1}$$

where the function $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ maps the state $\mathbf{x} \in \mathbb{R}^d$ to its time derivative, and the observations $\mathbf{y}_k$ are taken at a constant time interval denoted Δt. The goal of this work is to learn a model of f that can predict the system outputs at times in the future. In order to estimate the system outputs from an approximation of f, a time-stepping scheme is required. Without loss of generality, we employ explicit forward Euler integration [1] such that

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \hat{f}(\mathbf{z}_k, \boldsymbol{\theta})\Delta t, \tag{4.2a}$$

$$\mathbf{y}_k = \mathbf{z}_k + \boldsymbol{\eta}_k, \quad \boldsymbol{\eta}_k \sim \mathcal{N}(0, \sigma^2), \tag{4.2b}$$

where $\hat{f} : \mathbb{R}^d \times \mathbb{R}^p \mapsto \mathbb{R}^d$ is an approximation of the function f parameterized by unknown parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, $\mathbf{z}_k \in \mathbb{R}^d$ represents the state $\mathbf{x}_k$ estimated by the dynamics $\hat{f}$, and the term $\boldsymbol{\eta}_k$ represents the measurement noise and is assumed to be independent and identically distributed. Since the variance of the noise is typically unknown, σ is added to the estimation problem as a hyperparameter. Note that approximation error is not accounted for in this modelling framework. However, we select this output model (4.2b) because it is standard in the system identification literature [39] and provides a simple posterior that can be used for preliminary testing of the FTT approximation. Future work will build complexity by applying estimation approaches that account for model error, such as the one described in [18].

The posterior distribution $p(\boldsymbol{\theta}, \sigma|\mathbf{Y})$ representing the uncertainty in the parameters given a set of training data (Section 3) satisfies

$$p(\boldsymbol{\theta}, \sigma|\mathbf{Y}) \propto p(\mathbf{Y}|\boldsymbol{\theta}, \sigma)p(\boldsymbol{\theta}, \sigma), \quad \text{with} \tag{4.3}$$

$$p(\mathbf{Y}|\boldsymbol{\theta}, \sigma) = \prod_{k=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\mathbf{y}_k - \mathbf{z}_k(\boldsymbol{\theta})}{\sigma}\right)^2\right), \tag{4.4}$$

where $\mathbf{z}_k(\boldsymbol{\theta})$ is written as a function of $\boldsymbol{\theta}$ to emphasize its dependence on the approximated dynamics $\hat{f}$ that are parameterized by $\boldsymbol{\theta}$. The form of the prior distribution is problem-dependent and left to the user. In this report, the parameters $\boldsymbol{\theta}$ and σ are always assumed to be independent such that $p(\boldsymbol{\theta}, \sigma) = p(\boldsymbol{\theta})p(\sigma)$. This independence assumption is used for simplicity, but it is not restrictive. Correlations between $\boldsymbol{\theta}$ and σ can still be introduced into the posterior through the contribution of the likelihood. Moreover, since we use weakly informative priors throughout this work, the contribution of the prior is minimal.

5. Results. In this section, the methodology of combining FTTs with VI is applied to two test problems, and the method's utility for function approximation and uncertainty quantification is evaluated. The first example uses a time-independent test function to highlight the challenges of using FTTs for function approximation. The second example considers a dynamical model known as the Lorenz '63 system to demonstrate the feasibility of pairing FTTs with the approach of Section 4 for the estimation of dynamical systems.

5.1. Friedman function. The Friedman function [17] is a five-dimensional test problem for function approximation and is defined as

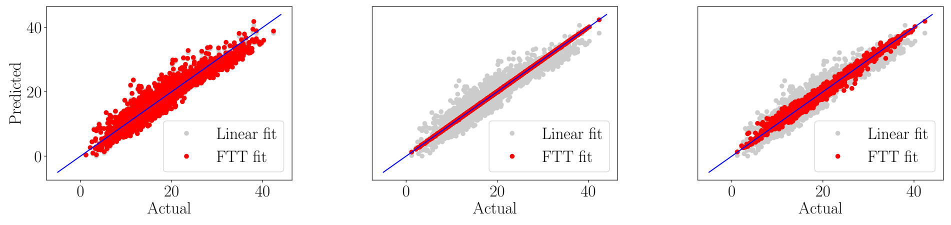
$$y = 10\sin\left(\frac{\pi}{4}(x_1+1)(x_2+1)\right) + 5(x_3-1)^2 + 5(x_4+1) + \frac{5}{2}(x_5+1). \tag{5.1}$$

In the following we construct and test FTT approximation, with varying FTT ranks and Legendre polynomials of varying degree for each FTT core, using 7,000 model evaluations of this function. The inputs are drawn uniformly at random from the interval [-1, 1], i.e. $\mathbf{x}^{(i)} \sim \mathcal{U}[-1,1]^5$ for $i = 1, \ldots, 7000$. Then, each input is pushed through the Friedman function to generate its corresponding output $y^{(i)}$. We partition the dataset into $N = 5,000$ pairs $(\mathbf{x}^{(i)}, y^{(i)})$ for training and 2,000 pairs for testing. The univariate functions inside each FTT core are Legendre polynomials. We consider several polynomial orders and FTT ranks for this example problem.

5.1.1. Deterministic learning. In this section, we learn the coefficients of the FTT using deterministic optimization. Specifically, assuming uniform priors $p(\theta)$, we find the maximum *a posteriori* (MAP) point of the posterior distribution by minimizing

$$\mathcal{J}(\boldsymbol{\theta}) = \sum_{i=1}^{N}\left(y^{(i)} - \hat{f}(\mathbf{x}^{(i)}, \boldsymbol{\theta})\right)^2, \tag{5.2}$$

where $\hat{f}$ is an FTT parameterized by $\boldsymbol{\theta}$. The nonlinear interactions of the parameters during the evaluation of the FTT make this cost function's surface non-convex and difficult to optimize, so a good initial point for optimization is critical. To address this issue, Ref. [20] initialized optimization using a linear fit of the data represented in FTT format. For this experiment, we extended this idea into a hierarchical approach. A low order/rank FTT was trained first, and the optimization result was used as a starting point for training an FTT with larger order/rank. Specifically, an FTT with polynomial order 1 and rank 2 was found



(a) Order 1; Rank 2 (b) Order 4; Rank 3 (c) Order 6; Rank 5

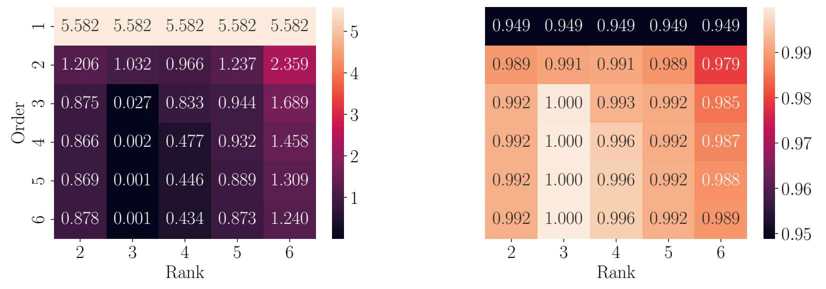
Fig. 5.1: Parity plots at different order-rank pairs over the testing data. The blue line represents an exact fit and is overlaid for reference.

using linear least squares. Then, the rank was incremented and the new FTT was trained using non-linear optimization starting from the initial point given by the optimum of the previous learned FTT. This procedure was continued for order 1 up to rank 6. At this point, the order was incremented and the FTT of the same rank but lower order was used as a starting point. This procedure was continued up to order 6. Note that the choice of incrementing the rank first and the order second was an arbitrary one. Based on the authors' experience, incrementing the order first and the rank second does not produce statistically different results.

After completing the aforementioned training procedure, three order/rank pairs were selected and parity plots depicted in Figure 5.1 were created using testing data. The three order/rank pairs were chosen to be the smallest FTT with order 1 and rank 2, an intermediate-sized FTT with order 4 and rank 3, and a large FTT with order 6 and rank 5. In Figure 5.1a, the FTT fit is not much better than the linear fit since the order/rank is so small. The intermediate-sized FTT in Figure 5.1b fits the data well with all of the markers lying very close to the blue line that represents an exact fit. The large FTT in Figure 5.1c, however, shows worse performance than the medium FTT despite being more expressive. A possible explanation for this behavior is that the greater expressiveness leads to overfitting the training data.

The mean squared error (MSE) of the FTT estimates on the testing data are presented in a heatmap in Figure 5.2a. As the order increases, the MSE decreases at all ranks except for rank 2, which begins to overfit around order 5. As the rank increases, the MSE decreases at first, but after rank 3 it begins to rise. This rise in MSE across the ranks is possibly also due to overfitting, but for a fixed rank, the MSE actually decreases as order increases. While increasing order and rank both increase the number of unknowns that must be learned, the number of unknowns only increases linearly with order but quadratically with rank. Thus, increases in rank can lead to overfitting more easily. Figure 5.2b shows the Pearson correlation coefficients between the predicted and actual testing values. This figure shows similar trends to the MSE heatmap. An interesting future experiment would be to observe how these heatmaps change if the previous rank is used as a starting point rather than the previous order.

5.1.2. Sampling. This section compares the performance of MCMC and VI for sampling the parameter posteriors of FTTs. Since the authors' past experience has shown that sampling from FTTs can be quite challenging, a simpler, two-dimensional test function was considered before attempting the five-dimensional Friedman function. Specifically, we



(a) Testing MSE (b) Pearson correlation coefficient

Fig. 5.2: Figure 5.2a shows the mean squared error between predicted and actual values, and Figure 5.2b shows the correlation between predicted and actual values.

learned an order-1 and rank-2 FTT approximation of

$$f(\mathbf{x}) = \sin\left(\frac{\pi}{4}(x_1+1)(x_2+1)\right), \tag{5.3a}$$

$$y = f(\mathbf{x}) + \xi, \quad \xi \sim \mathcal{N}(0,4), \tag{5.3b}$$

which we henceforth refer to as the 2D modified Friedman function. This FTT structure contains eight learnable parameters. Additive Gaussian noise was added to the output data, which has a smoothing effect on the posterior that makes sampling easier. The prior on the parameters was specified as follows

$$p(\boldsymbol{\theta}, \sigma) = \begin{cases} \frac{1}{2\pi}\exp\left(-\frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{100}\right)\frac{1}{\pi}\exp\left(-\frac{\sigma^2}{2}\right), & \text{if } \sigma > 0; \\ 0, & \text{otherwise,} \end{cases} \tag{5.4}$$

where the distribution over $\boldsymbol{\theta}$ has a wide variance of 10 to reflect the fact that we do not have any prior information on the FTT parameters. Gaussian variational distributions, with and without the mean-field approximation, were used to draw 10^4 samples. These samples were then compared with 10^4 samples drawn using an MCMC algorithm known as delayed rejection adaptive Metropolis (DRAM) [22].[1]

To illustrate the complexity of the posterior distribution, the 1D and 2D marginal distributions of the parameters in the first FTT core are plotted in Figure 5.3. The blue represents the MCMC samples, the orange the mean-field Gaussian VI samples, and the green the multivariate Gaussian VI samples. In this example, the two VI algorithms appear to have converged to the same mean values, and the multivariate distribution has slightly larger variance. This figure shows how the posterior of the FTT parameters possesses complex correlations that make MCMC sampling and VI difficult. The VI algorithms appear to converge to a local approximation of the posterior.

Note that the 2D modified Friedman function is bounded between -1 and 1, but the noise used in this example had a variance of 4. Although the noise is unrealistically large compared to the signal, it was found that this larger noise value allowed MCMC sampling

[1]MCMC was actually used to generate 10^6 samples but 10^5 were discarded as burn-in, and the remaining samples 10^4 were drawn at regular intervals to avoid using highly correlated MCMC samples.

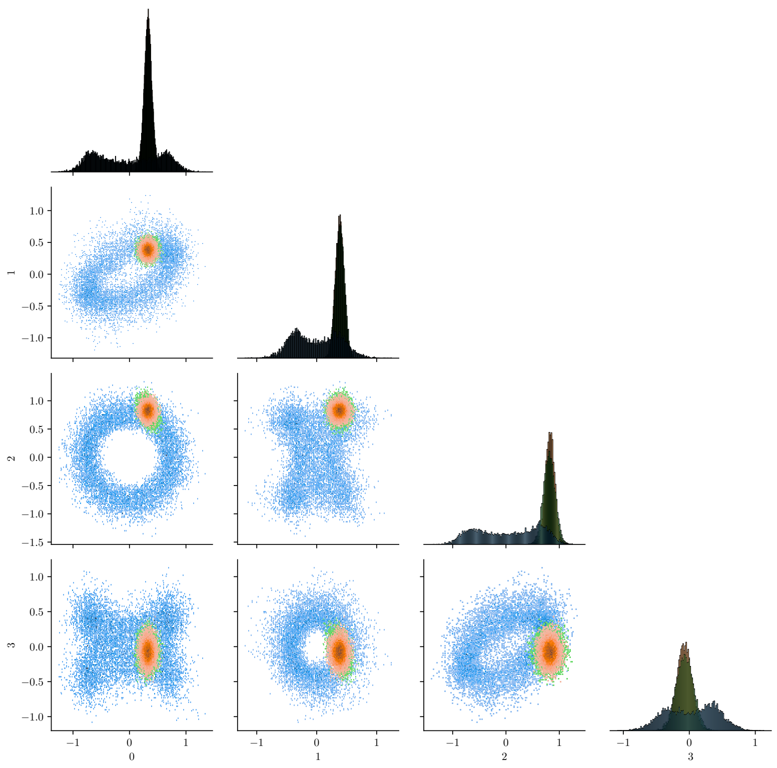
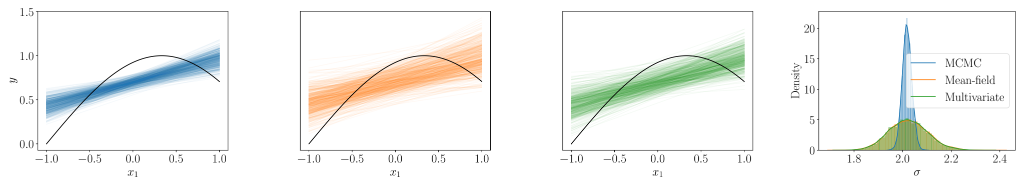


Fig. 5.3: 1D and 2D marginals of the parameters in the 1st FTT core. Blue are MCMC samples, orange are mean-field VI samples, and green are multivariate VI samples. The mean-field and multivariate VI samples overlap in this case.

to converge in a timely manner, whereas for lower noise values, the sampler tended to get stuck in a local region of the posterior.

Clearly, the VI algorithms considered here cannot properly capture the posterior distribution, but in many applications, the uncertainty in the output predictions is of greater interest than that in the parameters. Figure 5.4 shows 500 samples from each algorithm pushed through the FTT function. This figure also gives the marginal distribution of σ, which represents the estimated distribution of errors of each algorithm. In the push-forward plots, x_2 was fixed at 0.5 and x_1 was allowed to vary. It is observed that the MCMC samples give the smallest spread in the output, despite having the greatest variance in parameter space. This is a result of the approximation errors of the variational Gaussian distributions. The VI approximations overestimate the probability density in certain regions of the parameter posterior, which causes there to be a greater proportion of samples from low-probability regions. Since the output variance depends, in part, on the variation of the probabilities of the parameters, we therefore observe greater variance in the output from the VI approximations. Also note that the output distributions produced by the two VI algorithms have roughly the same spread, but the multivariate Gaussian output variance is slightly smaller for this value of x_2. This suggests that the greater flexibility offered by the multivariate Gaussian distribution allows for a better local approximation of the posterior.



(a) MCMC (b) Mean-field (c) Multivariate (d) σ marginals

Fig. 5.4: Push-forward of samples and estimated marginal distribution of σ from each inference algorithm. The black line represents the truth.

The σ marginals in Figure 5.4d show that the mean of the output errors is roughly the same across the three algorithms. The variance of σ, however, is much smaller when using MCMC, and approximately equal between the two VI algorithms in this example. These trends are similar to what was observed in the output plots, suggesting that the σ marginals can be used as a quick way of assessing the uncertainty in the output rather than pushing samples through a, possibly expensive, forward model.

The same inference approach and sampling procedure used on the simplified function was applied to the full, five-dimensional Friedman function with only a few small changes. The FTT is still order 1 and rank 2, but now has five cores instead of two. For this experiment, no noise was added to the output data, and the prior for $\sigma > 0$ was

$$p(\boldsymbol{\theta}, \sigma) = \frac{1}{2\pi} \exp\left(-5\boldsymbol{\theta}^T\boldsymbol{\theta}\right) \frac{1}{\pi} \exp\left(-\frac{\sigma^2}{2}\right), \tag{5.5}$$

and 0 otherwise. The 1D and 2D marginals of the parameters in the first FTT core are shown in Figure 5.5. It is apparent that the MCMC sampling did not converge as evidenced by the absence of the manifolds seen in the previous example. This figure serves to show the performance of sampling from FFTs when applied to a more realistic dataset. Even with one million samples, the MCMC sampler does not appear close to convergence, preventing any meaningful assessment of the VI approximation when estimating the FTT parameters.

The push-forwards of each algorithm are presented in Figure 5.6. Here, the fixed dimensions of $\mathbf{x}$ are set to random values between -1 and 1. Similar to the simplified example, MCMC shows the smallest output variance. This time, the variance reduction from using the multivariate variational distribution over the mean-field one is more significant. An interesting direction for future research would be investigating the validity of the quantification of output uncertainty from sampling only a local region of a posterior when the model parameters are non-unique. In such a case, samples covering the posterior in the output space could be generated much more quickly than samples covering the posterior parameter space. Another direction could be selecting a prior in such way that the symmetries in the posterior are broken, allowing for more efficient sampling [11].

5.2. Lorenz '63. In this example, the FTT framework is used to approximate a dynamical system. Consider the Lorenz '63 system [29] defined as

$$\dot{x}_1 = \sigma(x_2 - x_1), \tag{5.6a}$$
$$\dot{x}_2 = x_1(\rho - x_3) - x_2, \tag{5.6b}$$
$$\dot{x}_3 = x_1 x_2 - \beta x_3, \tag{5.6c}$$

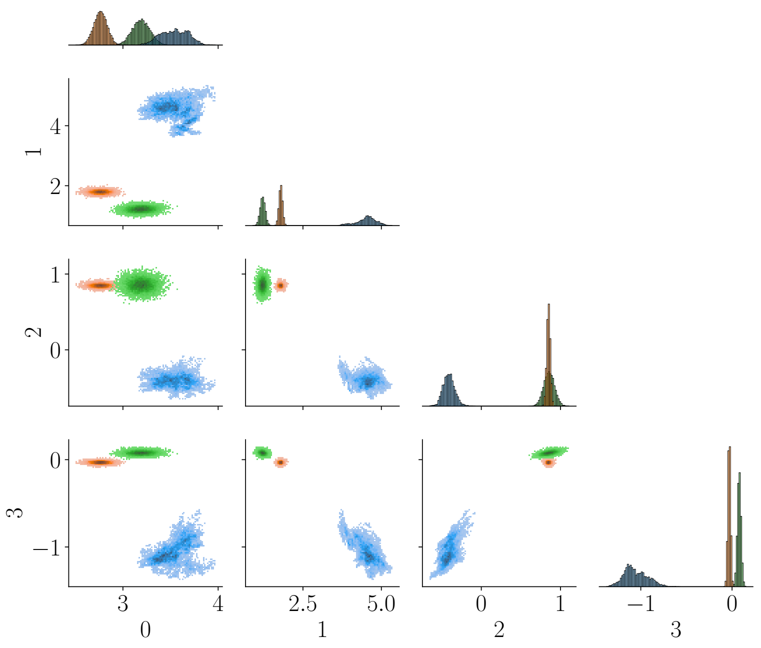
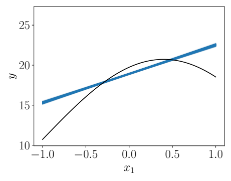
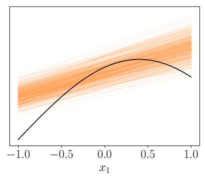


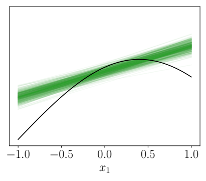
Fig. 5.5: 1D and 2D marginals of the parameters in the 1st FTT core.



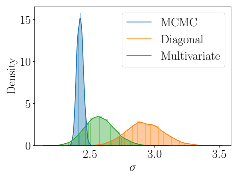
(a) MCMC



(b) Mean-field



(c) Multivariate



(d) σ marginals

Fig. 5.6: Push-forward of samples and estimated marginal distribution of σ from each inference algorithm. The black line represents the truth.

where a common choice of parameters of $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$ is used. The Lorenz system is an example of a deterministic system that exhibits chaotic behavior and can pose a challenge for system identification techniques since even small errors will grow exponentially with time.

The time derivatives of the Lorenz system can be represented as a set of FTT models as

$$\dot{x}_1 = \begin{bmatrix} -\sigma x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \sigma x_2 \end{bmatrix}, \tag{5.7a}$$

$$\dot{x}_2 = \begin{bmatrix} \rho x_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1/\rho \\ -x_2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -x_3 \end{bmatrix}, \tag{5.7b}$$

$$\dot{x}_3 = \begin{bmatrix} x_1 & 1 \end{bmatrix} \begin{bmatrix} x_2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -\beta x_3 \end{bmatrix}. \tag{5.7c}$$

We use this formulation as our model of the dynamics and attempt to learn the unknown parameters $\boldsymbol{\theta} := \begin{bmatrix} \sigma & \rho & \beta \end{bmatrix}^T$ using the methodology described in Section 4. This approach does not technically use the FTT structure, but we parameterize the model in this way to improve identifiability of the model. Furthermore, writing the system in an FTT format allows us to begin introducing other parameters into the inference problem in the future, slowly building complexity.

To generate the data, the system is first simulated for 50s from a random initial point to allow the solution to reach the attractor. The final point of this simulation is then used as the initial condition for numerical integration using forward Euler for 2.0s with $\Delta t = 0.02$. The state is measured at each timestep of this integration procedure without added noise, i.e. $\mathbf{y}_k = \mathbf{x}_k$ for $k = 1, \ldots, 100$.

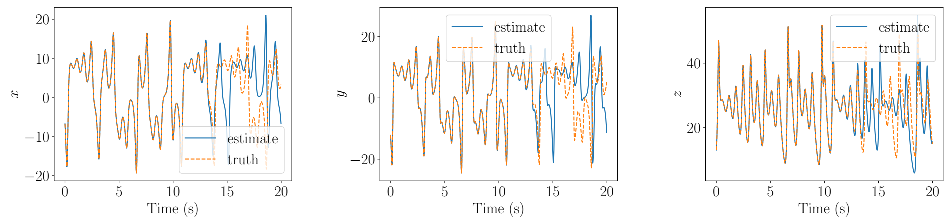
The inference problem is formulated according to the methodology described in Section 4, and a timestep of $\Delta t = 0.02$ is used within the learning procedure. Using the same integration scheme for both data generation and learning simplifies the model training by removing discrepancies in integration between the data and the model prediction. Without this adjustment, there would be estimation error regardless of the parameter values. It is expected that this discrepancy grows rapidly due to the system's chaotic nature. We further mitigate this issue by using a short training period of 2.0s, which avoids challenges related to error accumulation over longer time spans. Addressing this issue requires algorithms that can handle model uncertainty, e.g., as in Ref. [18], and is the topic of future work.

We do not assume any prior information on the parameters and consequently place an improper uniform prior, i.e., a distribution with equal probability over the entire real space, on them. Then, the MAP point of the parameters is found through optimization of the negative log posterior. Starting from the same initial point as the training data, the estimated system is simulated for 20s, and the estimated trajectory is compared to a trajectory simulated with the true parameters in Figure 5.7. The discrepancy between the approximate model and the Lorenz model evolution is plotted in Figure 5.8. The inferred model provides a close approximation of the truth for about 10s before the error starts to blow up as a result of system's chaotic nature.

6. Conclusions. In this report, we evaluated a variational inference approach for the purpose of quantifying uncertainties in functional tensor-train approximations. We found that Markov chain Monte Carlo algorithms are challenged by the lack of identifiability in the functional tensor-train structure. However, initial results suggested that Gaussian approximations of the posterior distribution found with variational inference provide a conservative estimate of the uncertainty in the functional tensor-train outputs.

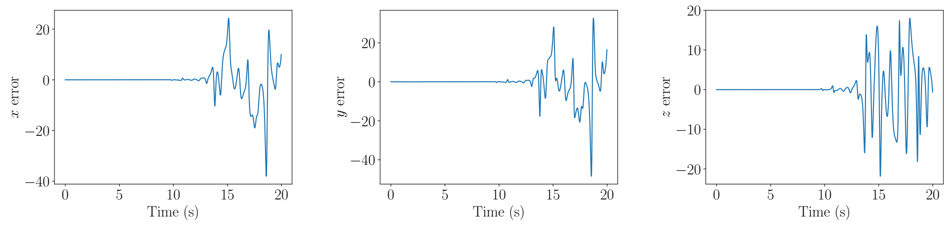
A number of future research directions were identified. These include exploring more expressive variational distribution families, investigating the performance of gradient-based Markov chain Monte Carlo samplers, and exploring the effect of priors on eliminating symmetries in the parameter posterior without sacrificing expressiveness in the output space.

Functional tensor-trains were also applied for estimating the Lorenz '63 system. Given a number of simplifying assumptions, the estimated model performed well, providing reliable



(a) x (b) y (c) z

Fig. 5.7: Trajectories of the estimated and truth systems integrated with forward Euler.



(a) x (b) y (c) z

Fig. 5.8: Residuals between the estimated and truth trajectories of Figure 5.7.

predictions for roughly five times longer compared to the extent of the training period. Future work will begin to remove the simplifying assumptions and also investigate the scalability of the approach to higher dimensional systems.

Acknowledgments. N. Galioto, C. Safta and J. Jakeman were supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute. A. Gorodetsky was supported by the Department of Energy Office of Scientific Research, ASCR under grant DE-SC0020364. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari, *Learning dynamical systems from partial observations*, arXiv preprint arXiv:1902.11136, (2019).

[2] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, *Spectral tensor-train decomposition*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2405–A2439.

[3] I. Bilionis and N. Zabaras, *Multidimensional adaptive relevance vector machines for uncertainty quantification*, SIAM Journal on Scientific Computing, 34 (2012), pp. B881–B908.

[4] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, *Pyro: Deep Universal Probabilistic Programming*, Journal of Machine Learning Research, (2018).

[5] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational inference: A review for statisticians*, Journal of the American statistical Association, 112 (2017), pp. 859–877.

[6] V. Böhm, F. Lanusse, and U. Seljak, *Uncertainty quantification with generative models*, arXiv preprint arXiv:1910.10046, (2019).

[7] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937.

[8] M. Chevreuil, R. Lebrun, A. Nouy, and P. Rai, *A least-squares method for sparse low rank approximation of multivariate functions*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 897–921.

[9] R. R. Coifman and S. Lafon, *Diffusion maps*, Applied and computational harmonic analysis, 21 (2006), pp. 5–30.

[10] P. G. Constantine, *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, SIAM, 2015.

[11] S. De, H. Salehi, and A. Gorodetsky, *Efficient mcmc sampling for bayesian matrix factorization by breaking posterior symmetries*, arXiv preprint arXiv:2006.04295, (2020).

[12] S. Dolgov, K. Anaya-Izquierdo, C. Fox, and R. Scheichl, *Approximation and sampling of multivariate probability distributions in the tensor train decomposition*, Statistics and Computing, 30 (2020), pp. 603–625.

[13] S. Dolgov, D. Kalise, and K. K. Kunisch, *Tensor decomposition methods for high-dimensional hamilton–jacobi–bellman equations*, SIAM Journal on Scientific Computing, 43 (2021), pp. A1625–A1650.

[14] S. V. Dolgov, B. N. Khoromskij, and I. V. Oseledets, *Fast solution of parabolic problems in the tensor train/quantized tensor train format with initial application to the fokker–planck equation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A3016–A3038.

[15] G. Evenbly and G. Vidal, *Tensor network states and geometry*, Journal of Statistical Physics, 145 (2011), pp. 891–918.

[16] S. Farquhar, L. Smith, and Y. Gal, *Try depth instead of weight correlations: Mean field is a less restrictive assumption for variational inference in deep networks*, in Bayesian Deep Learning Workshop at NeurIPS, 2020.

[17] J. H. Friedman, *Multivariate adaptive regression splines*, The Annals of Statistics, 19 (1991), pp. 1–67.

[18] N. Galioto and A. A. Gorodetsky, *Bayesian system id: optimal management of parameter, model, and measurement uncertainty*, Nonlinear Dynamics, 102 (2020), pp. 241–267.

[19] A. Gorodetsky, S. Karaman, and Y. Marzouk, *A continuous analogue of the tensor-train decomposition*, Computer methods in applied mechanics and engineering, 347 (2019), pp. 59–84.

[20] A. A. Gorodetsky and J. D. Jakeman, *Gradient-based optimization for regression in the functional tensor-train format*, Journal of Computational Physics, 374 (2018), pp. 1219–1238.

[21] A. A. Gorodetsky, C. Safta, and J. D. Jakeman, *Reverse-mode differentiation in arbitrary tensor network format: with application to supervised learning*, Journal of Machine Learning Research, 23 (2022), pp. 1–29.

[22] H. Haario, M. Laine, A. Mira, and E. Saksman, *Dram: efficient adaptive mcmc*, Statistics and computing, 16 (2006), pp. 339–354.

[23] H. Harbrecht, J. Jakeman, and P. Zaspel, *Cholesky-based experimental design for Gaussian process and kernel-based emulation and calibration*, Communications in Computational Physics, 29 (2021), pp. 1152–1185.

[24] M. D. Hoffman and A. Gelman, *The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo*, Journal of Machine Learning Research, 15 (2014), pp. 1593–1623.

[25] H. Hoffmann, *Kernel pca for novelty detection*, Pattern recognition, 40 (2007), pp. 863–874.

[26] J. D. Jakeman, F. Franzelin, A. Narayan, M. Eldred, and D. Plfüger, *Polynomial chaos expansions for dependent random variables*, Computer Methods in Applied Mechanics and Engineering, 351 (2019), pp. 643 – 666.

[27] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM Journal on Scientific Computing, 31 (2009), pp. 3002–3026.

[28] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM review, 51 (2009), pp. 455–500.

[29] E. N. Lorenz, *Deterministic nonperiodic flow*, Journal of atmospheric sciences, 20 (1963), pp. 130–141.

[30] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, *Kernel pca and de-noising in feature spaces*, Advances in neural information processing systems, 11 (1998).

[31] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis, *Diffusion maps, spectral clustering*

and reaction coordinates of dynamical systems, Applied and Computational Harmonic Analysis, 21 (2006), pp. 113–127.

[32] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.

[33] ———, *Constructive representation of functions in low-rank tensor formats*, Constructive Approximation, 37 (2013), pp. 1–18.

[34] D. Phan, N. Pradhan, and M. Jankowiak, *Composable effects for flexible and accelerated probabilistic programming in numpyro*, arXiv preprint arXiv:1912.11554, (2019).

[35] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2005.

[36] G. T. Reddy, M. P. K. Reddy, K. Lakshmanna, R. Kaluri, D. S. Rajput, G. Srivastava, and T. Baker, *Analysis of dimensionality reduction techniques on big data*, IEEE Access, 8 (2020), pp. 54776–54788.

[37] L. Richter, L. Sallandt, and N. Nüsken, *Solving high-dimensional parabolic pdes using the tensor train format*, in International Conference on Machine Learning, PMLR, 2021, pp. 8998–9009.

[38] P. Schmid, *Dynamic mode decomposition of numerical and experimental data*, Journal of Fluid Mechanics, 656 (2010), p. 5–28.

[39] J. Schoukens and L. Ljung, *Nonlinear system identification: A user-oriented road map*, IEEE Control Systems Magazine, 39 (2019), pp. 28–99.

[40] J. Swiatkowski, K. Roth, B. Veeling, L. Tran, J. Dillon, J. Snoek, S. Mandt, T. Salimans, R. Jenatton, and S. Nowozin, *The k-tied normal distribution: A compact parameterization of gaussian mean field posteriors in bayesian neural networks*, in International Conference on Machine Learning, PMLR, 2020, pp. 9289–9299.

[41] Y. Wang, H. Yao, and S. Zhao, *Auto-encoder based dimensionality reduction*, Neurocomputing, 184 (2016), pp. 232–242.

[42] S. Wold, K. Esbensen, and P. Geladi, *Principal component analysis*, Chemometrics and intelligent laboratory systems, 2 (1987), pp. 37–52.

[43] D. Xiu and G. Karniadakis, *The Wiener-Askey Polynomial Chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.

THE QUANTUM QUERY COMPLEXITY OF TRIANGLE DETECTION IN THE ADJACENCY LIST MODEL

AMIN GILANI [*], BLAKE HOLMAN [†], JOHN KALLAUGHER [‡], AND OJAS PAREKH [§]

Abstract. The problem of determining whether a graph G on N nodes admits a triangle is one of the most well-studied graph problems in both the classical and quantum settings. Generally, algorithms are allowed to make queries to learn 1) the ith neighbor of a node, 2) the degree of a node, and 3) whether two nodes are connected. In the classical setting, almost all work on triangle-related problems uses all three types of queries. In the quantum setting, however, work has also been done using only queries about the connectivity between nodes. In this work, we take steps toward understanding the quantum query complexity of detecting whether G contains a triangle. In particular, we show that the quantum query complexity determining whether a node participates in a triangle is $\Theta(N)$, and the quantum query complexity of determining whether an edge participates in a triangle is $\Theta(N^{2/3})$.

1. Introduction.

Three nodes u, v, w in a graph $G = (V, E)$ form a triangle if they are all adjacent in G. The problems of detecting, finding, counting, and listing triangles are well studied in a variety of classical settings [4, 5, 7, 10, 14] and have applications in networking, social systems, and bioinformatics [1, 3, 9, 11]. Classical algorithms for triangle-related problems generally use a specific model for accessing the graph, which allows for three types of queries:

1. *neighborhood queries* which take a node v and an index i and outputs the ith neighbor of v,
2. *degree queries* which take a node v and output the degree of v denoted d_v, and
3. *pair or edge queries* which take two nodes u and v and output whether they are connected.

The access model associated with only neighborhood and degree queries is called the *(adjacency) list model*, while the access model with only pair queries is called the *(adjacency) matrix model*. We call the query model with access to all three types of queries the *augmented list model*. The augmented list model is the most common for triangle problems. In the augmented list model Eden, Levi, Ron, and Seshadhri [5] give an algorithm which achieves the optimal query complexity of $\tilde{O}\left(\frac{N}{T^{1/3}} + \min\{\frac{m^{3/2}}{T}, \sqrt{m}\}\right)$ (here, the tilde hides polylog factors and polynomial dependency on the error parameter ε). The reason for this is that triangle problems are classically intractable in the list and matrix query models. A simple argument shows that even approximate triangle counting in the matrix model requires $\Omega(N^2)$ queries, and in the list model there is no sublinear query algorithm for approximately counting triangles [6].

Quantum algorithms can query the graph oracles on superpositions of inputs. As a result, quantum algorithms for triangle counting in the matrix and list models do have sublinear query complexity. The quantum query complexity of triangle finding in the matrix model is $\tilde{O}(N^{5/4})$, which can be easily modified to be a triangle counting algorithm with the same quantum query complexity. In the augmented list model, the quantum query complexity of triangle counting is $\tilde{O}\left(\frac{\sqrt{N}}{T^{1/6}} + \frac{m^{3/4}}{\sqrt{T}}\right)$. While triangle problems in the matrix and augmented list models have been well studied, prior to this work nothing was known about the quantum query complexity of triangle problems in the list model.

[*]University of Maryland, asgilani@umd.edu
[†]Purdue University, holman14@purdue.edu,
[‡]Sandia National Laboratories, jmkall@sandia.gov
[§]Sandia National Laboratories, odparek@sandia.gov

1.1. Our Contributions. We make progress towards characterizing the quantum query complexity of triangle problems in the list model. There are several challenges in working with the list model. For example, with pair queries it only takes 3 queries to determine whether some nodes u, v, and w form a triangle. However, in the list model, we must search the neighbors of these nodes to determine whether they are all connected. First, we consider the problem of determining whether a node participates in a triangle. We give an algorithm that uses $O(N)$ queries and a matching lower bound.

Theorem 1.1. *In the adjacency list model, the quantum query complexity of determining whether a node in a graph on N nodes participates in a triangle is $\Theta(N)$.*

Next, we consider the problem of determining whether an edge $\{u, v\}$ participates in a triangle. The upper bound is simple, as it reduces to determining whether the neighborhoods of u and v are disjoint, which is well studied in the quantum setting [2]. The lower bound is considerably more difficult than the previous problem. We give a reduction from the *Element Distinctness Problem*, which gives a function f and asks whether f is one-to-one.

Theorem 1.2. *In the adjacency list model, the quantum query complexity of determining whether an edge in a graph on N nodes participates in a triangle is $\Theta(N^{2/3})$.*

2. Preliminaries.

In this section, we give the background and notation on graphs and their quantum query models.

2.1. Graphs and Query Models. Let $G = (V, E)$ be a graph on N nodes and m edges. We always use the convention that $V = \{0, \ldots, N-1\}$. A vertex u is a neighbor of v if u and v are connect in G, and we let $\Gamma(v)$ denote the neighbors of v. Additionally, for any $S \subset V$, we let $\Gamma(S) = \bigcup_{v \in S} \Gamma(v)$. Additionally, we let $\Gamma(v, i)$ to be the ith neighbor of v according to an arbitrary but fixed order for each vertex. The degree of a vertex v is denoted $d_v = |\Gamma(v)|$ and we let $d_{\max} = \max_{v \in V} d_v$.

There are three main query models which allow us to gain information about G in many different ways. In *adjacency matrix model* (or just matrix model), we have access to an oracle which, given nodes u and v, tell us whether or not $\{u, v\} \in E$. In the *adjacency list model* we have access to two oracles: a degree oracle and a neighborhood oracle. The degree oracle takes a node v and returns its degree d_v. The neighborhood oracle takes in a vertex v and an index i, returning the ith (starting from zero) neighbor with respect to an arbitrary but fixed order. If $i \geq d_v$, then the oracle returns a special symbol $\perp$. The last query model is the *augmented adjacency list model* which is simply the union of the matrix and adjacency list model.

2.2. Quantum Computation. A classical bit is either in the state 0 or the state 1. We can represent these states via a vector such that the zero state is $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the one state is $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. A qubit, however can be in any state of the form

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

for $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$. In order to gain information about $|\psi\rangle$, we must *measure* ψ. Upon measuring, $|\psi\rangle$ collapses to $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. We will also use $\langle\psi|$ to represent $|\psi\rangle^\dagger$ and $\langle\psi|\phi\rangle$ to represent $\langle\psi| |\phi\rangle$.

More generally, for a bit-string $s = s_1, \ldots, s_n \in \{0, 1\}^n$, we can represent s as a vector v of length 2^n, where $v_s = 1$ (here we are indexing with respect to the integer corresponding to s) and all other entries are zero. Equivalently,

$$v = |s\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_n\rangle\,,$$

where $\otimes$ denotes the tensor product between two vectors. Intuitively, writing v as a tensor product of single-qubit states allows us to view v as the bit-string it represents. Additionally, we use the following equivalent notation:

$$|a\rangle \otimes |b\rangle = |a\rangle\,|b\rangle = |a, b\rangle = |ab\rangle\,.$$

A quantum circuit acting on n qubits can be represented as a $2^n \times 2^n$ matrix U. Quantum circuits are inherently reversible, meaning that after applying U to some state $|\phi\rangle$, there exists a quantum circuit U^{-1} which undoes the effect of U. Moreover, due to the natural rules of measurement above, it is clear that U must maintain the 2-norm of $|\psi\rangle$. We call such matrices *unitary*. Since U is unitary, it can be shown that $U^\dagger = U^{-1}$.

We make use of the following quantum subroutine. Given quantum query access to a function $f : [N] \to \{0, 1\}$, Grover's search algorithm [8] can find a marked item x (meaning $f(x) = 1$) or report that none exist using $O(\sqrt{N})$ queries to f (as opposed to classical algorithms which require $\Omega(N)$ queries).

3. Quantum Algorithms for Triangle Problems in the Adjacency List Model. The first problem we consider is determining whether a vertex v participates in a triangle. In the matrix model, this can be done straightforwardly using Grover's search algorithm to find pairs of nodes $u, w \in V$ in which u, v, and w are all adjacent in G, using this uses $O(N)$ queries. In the adjacency list model, however, we cannot simply check whether pairs of vertices are connected in constant time. We give an $O(N)$ algorithm for the problem.

Lemma 3.1. *The problem of determining whether a node participates in a triangle in a graph on N nodes has quantum query complexity $O(N)$.*

Consider the following method:

1. Query the neighborhood oracle to find the neighborhood $u_1, \ldots, u_{d_v}$ of v.
2. Search over the neighbors of u_i to find w which is also a neighbor of v.

The first step costs $O(N)$ queries, as we are classically recording each of v's neighbors. Since we have stored the neighbors of v, it doesn't take any additional queries to determine whether a neighbor w of u_i is a member of $\Gamma(v)$. So, the query complexity of the second step is $O(\sqrt{|\Gamma(\Gamma(v))|}) = O(N)$, resulting in Lemma 3.1.

Next, we consider the query complexity of determining whether an edge $\{u, v\}$ admits a triangle. Again this can be done in the adjacency model by searching for a node w which is connected to both u and v in $O(\sqrt{N})$ queries. We show that in the adjacency model we can determine whether a vertex v forms a triangle with u and v or report that none exists using $O(N^{2/3})$ quantum queries.

Lemma 3.2. *The problem of determining whether an edge participates in a triangle in a graph on N nodes has quantum query complexity $O(N^{2/3})$.*

This problem is similar to the *element distinctness problem.*

Definition 3.3. *Let* ElementDistinctness *be the problem of determining whether a function $f : [N] \to [M]$ with $M = \Theta(N)$ is one-to-one.*

Ambainis gave a quantum algorithm that solves element distinctness using $O\left(N^{2/3}\right)$ queries [2]. In our case, we have an edge $\{u, v\}$ and a list consisting of the entries (with duplicates) of $\Gamma(u)$ and $\Gamma(v)$ of size at most $2d_{\max}$. The proof of Lemma 3.2 follows by applying the element distinctness algorithm to determine whether the edge is a triangle edge.

4. Lower bounds in the Adjacency List Model. In this section, we give lower bounds for triangle-related problems in the adjacency list model.

4.1. Triangle Vertex Determination. First, we consider the problem of deciding whether a vertex participates in a triangle. We give a reduction from the problem of deciding whether a bitstring of length M contains a one, which is known to require $\Omega(\sqrt{M})$ quantum queries. In the previous section, we gave an algorithm that tests whether a node is a triangle node using $O(N)$ queries.

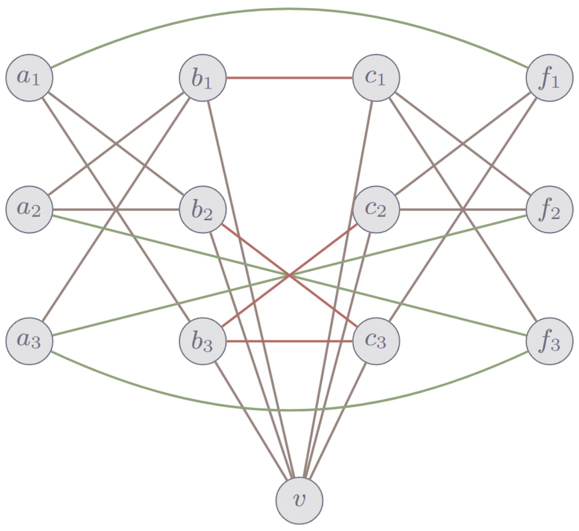


Fig. 4.1. *Above is the graph associated with the string $x' = 10011$ from Lemma 4.1. The associated matrix is* $X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$. *Green edges between a_i and f_j and red edges between b_i and c_j correspond to the case where $X_{ij} = 1$. The red edges create the triangle $b_i c_j v$.*

Lemma 4.1. *In the adjacency list model, determining whether a node v in a graph G is part of a triangle requires $\Omega(N)$ queries. .*

Proof. Given some $x' \in \{0,1\}^M$, let N be the smallest integer such that $\binom{N}{2} \geq M$ and let $x = x' \| 0^{\binom{N}{2}-M}$, where $\|$ denotes string concatenation. Now define the symmetric $N \times N$ matrix X which encodes two copies of x in the following way. First, we construct the upper triangular matrix X' which takes the values x_i in the natural order. We let $X = X' + X'^T - \operatorname{diag}(X')$. Next, we construct a graph $G = (V, E)$ on $4N + 1$ nodes. Let $A = \{a_1, \ldots, a_N\}$, $B = \{b_1, \ldots, b_N\}$, $C = \{c_1, \ldots, c_N\}$, and $F = \{f_1, \ldots, f_N\}$. Finally, we let V $= A \cup B \cup C \cup F \cup \{v\}$. For each $u \in B \cup C$, $\{u, v\}$ is an edge in G. Next, if $X_{ij} = 1$ then $\{a_i, f_j\}, \{a_j, f_i\}, \{b_i, c_j\}, \{b_j, c_i\} \in E$. If $X_{ij} = 0$, then $\{a_i, b_j\}, \{a_j, b_i\}, \{c_i, f_j\}, \{c_j, f_i\} \in E$.

Then we can define the jth neighbor of a_i to be f_j if $X_{ij} = 1$ and b_j otherwise. The jth neighbor of b_i is c_j if $X_{ij} = 1$ and a_j otherwise. The jth neighbors of c_i and f_i are defined analogously. Since each node in $A \cup B \cup C \cup F$ has degree N, we do not have to query x to simulate the degree oracle. Lastly, each query to the neighborhood can be answered with one query to x.

Now, v, b_i, and c_j form a triangle if and only if $X_{ij} = 1$. So, given an algorithm for the triangle vertex problem which uses at most q quantum queries, then we can find a 1 in x

using $O(\sqrt{q})$ queries. Since we know that the problem of deciding whether x contains a 1 has query complexity $\Omega(\sqrt{N})$, we know that $q = \Omega\left(\sqrt{\binom{N}{2}}\right) = \Omega(N)$. □

Theorem 1.1 follows from Lemmas 3.1 and 4.1.

4.2. Triangle Edge Determination. In this section we show that in the adjacency list model, the problem of determining whether an edge is a triangle edge has quantum query complexity $\Omega(N^{2/3}/\log N)$, completing the proof of Theorem 1.2. To accomplish this, we show that the triangle edge problem requires just as many queries (asymptotically) as the *element distinctness problem*, which asks whether a function $f : [N] \to [M]$ is one-to-one (i.e. has no collision). Prior work showed that the promise problem in which $M = N$ and there is at most one collision in f also has quantum query complexity $\Omega(N^{2/3})$ [13]. We will let UElementDistinctness denote this special case of element distinctness.

The reduction from UElementDistinctness to the problem of determining whether an edge is a triangle edge relies on the following ideas. Suppose we have disjoint sets $W = \{w_1, \dots, w_{\ell/2}\}$ and $Z = \{z_1, \dots, z_{\ell/2}\}$ which are both subsets of the domain of our function f. Suppose further that between W and Z there is either a single collision pair (and no more) w_i and z_j such that $f(w_i) = f(z_j)$ and $i \neq j$ or there are none. Now we want to set up a graph with an edge $\{u, v\}$ such that $\{u, v\}$ participates in a triangle if and only if some w_i and z_j form a collision. A natural approach is to have a set of N vertices $L_1, \dots, L_N$ with an edge from u to L_i whenever $f(w_j) = i$ and an edge from v to L_i whenever $f(z_j) = i$. At a glance, this seems to work, as the resulting graph G has a triangle whenever W and Z have a collision. The problem, however, is that determining the degree of L_i is essentially the same as checking whether a list contains a certain element, meaning we cannot simulate it with a constant number of queries to f. To circumvent this problem, we add an additional N nodes $R_1, \dots, R_N$ and connect L_i to R_j whenever $f(w_j) \neq i$ and $f(z_j) \neq i$. Now we can straightforwardly implement the neighborhood and degree oracles with only a constant number of queries to f.

To increase our success probability, we split up our input $X = \langle x_1, \dots, x_N \rangle$ into more than two sublists, each of size $\ell = \Theta(N)$ so that the probability that a collision pair is in the same sublist is small. We can now construct the graph mentioned above on every pair of these sublists to determine whether f has a collision with probability $\gg 1/2$.

Lemma 4.2. *The problem of determining whether an edge of a graph on N nodes is a triangle edge requires $\Omega\left(N^{2/3}\right)$ queries.*

Proof. Let $f : [N] \to [N]$ be an instance of UElementDistinctness. Let $\mathcal{A}$ be an $A(N)$-query algorithm for triangle edge detection. We construct the $B(N)$-query algorithm $\mathcal{B}$, which does the following on input f. Let π be a random permutation of $[N]$, and let $F = \langle f(\pi(1)), \dots f(\pi(N)) \rangle$. Now we divide F into N/ℓ sublists $F^i = \langle F_{\ell(i-1)+1}, \dots, F_{\ell i} \rangle$ for $\ell = \frac{N}{3}$. The probability that no sublist contains a collision pair and any two sublists which share a value share it at different indices is at least $1 - \left(\frac{\ell}{N}\right)^2 \cdot \frac{N}{\ell} - \frac{1}{\ell} = 2/3 - o(1)$. For now, assume this is the case.

The graph $G_{ij} = (V, E)$ is constructed as follows. The vertex set V of size $2N + 2$ consists of two sets of nodes corresponding to the range $L_1, \dots, L_N$ and $R_1, , \dots, R_N$ as well as two helper nodes u and v. The edge set is defined such that

$$\begin{aligned} E = \{&\{u,v\}\} \\ &\cup \{\{u, L_a\} : a \in F^i\} && u \text{ is adjacent to nodes corresponding to } F^i \\ &\cup \{\{v, L_a\} : a \in F^j\} && v \text{ is adjacent to nodes corresponding to } F^j \\ &\cup \{\{L_a, R_b\} : F^i[b], F^j[b] \neq a\} && L_a \text{ is adjacent to } R_b \text{ if } F^i[b] \neq a \text{ and } F^j[b] \neq a. \end{aligned}$$

First, we will show that neighborhood queries can be answered using $O(1)$ queries to f. The bth neighbor of nodes u and v can be answered by simply querying the bth entry of F^i and F^j, respectively. The bth neighbor of L_a is

$$\Gamma(L_a, b) = \begin{cases} R_b & \text{if } F^i[b], F^j[b] \neq a \\ u & \text{if } F^i[b] = a \\ v & \text{if } F^j[b] = a, \end{cases}$$

so any neighbor query can be answered with $O(1)$ queries to the list. Recall that we are assuming that $F^i[b] \neq F^j[b]$ for all b. For the R_a nodes, we let $\Gamma(R_a, b) = L_{b'}$, where $b' = b + \left|\{r \leq i : F^i[a] = r \text{ or } F^j[a] = k\}\right|$, which can be answered by querying $F^i[a]$ and $F^j[a]$. By our assumption $d_{R_a} = N - \left|\{r \in [N] : F^i[a] = r \text{ or } F^j[a] = r\}\right|$. So, answering the degree queries also only takes $O(1)$ queries to f. Nodes u and v have degree ℓ and each L_a has degree N. Thus, the probability $\mathcal{B}$ successfully find a collision is at least $2/3 - o(1)$ which can be amplified using majority voting. Furthermore, $B(N) = O(A(N))$, so $A(N) = \Omega(N^{2/3}/\log N)$. ◻

Theorem 1.2 follows from Lemmas 3.2 and 4.2.

5. Conclusion and Open Problems.

In this work, we show that the quantum query complexity of the problem of determining whether a node participates in a triangle is $\Theta(N)$ and for edges is $\Theta(N^{2/3})$. The natural next step is to give nontrivial upper or lower bounds on the quantum query complexity of determining whether a graph has a triangle. A promising avenue may be to use similar techniques as we did to lower bound the edge triangle problem. We can generalize the element distinctness problem to the case where we are given any function $f : [N] \to [M]$ and are asked to determine whether f is one-to-one. In this case, the problem of determining whether at least 1 of N instances of this new problem is one-to-one has query complexity $\Omega(N^{7/6})$ [12]. Likewise, it is still open whether there is a $o(N^{3/2})$ algorithm for determining whether a graph admits a triangle in the adjacency list model.

REFERENCES

[1] M. Al Hasan and V. S. Dave, *Triangle counting in large networks: a review*, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8 (2018), p. e1226.

[2] A. Ambainis, *Quantum walk algorithm for element distinctness*, SIAM Journal on Computing, 37 (2007), pp. 210–239.

[3] B. Andreopoulos, C. Winter, D. Labudde, and M. Schroeder, *Triangle network motifs predict complexes by complementing high-error interactomes with structural information*, BMC bioinformatics, 10 (2009), p. 196.

[4] A. Azad, A. Buluç, and J. Gilbert, *Parallel triangle counting and enumeration using matrix algebra*, in 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IEEE, 2015, pp. 804–811.

[5] T. Eden, A. Levi, D. Ron, and C. Seshadhri, *Approximately counting triangles in sublinear time*, SIAM Journal on Computing, 46 (2017), pp. 1603–1646.

[6] M. Gonen, D. Ron, and Y. Shavitt, *Counting stars and other small subgraphs in sublinear-time*, SIAM Journal on Discrete Mathematics, 25 (2011), pp. 1365–1411.

[7] O. Green, P. Yalamanchili, and L.-M. Munguía, *Fast triangle counting on the gpu*, in Proceedings of the 4th Workshop on Irregular Applications: Architectures and Algorithms, 2014, pp. 1–8.

[8] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 212–219.

[9] P. W. Holland and S. Leinhardt, *A method for detecting structure in sociometric data*, American Journal of Sociology, 76 (1970), pp. 492–513.

[10] R. Pagh and C. E. Tsourakakis, *Colorful triangle counting and a mapreduce implementation*, Information Processing Letters, 112 (2012), pp. 277–281.

[11] A. Portes, *Social capital: Its origins and applications in modern sociology*, Annual Review of Sociology, 24 (1998), pp. 1–24.

[12] B. W. Reichardt, *Reflections for quantum query algorithms*, in Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, USA, 2011, Society for Industrial and Applied Mathematics, p. 560–569.

[13] A. Rosmanis, *Adversary lower bound for element distinctness with small range*, arXiv preprint arXiv:1401.3826, (2014).

[14] T. Schank and D. Wagner, *Finding, counting and listing all triangles in large graphs, an experimental study*, in Experimental and Efficient Algorithms, S. E. Nikoletseas, ed., Berlin, Heidelberg, 2005, Springer Berlin Heidelberg, pp. 606–609.

NUMERICAL EVALUATION OF RANDOM SKETCH GMRES

ANDREW J. HIGGINS*, ERIK G. BOMAN†, JENNIFER A. LOE†, AND ICHITARO YAMAZAKI†

Abstract. Krylov subspace methods (KSMs) are state of the art iterative solvers for large, sparse linear systems of equations of the form $Ax = b$. GMRES is a robust KSM for non-symmetric problems that minimizes the computed solution's residual norm over the generated Krylov subspace. Recent advances in Numerical Linear Algebra suggest randomized methods can boost performance without sacrificing much accuracy, and very recent articles have incorporated randomized techniques into KSMs. This manuscript is dedicated to experimentation and analysis of randomized variants of GMRES, along with a review of the prerequisite randomized linear algebra theory used to design the algorithms.

The randomization technique analyzed in this manuscript uses a "sketch" matrix to reduce the dimension of the associated GMRES least squares problem while preserving the least squares norm within a specific tolerance. A recently proposed algorithm called "sketched GMRES" (sGMRES) is a variant of GMRES using a partially orthogonalized Krylov basis along with a sketched least squares solution. It is evident that the computational cost per iteration of sGMRES can be far lower than GMRES if an appropriate sketch matrix is used, however, it is unclear whether the method will converge similarly enough to GMRES to ensure that sGMRES is beneficial. Thus, in this manuscript, we compare sGMRES to GMRES from the standpoint of robustness. We show that while the sGMRES method with efficient sketching is more computationally efficient, its robustness is inhibited by the partial orthogonalization. Other strategies for producing the sGMRES Krylov basis that avoid full orthogonalization are explored, but suffer failures similar to those of the traditional sGMRES approach. However, experiments show sGMRES may have potential for some practical applications, including non-symmetric computational fluid dynamics problems.

1. Introduction.

Krylov subspace methods (KSMs) are the state of the art iterative algorithms for solving large, sparse systems of equations of the form $Ax = b$, which are ubiquitous throughout scientific computing [8]. When the coefficient matrix A is non-symmetric, often times the Generalized Minimal Residual Method (GMRES) is used to solve the system, due to its robustness and monotone convergence guaranteed by the residual minimization properties of the algorithm [11]. In spite of its robustness, a major drawback of GMRES is its computational cost due to its long-tail recurrence relation used to create a basis for the Krylov solution space. While alternative KSMs for non-symmetric systems exist that are less computationally expensive per iteration such as BiCGSTAB, the residual is not minimal in each iteration, and therefore convergence is non-monotone, and can sometimes require far more iterations to converge than GMRES [13]. For this reason, alternative variants of GMRES and GMRES-like algorithms are still of great interest in the Numerical Linear Algebra community.

Development and analysis of randomized techniques is cutting edge research in the field of Numerical Linear Algebra. In particular, the integration of randomized techniques into Krylov subspace methods for solving linear systems has only begun in the past couple of years. One popular randomized approach in Numerical Linear Algebra is the concept of random "sketching", which is a dimension reduction technique via a short and wide random matrix S that ensures $\langle x, y\rangle \approx \langle Sx, Sy\rangle$ and therefore $\|x\|_2 \approx \|Sx\|_2$ with high probability [2]. Very recently, two versions of GMRES that leverage this sketching strategy have been proposed [2, 9]. The one we will focus on in this manuscript was proposed by Nakatsukasa and Tropp in a paper that has not yet been published, which they refer to as "sketched GMRES" (sGMRES) [9]. The idea of the sGMRES algorithm is to produce the Krylov basis using a truncated orthogonalization, and then apply a random sketch to reduce the size of the corresponding least squares problem, which is then solved directly via a QR factorization.

*Temple University, andrew.higgins@temple.edu

†Sandia National Laboratories, egboman@sandia.gov, jloe@sandia.gov, iyamaza@sandia.gov

1.1. Random Sketching. Given $A \in \mathbb{C}^{n,n}$, let $s \ll n$, and define the *sketch matrix* $S \in \mathbb{C}^{s,n}$. The sketch matrix S is typically chosen to be a *subspace embedding*, or a linear map to a lower dimensional space that preserves the ℓ_2-inner product of all vectors within the subspace up to a factor of $\sqrt{1 \pm \varepsilon}$ for some $\varepsilon \in (0,1)$ [9, 12]. Such embeddings also preserve ℓ_2-norms in a similar way [2].

Definition 1.1 (ε-subspace embedding). *Given $\varepsilon \in (0,1)$, a sketch matrix $S \in \mathbb{C}^{s,n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{C}^n$ if $\forall x, y \in \mathcal{V}$,*

$$|\langle x, y\rangle - \langle Sx, Sy\rangle| \leq \varepsilon \|x\|_2 \|y\|_2. \tag{1.1}$$

Equation (1.1) provides a straightforward relation between the sketching matrix and the preservation of the ℓ_2-norm.

Corollary 1.2. *If the sketch matrix $S \in \mathbb{C}^{s,n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{C}^n$, then $\forall x \in \mathcal{V}$,*

$$\sqrt{1-\varepsilon}\ \|x\|_2 \leq \|Sx\|_2 \leq \sqrt{1+\varepsilon}\ \|x\|_2 . \tag{1.2}$$

Corollary 1.2 implies that we can also bound the singular values of a matrix V by the sketched matrix SV. This implies that if SV is well conditioned, then so is V.

Corollary 1.3. *If the sketch matrix $S \in \mathbb{C}^{s,n}$ is an* ε-subspace embedding *for the subspace $\mathcal{V} \subset \mathbb{C}^n$, and V is a matrix whose columns form a basis of $\mathcal{V}$, then*

$$(1+\varepsilon)^{-1/2}\ \sigma_{min}(SV) \leq \sigma_{min}(V) \leq \sigma_{max}(V) \leq (1-\varepsilon)^{-1/2}\ \sigma_{max}(SV). \tag{1.3}$$

Proofs for Corollary 1.2 and 1.3 are provided by Balabanov and Grigori in their paper on Randomized Gram-Schmidt [2].

The limitation of ε-subspace embeddings presented in Definition 1.1 is that one needs to know the subspace $\mathcal{V} \subset \mathbb{C}^n$ a priori to ensure the sketch matrix approximately preserves norms and inner products. To use sketched techniques in Krylov subspace methods efficiently, we should instead use a sketch matrix that does not require complete prior knowledge of the subspace, since Krylov subspaces are generated as the algorithm iterates. This can be accomplished by using (ε, δ, d) *oblivious ℓ_2-subspace embeddings* [2].

Definition 1.4 ((ε, δ, d) oblivious ℓ_2-subspace embedding). *A sketch matrix $S \in \mathbb{C}^{s,n}$ is an (ε, δ, d)* oblivious ℓ_2-subspace embedding *if it is an ε-subspace embedding for any fixed d-dimensional subspace $\mathcal{V} \subset \mathbb{C}^n$ with probability at least $1-\delta$.*

There are several examples of (ε, δ, d) oblivious ℓ_2-subspace embeddings. Given a Gaussian matrix $G \in \mathbb{C}^{s,n}$, a concrete example of a (ε, δ, d) oblivious ℓ_2-subspace embedding is $S = \frac{1}{\sqrt{s}}G$ where $s = \Omega(\varepsilon^{-2} \log d \log(1/\delta))$ [12]. Nakatsukasa and Tropp claim that an "optimal scaling" of the embedding length follows the relation $s \approx d/\varepsilon^2$ [9]. Using $\varepsilon = 1/\sqrt{2}$, they describe a simple relation between the embedding dimension (sketch size) s and the subspace dimension d by $s = 2(d+1)$ which is consistent with this "optimal scaling", but in principle, one could choose a different value of ε to construct a different sketch size.

Due to its simplicity in implementation, we conducted our experiments using a Gaussian sketch matrix S of this form, as we were more concerned with robustness and stability of the sketching methods than we were with performance in this manuscript. It should be noted for performance considerations that a dense Gaussian sketch matrix is not a practical choice of a sketch matrix for large-scale problems. Other (ε, δ, d) oblivious ℓ_2-subspace embeddings that can be applied efficiently and can be stored in a sparse format include sub-sampled randomized Hadamard and Fourier transforms (SRHT and SRFT, respectively), and "sparse dimension reduction maps" [2, 9].

1.2. GMRES Algorithm. GMRES is an algorithm designed to solve linear systems of equations of the form $Ax = b$, where $A \in \mathbb{C}^{n,n}$, $b \in \mathbb{C}^n$. By design, GMRES(m) chooses a solution that solves the least squares problem

$$x_m = \underset{x \in x_0 + \mathcal{K}_m(A, r_0)}{\arg\min} \|b - Ax\|_2 , \tag{1.4}$$

where $r_0 = b - Ax_0$, and

$$\mathcal{K}_m(A, r_0) = \operatorname{span}(r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0) \tag{1.5}$$

is the m-dimensional Krylov subspace generated by A and r_0.

In principle, GMRES can use any basis for the Krylov subspace $\mathcal{K}_m(A, r_0)$ to construct its approximate solution. If V_m is a matrix whose columns form a basis of $\mathcal{K}_m(A, r_0)$, then regardless of how the basis is constructed, the GMRES(m) solution is mathematically equivalent to

$$x_m = x_0 + V_m y_m, \qquad y_m = \underset{y \in \mathbb{C}^m}{\arg\min} \|b - AV_m y\|_2 \tag{1.6}$$

in exact arithmetic. In practice, if one is not careful with the construction of the basis of $\mathcal{K}_m(A, r_0)$, then V_m (and consequently the least squares matrix AV_m) will become severely ill-conditioned, leading to a rank-deficient basis in finite precision, thereby causing stagnation of GMRES(m). This is one of the reasons the traditional implementation of the GMRES(m) algorithm uses a full orthogonalization of the Krylov basis vectors via Modified Gram-Schmidt (MGS). By the orthogonality of the Krylov basis vectors V_m, one can show that

$$y_m = \underset{y \in \mathbb{C}^m}{\arg\min} \left\|\beta e_1 - H_{m+1,m} y\right\|_2 \tag{1.7}$$

is equivalent to equation (1.6), where $H_{m+1,m} \in \mathbb{C}^{m+1,m}$ is an upper Hessenberg matrix, $\beta = \|b\|_2$, and e_1 is the first column of the $m \times m$ identity matrix. Solving the Hessenberg least squares problem in equation (1.7) can be done efficiently using Givens rotations [10].

GMRES boasts two key advantages over other Krylov subspace methods. The first being that in exact arithmetic, it is guaranteed to converge to the correct solution in at most n iterations for any system of equations where $A \in \mathbb{C}^{n,n}$ is square and non-singular, and therefore is a more robust method than many other iterative methods for nonsymmetric systems. This fact is a consequence of equation (1.4), the fact that the true solution $A^{-1}b \in x_0 + \mathcal{K}_n(A, r_0)$ (which itself follows from the Caley-Hamilton theorem), and the fact that GMRES with a fully orthogonalized basis is backwards stable. The second key advantage of GMRES is that its convergence is monotone, which also follows from equation (1.4).

The major drawback of the traditional GMRES algorithm based on full orthogonalization is that GMRES uses a long-tail recurrence relation to compute the Krylov basis, and therefore has a relatively high computational complexity per iteration. Since the full orthogonalization of the Krylov basis after m iterations requires $O(nm^2)$ FLOPs in total, it is advantageous to use restarted GMRES with a relatively small restart cycle m to keep the computational cost low.

The drawback to restarting GMRES is that the convergence is dependent on the restart parameter m, and is very difficult to predict how m should be chosen a priori. Although it is counterintuitive, using a larger restart parameter m in restarted GMRES does not necessarily mean the method will converge in fewer iterations [5].

1.3. Newton Krylov Basis. Ideally, one can form a Krylov basis that is better conditioned than the standard Krylov basis $\{b, Ab, A^2b, \ldots, A^{m-1}b\}$ without incurring significantly more computational cost. One strategy to control the conditioning of the Krylov basis, albeit not as well as a full orthogonalization does, is to use a Newton basis [1]. The Newton Krylov basis of $\kappa_m(A, b)$ is defined as

$$\left\{ b,\ (A-\theta_1 I)b,\ (A-\theta_2 I)(A-\theta_1 I)b,\ \ldots\ ,\ \prod_{j=1}^{m}(A-\theta_j I)\ b \right\}, \tag{1.8}$$

where $\theta_1, \theta_2, \ldots, \theta_m$ are the Ritz values from one cycle of GMRES(m) (i.e., the eigenvalues of the Hessenberg matrix $H_{m+1,m}$) [1].

A brief procedure outlining how to form a Newton basis of the Krylov subspace $\mathcal{K}_m(A, r) = \text{span}\{r, Ar, \ldots, A^{m-1}r\}$ is given in Algorithm 1. In order to obtain the Ritz values $\theta_1, \theta_2, \ldots, \theta_m$, one must first perform one cycle of GMRES(m).

Algorithm 1 Newton Krylov Basis

```
  Input:  Coefficient matrix $A \in \mathbb{R}^{n,n}$, cycle length $m$, initial basis vector $r \in \mathbb{R}^n$,
          Ritz values (using Leja ordering) $\theta_1, \theta_2, \ldots, \theta_m$
  Output: Newton Krylov basis $V_m \in \mathbb{R}^{n,m}$
1: $v_1 = r/\|r\|_2$
2: for $j = 1, \ldots, m$ do
3:     $w = Av_j - \theta_j v_j$
4:     $v_{j+1} = w/\|w\|_2$
5: end for
```

Asymptotically, computing the Newton Krylov basis is no more computationally expensive than the standard basis, because the Ritz values are pre-computed once before beginning the algorithm.

1.4. sGMRES Algorithm. Nakatsukasa and Tropp proposed the sGMRES algorithm in 2021, which forms the Krylov basis matrix V_m using a partial orthogonalization, and then solves equation (1.4) using a random sketch [9]. In particular, given a sketch matrix S, their method directly solves the sketched least squares problem

$$y_m = \underset{y \in \mathbb{C}^m}{\arg\min} \|Sr_0 - SAV_m y\|_2 \tag{1.9}$$

via a QR factorization, and then updates the solution $x_m = x_0 + V_m y_m$ [9]. The idea is that if S is a (ε, δ, d) oblivious ℓ_2-subspace embedding, then it follows from equation (1.2) that

$$\sqrt{1-\varepsilon}\ \|r_0 - AV_m y_m\|_2 \leq \|Sr_0 - SAV_m y_m\|_2 \leq \sqrt{1+\varepsilon}\ \|r_0 - AV_m y_m\|_2\,. \tag{1.10}$$

Therefore, when the sketched residual norm $\|Sr_0 - SAV_m y_m\|_2$ is small enough to converge, the true residual norm $\|r_0 - AV_m y_m\|_2$ should also be small enough to converge, and conversely, if the sketched residual norm does not converge, the true residual norm should not converge.

Moreover, one can expect the convergence of sGMRES to be driven by the choice of the basis V_m rather than the sketching. To see this, consider the "unsketched" version of the sGMRES algorithm where we produce the Krylov basis V_m in the same way as sGMRES (e.g., with partial orthogonalization or some other basis construction). Then this "unsketched" sGMRES will have residual norm $\|r_0 - AV_m\tilde{y}_m\|_2$, where

$$\tilde{y}_m = \underset{y \in \mathbb{C}^m}{\arg\min} \|r_0 - AV_m y\|_2 \,. \tag{1.11}$$

Since the sketched least squares solution y_m is not the minimizer of $\|r_0 - AV_m y\|_2$, it follows from equation (1.10) that

$$\sqrt{1-\varepsilon}\ \|r_0 - AV_m\tilde{y}_m\|_2 \le \sqrt{1-\varepsilon}\ \|r_0 - AV_m y_m\|_2 \le \|Sr_0 - SAV_m y_m\|_2 \,. \tag{1.12}$$

Moreover, since $\tilde{y}_m$ is not the minimizer of the sketched least squares norm $\|Sr_0 - SAV_m y\|_2$, it follows that

$$\|Sr_0 - SAV_m y_m\|_2 \le \|Sr_0 - SAV_m\tilde{y}_m\|_2 \le \sqrt{1+\varepsilon}\ \|r_0 - AV_m\tilde{y}_m\|_2 \,. \tag{1.13}$$

Therefore,

$$\sqrt{1-\varepsilon}\ \|r_0 - AV_m\tilde{y}_m\|_2 \le \|Sr_0 - SAV_m y_m\|_2 \le \sqrt{1+\varepsilon}\ \|r_0 - AV_m\tilde{y}_m\|_2 \,, \tag{1.14}$$

which implies the computation of the basis for sGMRES solely determines the convergence of the algorithm. If V_m is severely rank-deficient, then the "unsketched" sGMRES residual norm $\|r_0 - AV_m\tilde{y}_m\|_2$ will stagnate, as new columns of V_m will not add new information to the least squares solution, and by equation (1.14), this implies sGMRES will also stagnate. Conversely, if the basis V_m is chosen so that the "unsketched" sGMRES residual norm $\|r_0 - AV_m\tilde{y}_m\|_2$ converges, then sGMRES will also converge by equation (1.14).

Pseudocode for m iterations of sGMRES with a k-vector partial orthogonalization, denoted 'sGMRES(m)' is shown in Algorithm 2. Nakatsukasa and Tropp show that m iterations of sGMRES incurs only $O(m^3 + nm\log(m))$ FLOPs provided a $k \ll m$ vector partial orthogonalization and a highly efficient SRFT sketch matrix are used, while GMRES incurs $O(nm^2)$ FLOPs, and is therefore more expensive since $n \gg m$ [9]. Their FLOP analysis excludes the sparse matrix-vector multiplications used to produce new Krylov vectors because these computational costs are the same for both the GMRES and sGMRES algorithms, and therefore are not necessary to include when comparing the costs of the two algorithms.

While the partial orthogonalization procedure used in sGMRES leads to computational savings, the resulting Krylov basis V_m should not be expected to maintain full or even almost full rank. Hence, we can expect unrestarted sGMRES to stagnate much earlier than GMRES. To combat this, one can restart sGMRES in an analagous way to GMRES. However, one should keep in mind that restarts are not guaranteed to resolve stagnation issues in KSMs and can drastically change the convergence behavior in unanticipated ways.

Provided the sketch matrix is a (ε, δ, d) oblivious ℓ_2-subspace embedding, the choice of the sketch should not substantially affect convergence, as they will preserve the norms of the least squares solution within a factor of ε and all sGMRES implementations produce the same Krylov subspaces, regardless of the sketch matrix.

Algorithm 2 sGMRES(m) with a k-vector partial orthogonalization

```
Input:  Non-singular matrix $A \in \mathbb{C}^{n,n}$, sketch matrix $S \in \mathbb{C}^{s,n}$,
        RHS vector $b \in \mathbb{C}^n$, initial guess $x_0 \in \mathbb{C}^n$
Output: Solution $x_m \in \mathbb{C}^n$
1:  $r_0 = b - Ax_0$
2:  $\beta = \|r_0\|_2, v_1 = r_0/\beta$
3:  for $j = 1, \ldots, m$ do
4:      $w = Av_j$
5:      for $i = \max(1, j-k+1), \ldots, j$ do
6:          $w = w - (v_i^* w)v_i$
7:      end for
8:      $v_{j+1} = w/\|w\|_2$
9:  end for
10: $y_m = \arg\min_y \|SAV_m y - Sb\|_2$
11: $V_m = [v_1|\ldots|v_m]$
12: $x_m = x_0 + V_m y_m$
```

2. Computational Experiments. Randomized GMRES variants can only be considered promising if they are more robust or computationally efficient than previously studied efficient GMRES implementations and GMRES-like algorithms. While the primary motivation for incorporating randomized techniques into GMRES is driven by parallel efficiency considerations, to understand whether sGMRES is worth implementing into a high-performance software library like Trilinos, we first analyzed the robustness of sGMRES. Thereafter, we attempted to improve the stability of the Krylov basis construction by leveraging a Newton basis. Finally, we consider whether sGMRES–with or without a Newton basis–has value for practical applications.

2.1. Robustness of sGMRES. Nakatsukasa and Tropp present promising results for the sGMRES algorithm in terms of both efficiency and robustness [9]. However, there is a commonality between their examples showing positive results: the problems are nearly symmetric. In particular, they show two examples where GMRES and sGMRES converge at about the same rate: the first using the `t2em` matrix from the SuiteSparse Library [4], and another using a 2D Laplacian matrix. The `t2em` matrix has 99.9% pattern and numeric symmetry, and Laplacian matrices are all symmetric.

This is significant because when one has a symmetric matrix, the full orthogonalization performed by the Arnoldi procedure in GMRES reduces to the symmetric Lanczos procedure. The Lanczos procedure for symmetric problems produces a fully orthogonalized Krylov basis using a short-term recurrence (in particular, a three-term recurrence). Therefore, when the problem is symmetric, the sGMRES method with a k-vector partial orthogonalization should succeed, since a full orthogonalization reduces to a 3-vector partial orthogonalization in exact arithmetic. To avoid getting a full orthogonalization in the Laplacian example, Nakatsukasa and Tropp use only a 2-vector partial orthogonalization, but one should note that this is nearly a full orthogonalization. In the `t2em` example, the matrix is extremely close to symmetric, and a 4-vector partial orthogonalization is used. By the near symmetry of the problem, a 3-vector partial orthogonalization would likely be close to fully orthogonal, and so a 4-vector partial orthogonalization is unlikely to suffer from a significant loss of orthogonality.

Nakatsukasa and Tropp illustrate one hard example where sGMRES fails while GMRES does not, which uses the `fs_680_1` matrix from SuiteSparse. The authors recognize that there are other examples where the partial orthogonalization does not sufficiently avoid stagnation, but do not elaborate much further. It is worth noting that `fs_680_1` is the only example presented that is far from symmetric, with 51.1% pattern symmetry and 0% numeric symmetry [4].

For problems that are far from symmetric, one should not expect the partial orthogonalization to maintain an orthogonal basis. Since the standard Krylov basis in Equation (1.5) is severely ill-conditioned, the loss of orthogonality induced by the partial orthogonalization can cause the rank of the sGMRES Krylov basis B_m deteriorate, while the fully orthogonalized GMRES Krylov basis remains full rank. This results in earlier stagnation of the sGMRES residual, as mentioned in Section 1.4.

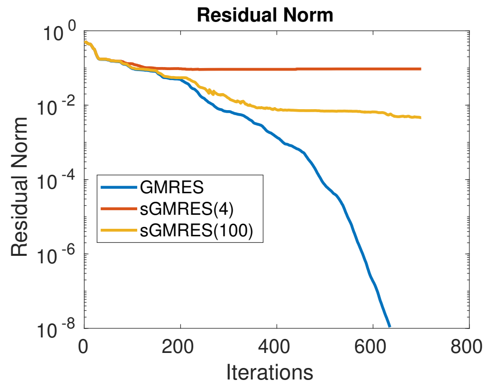
To illustrate this, we tested additional difficult, highly non-symmetric problems for sGMRES to solve. In all our experiments in this subsection, we solved $Ax = b$ using both GMRES and sGMRES, where $A \in \mathbb{R}^{n,n}$ is detailed on a case-by-case basis and $b \in \mathbb{R}^n$ is a sinusoidal vector where $b_k = \sin(k)$ for $k = 1, 2, \ldots, n$. Given a Gaussian matrix $G \in \mathbb{C}^{s,n}$ with $s = 2(m + 1)$ where m is the restart parameter for GMRES and sGMRES, then the sketch matrix in sGMRES is $S = \frac{1}{\sqrt{s}}G$.

The first example we show uses the `lhr04` matrix from the SuiteSparse library with 1.4 added to each diagonal entry, which we refer to as "$\texttt{lhr04} + 1.4I$" for brevity. The original `lhr04` matrix is an extremely challenging problem, as it is highly non-symmetric with 1.5% pattern symmetry and 0% numeric symmetry, and is severely ill-conditioned with a condition number of 4.1×10^{18} [4]. Adding 1.4 to each diagonal entry preserves the symmetry properties, but reduces the condition number down to 1.0×10^5. Thus, the problem is constructed in such a way that it is seemingly innocuous at first glance, but is close to an extremely challenging problem.

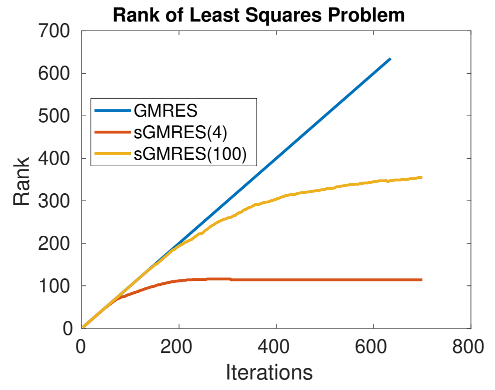
In our experiments with $\texttt{lhr04} + 1.4I$, we did not use restarts, and we set the restart parameter (i.e., the maximum number of iterations in this case) to $m = 700$. We compared GMRES to two different versions of sGMRES: the first being sGMRES with only a $k = 4$ vector partial orthogonalization which we refer to as sGMRES(4) in the plots, and then sGMRES with a $k = 100$ vector partial orthogonalization which we refer to as sGMRES(100). We plot the residual norm convergence behavior of each algorithm in Figure 2.1 (a). In Figure 2.1 (b), we plot $\text{rank}(AB_j)$ as the number of iterations increase where B_j is the Krylov basis for the method at iteration j using MATLAB's `rank` function for the numerical rank with default settings[1]. The behavior of $\text{rank}(AB_j)$ is of interest since AB_j is the coefficient matrix in the least squares problem (1.6). Thus, when $\text{rank}(AB_j) = \text{rank}(AB_{j+1})$, then the residual norm does not decrease from iteration j to iteration $j + 1$, since the least squares problem at iteration $j+1$ does gain any new information that was unavailable at iteration j. Therefore, we can expect worse convergence behavior of any GMRES type of method when AB_j is rank-deficient. Figure 2.1 (b) shows that the partial orthogonalization can lead to a severely low rank least squares matrix, even when a 100-vector partial orthogonalization is used. Figure 2.1 (a) shows this loss of rank leads to stagnation in the sGMRES residual norm.

2.2. Newton Basis sGMRES. In an attempt to improve the robustness of sGMRES without resorting to a fully orthogonalized basis, we tested a few alternative basis constructions involving the Newton basis described in Section 1.3. This can be incorporated into sGMRES in several ways. The first way is to replace the truncated Arnoldi procedure in lines 4-8 of Algorithm 2 with lines 3-4 of Algorithm 1, which we refer to as "Newton Basis

[1]The default threshold for the numerical rank of a matrix $Z \in \mathbb{C}^{n,m}$ using MATLAB's `rank` function is $\max\{n, m\}\texttt{eps}(\|Z\|_2)$, where $\texttt{eps}(\|Z\|_2)$ is the distance between $\|Z\|_2$ and the next largest floating point number.



(a) Residual Norm convergence behavior



(b) Rank of the least squares matrix AB_j, where B_j is the Krylov basis at iteration j.

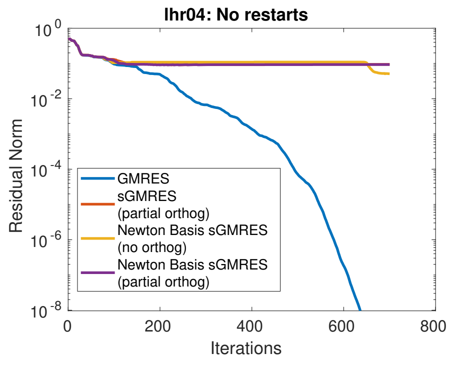
Fig. 2.1: Performance of non-restarted GMRES (blue) compared to sGMRES with a 4-vector partial orthogonalization (red) and a 100-vector partial orthogonalization (yellow).

sGMRES without orthogonalization". Additionally, one can still partially orthogonalize the basis, but replace line 4 in Algorithm 2 with line 3 of Algorithm 1 to try to stabilize the partial orthogonalization without incurring significantly more computational cost, which we refer to as "Newton Basis sGMRES with partial orthogonalization".

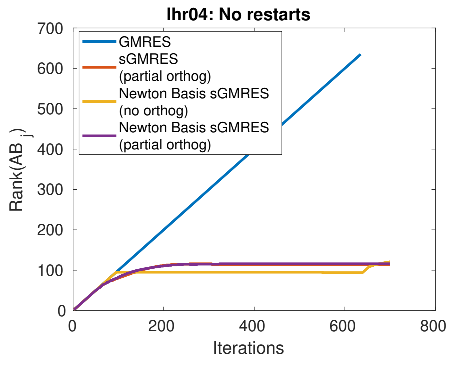
Figure 2.2 demonstrates experimental results for these Newton basis versions of sGMRES compared to the ordinary sGMRES with partial orthogonalization and to GMRES for the same `lhr04`$+1.4I$ test problem studied in section 2.1. All partially orthogonalized methods used a 4-vector partial orthogonalization. In figure 2.2 (a) and (b) we plot the residual norm and the rank of the least squares matrix AB_j at iteration j, just as we did in section 2.1, and in (c) we plot the condition number of the least squares matrix AB_j at iteration j. The results indicate that the use of the Newton basis, with or without partial orthogonalization, do not resolve the stagnation issue that sGMRES faces for this problem in general, due to a severely ill-conditioned, and ultimately rank-deficient least squares problem. However, we will investigate the effectiveness of these methods for practical Computational Fluid Dynamics problems in Section 2.3.

2.3. Effectiveness of sGMRES on Practical Computational Fluid Dynamics Problems. While it is understood from the experiments in Sections 2.1 and 2.2 that sGMRES with partial orthogonalization, a Newton basis, or a partially orthogonalized Newton basis can stagnate badly while GMRES converges well, the method is not entirely invalid just because it fails for a particularly difficult example. Indeed, while it is mathematically satisfying to understand whether or not a method is robust, we must consider how these methods perform on practical problems. Because sGMRES with a Newton basis without any orthogonalization can be executed very efficiently on high-performance machines, it may still be useful, provided the method works for realistic applications.

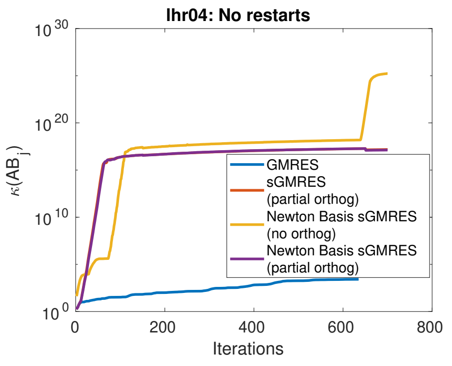
Let us first understand why the results shown thus far are not exactly a realistic demonstration of the sGMRES method's performance. First, the `lhr04` $+ 1.4I$ test problem was concocted to accomplish a specific task. The `lhr04` matrix is highly non-symmetric and severely ill-conditioned. Adding $1.4I$ to the matrix preserves the symmetry pattern, but reduces the condition number substantially. Therefore, `lhr04`$+1.4I$ is highly non-symmetric,



(a) Residual Norm convergence behavior



(b) Rank of the least squares matrix AB_j, where B_j is the Krylov basis at iteration j.



(c) $\kappa(AB_j)$, where B_j is the Krylov basis at iteration j.

Fig. 2.2: Performance of sGMRES with and without a Newton basis compared to GMRES

and almost severely ill-conditioned. This is not realistic for many physics and engineering applications.

Next, we did not use restarts on any of our methods and chose a restart parameter of 700. This is again unrealistic, because realistic implementations of any GMRES-type method would use a relatively small restart parameter, such as 50 in order to keep storage requirements and computational complexity low. Lastly, we did not use any preconditioning on our examples for simplicity.

This prompted us to consider applying the experiments in Section 2.2 to every Computational Fluid Dynamics (CFD) problem in SuiteSparse with 50,000 to 2,000,000 rows. We eliminated any symmetric positive definite (SPD) problem from our tests, as these were cases where Conjugate Gradient would obviously be preferable over any GMRES variant. We tested GMRES compared to sGMRES with a 4-vector partial orthogonalization (denoted as 'sGMRES(4)' in Figures 2.3 and 2.4), and sGMRES with a Newton Krylov basis without any orthogonalization (denoted as 'sGMRES Newton' in Figures 2.3 and 2.4). We allowed restarts for each method with a cycle length of 50, and used ILU(0) right-preconditioning. In large-scale parallel applications, ILU(0) would likely not be used in favor of other preconditioners that perform better in parallel, but for now we simply wanted to understand how the methods perform provided we have a reasonably good preconditioner, and real examples would choose preconditioners on a case-by-case basis. The ILU(0) preconditioner appears to perform well for these test problems, with the exception of the PR02R example shown in Figure 2.4 (f); this is due to the fact that the ILU(0) factorization produces a lower triangular part L with $\kappa(L) \approx 10^{58}$ and an upper triangular U with $\kappa(U) \approx 10^{63}$. Thus, the triangular solves done in the preconditioning step for this problem are very inaccurate, causing each of the algorithms to fail for this problem.

The residual norm convergence behavior for each test example are plotted in Figures 2.3 and 2.4. The title of each plot gives the name of the matrix used in the SuiteSparse library. The results show that Newton basis sGMRES without partial orthogonalization performs quite well. In spite of the method requiring more iterations than GMRES at times, we conjecture that the increased number of iterations is not substantial enough to outweigh the savings in computational cost by not orthogonalizing the basis vectors.

3. Conclusions.

In this manuscript, we tested the robustness of the sGMRES algorithm. We found that the algorithm can stagnate very quickly due to a severe loss in rank of the least squares matrix arising from the partially orthogonalized basis, even if one performs a partial orthogonalization with many vectors, while GMRES converges without issue for this same problem. We tested incorporating a Newton Krylov basis with and without the partial orthogonalization into sGMRES to improve the conditioning of the least squares problem, which also did not resolve the stagnation issue of sGMRES for a challenging, yet reasonably well conditioned problem.

However, we showed that Newton basis sGMRES without orthogonalization generally succeeds for practical CFD problems, and converges in nearly the same number of iterations as GMRES. This is interesting, as Newton basis sGMRES without orthogonalization is far less computationally expensive per iteration than GMRES. Provided Newton basis sGMRES without orthogonalization does not converge in significantly more iterations than GMRES, which was not the case in any of the experiments in Section 2.3, sGMRES should outperform GMRES on large parallel machines.

While the robustness of sGMRES is limited, ideas explored in this investigation may have additional benefits for other algorithms. Future work related to the sketching techniques used in this manuscript include incorporating sketching into the inner orthogonalization procedure in the block Arnoldi process, using ideas from a recent publication by Balabanov and Grigori on Randomized Gram-Schmidt [2]. This could help maintain a well-conditioned, full rank basis at low computational cost for a variety of methods using block Arnoldi, such as s-step GMRES and block GMRES [3, 6].

Sketching the least squares problem for block GMRES may also be worth investigating. Solving the block GMRES least squares problem is far more computationally expensive than the least squares problem for GMRES, since it involves converting a banded Hessenberg matrix to upper triangular form, compared to the very inexpensive Givens rotations used to convert the Hessenberg matrix to upper triangular form in GMRES [6, 7]. Thus, the computational savings realized by sketching the least squares problem for block GMRES may be more significant than those for GMRES.

4. Acknowledgments.

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

REFERENCES

[1] Z. Bai, D. Hu, and L. Reichel, *A Newton basis GMRES implementation*, IMA Journal of Numerical Analysis, 14 (1994), pp. 563–581.

[2] O. Balabanov and L. Grigori, *Randomized Gram–Schmidt process with application to GMRES*, SIAM Journal on Scientific Computing, 44 (2022), pp. A1450–A1474.

[3] A. T. Chronopoulos and S. K. Kim, *s-Step Orthomin and GMRES implemented on parallel computers*, Technical Report, University of Minnesota Supercomputing Institute, 1990.

[4] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software, 38 (2011).

[5] M. Embree, *The tortoise and the hare restart GMRES*, SIAM Review, 45 (2003), pp. 259–266.

[6] M. H. Gutknecht, *Block Krylov subspace methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, A. H. Siddiqi, I. S. Duff, and O. Christensen, eds., Anamaya Publishers, New Dehli, 2006, ch. 10, pp. 420–447.

[7] M. H. Gutknecht and T. Schmelzer, *Updating the QR decomposition of block tridiagonal and block Hessenberg matrices*, Applied Numerical Mathematics, 58 (2008), pp. 871–883.

[8] J. Liesen and Z. Strakoš, *Krylov subspace methods. Principles and analysis*, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 1st ed., 2013.

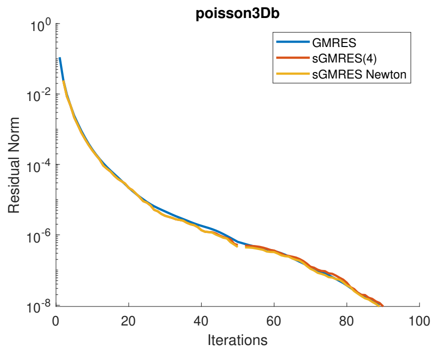
[9] Y. Nakatsukasa and J. A. Tropp, *Fast & accurate randomized algorithms for linear systems and eigenvalue problems*, 2021.

[10] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, 2nd ed., 2003.

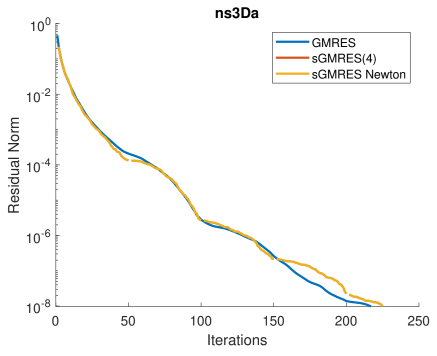
[11] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 7 (1986), p. 856–869.

[12] T. Sarlos, *Improved approximation algorithms for large matrices via random projections*, in 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 143–152.

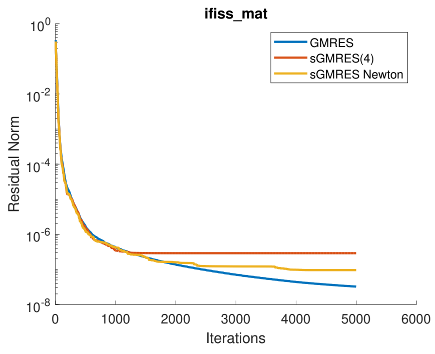
[13] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM Journal on Scientific and Statistical Computing, 13 (1992), pp. 631–644.



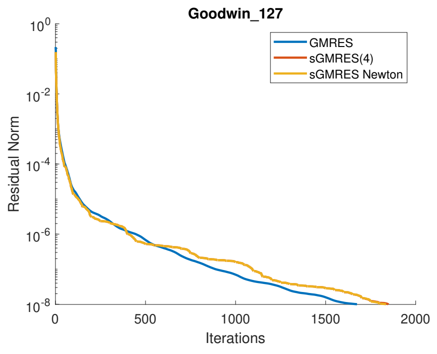
(a)



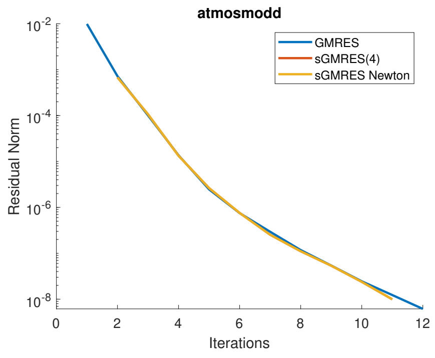
(b)



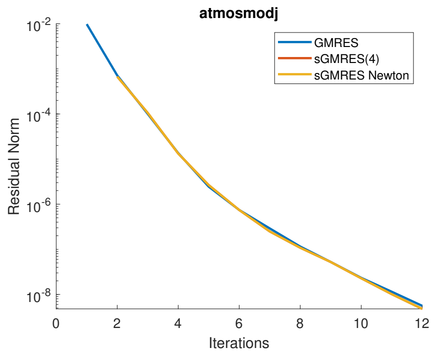
(c)



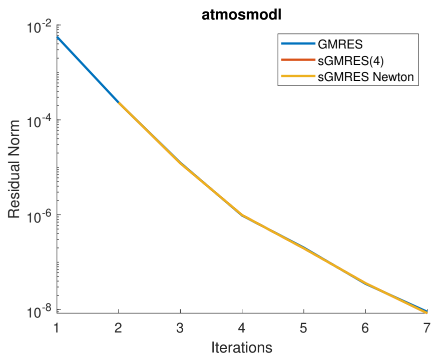
(d)



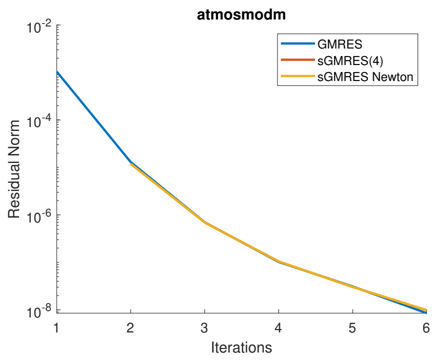
(e)



(f)

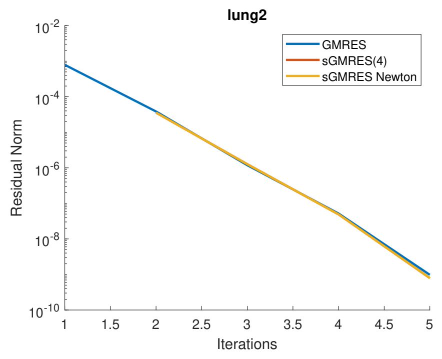


(g)

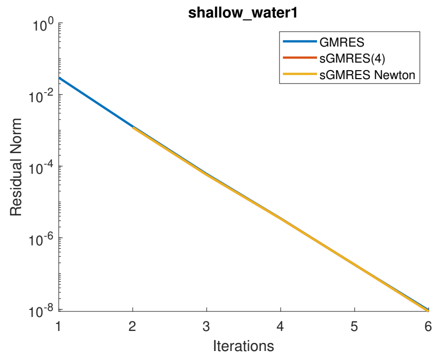


(h)

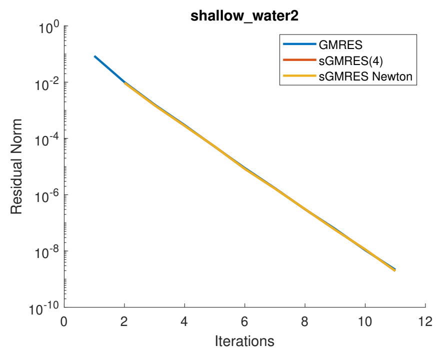
Fig. 2.3: Convergence behavior of sGMRES with a 4-vector partial orthogonalization and sGMRES with a Newton Krylov basis without orthogonalization compared to GMRES for CFD problems from SuiteSparse.



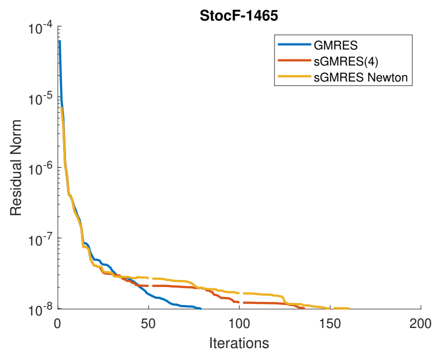
(a)



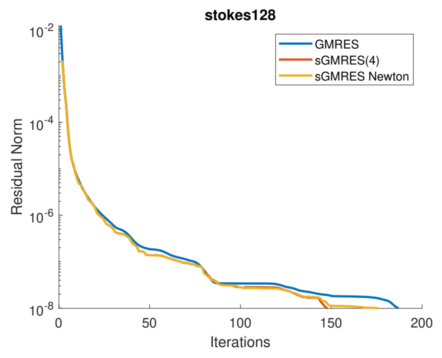
(b)



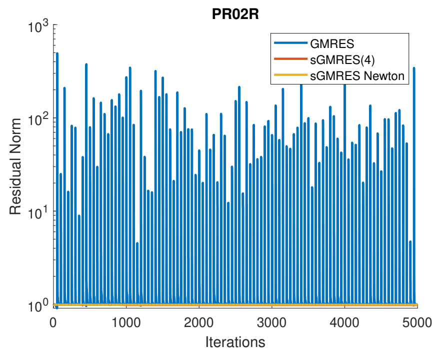
(c)



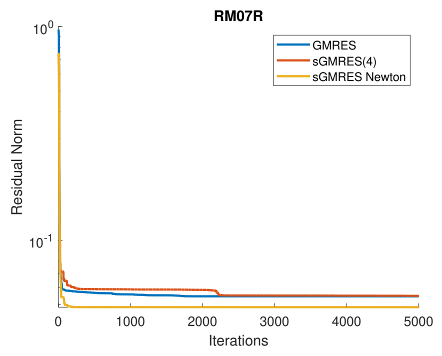
(d)



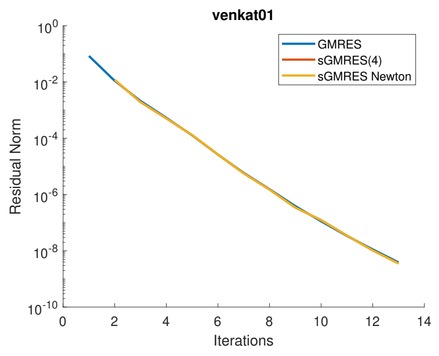
(e)



(f)



(g)



(h)

Fig. 2.4: Convergence behavior of sGMRES with a 4-vector partial orthogonalization and sGMRES with a Newton Krylov basis without orthogonalization compared to GMRES for CFD problems from SuiteSparse.

LYAPUNOV CONTROL-INSPIRED QUANTUM ALGORITHMS FOR GROUND STATE PREPARATION

JAMES B. LARSEN*, MATTHEW D. GRACE†, ANDREW D. BACZEWSKI†, AND ALICIA B. MAGANN†

Abstract. Finding the ground state of a quantum system is a well studied problem with applications spanning chemistry, physics, and material science. There has been a recent surge of interest from the quantum computing community in variational quantum algorithms for solving this problem. These algorithms utilize a hybrid quantum-classical framework to prepare ground states, requiring classical optimization protocols that become prohibitively expensive in higher dimensions. We address this challenge through the development of a feedback-based quantum algorithm for ground state preparation that is inspired by quantum Lyapunov control and is optimization free. We demonstrate this algorithm through numerical simulation of the Fermi-Hubbard model for correlated systems and explore aspects of the algorithm's performance. This new method provides a pathway to enhance our understanding of the Fermi-Hubbard model and other quantum systems.

1. Introduction.

In recent years, rapid progress in the development of quantum computing technologies has given rise to the current noisy intermediate-scale quantum (NISQ) era [32]. This has motivated substantial research exploring applications of NISQ devices in a variety of areas, including quantum simulation. In particular, the prospect of utilizing NISQ devices to find ground states of chemical and materials systems is currently receiving significant attention. A variety of approaches have been developed for this task [2], including Quantum Phase Estimation [7, 16], Imaginary Time Evolution [10, 28, 30], and the Variational Quantum Eigensolver (VQE) [31]. VQE is a hybrid quantum-classical algorithm, and may hold particular promise in the near-term. However, a practical challenge associated with VQE is the need to classically optimize over a set of quantum circuit parameters, a task whose complexity can increase rapidly with the dimension of the search space.

To avoid this increased complexity, we investigate an alternative formulation for finding ground states that is optimization free. Instead of classically optimizing over a parameterized circuit, we utilize a feedback law to sequentially set the parameters layer-by-layer, conditioned on feedback from qubit measurements. Our feedback law is based on quantum Lyapunov control, and it guarantees that the value of our objective function monotonically decreases with respect to circuit depth.

In particular, this method extends the Feedback-based ALgorithm for Quantum OptimizatioN (FALQON) [24, 25], a feedback-based approach for solving combinatorial optimization problems on quantum computers that utilizes the quantum approximate optimization algorithm (QAOA) ansatz from 2014 [9]. The QAOA ansatz is formed by a sequence of layers, where each layer is composed of alternating unitaries generated by a "problem" Hamiltonian, which is an Ising encoding of a combinatorial optimization problem [23], and a "driver" Hamiltonian. QAOA has subsequently inspired the development of other ansätze, including the Hamiltonian variational ansatz (HVA) introduced in [36] in 2015. The HVA aims to generalize the QAOA ansatz to other classes of Hamiltonians beyond Ising models and is based on a similar premise: each layer contains a sequence of parameterized unitaries generated by portions of the problem Hamiltonian. If the qubits are initialized in the ground state of a simple (e.g., noninteracting) portion of the problem Hamiltonian, then it can be shown that the ground state of the full problem Hamiltonian is reachable with enough layers using the HVA, invoking the adiabatic theorem just as in QAOA.

*Brigham Young University Department of Mathematics, ljamesb@byu.edu
†Sandia National Laboratories, abmagan@sandia.gov

Here, we explore applications of feedback-based quantum algorithms to finding ground states of the Fermi-Hubbard model. The Fermi-Hubbard model is a model of strongly correlated solid-state systems that was introduced by John Hubbard in 1963 [13] and has since been a subject of intense interest [1, 33]. The model describes the behavior of fermions that are allowed to hop between lattice sites and experience repulsive on-site interactions. Despite its simplicity, the Fermi-Hubbard model is capable of capturing a variety of orderings and certain details of its phase diagram remains an open topic of research. As an approximation to a more detailed model of numerous material systems, it is difficult to unambiguously constrain the model's phase diagram through experiments. Thus, classical and quantum computers provide the only way to unambiguously study the model in and of itself. At zero temperature, this involves approximating the ground state of the model for finite-size instances and extrapolating to the thermodynamic limit.

The 2D square lattice Fermi-Hubbard model is among the most studied variants of this model and it is thought to capture phenomena related to high-temperature superconductivity in hole-doped cuprates. A recent comparison of numerous state-of-the-art classical methods for studying this variant can be found in [21]. The authors find that the intermediate coupling regime near half-filling is especially difficult to study and has the most uncertain results; this may also be the regime relevant to cuprate superconductivity [34].

The difficulty of studying relevant regimes highlights one of the key challenges that motivates the use of quantum computers to study the Fermi-Hubbard model. Classical computers force us to choose between exact algorithms requiring resources that scale exponentially with the number of lattice sites, and approximate algorithms that are efficient but without easily quantifiable errors. The exponential cost of exact classical algorithms is nicely highlighted by one of the largest exact simulations done on a trapped instance of the Fermi-Hubbard model, with 22 sites and 17 electrons [37]. To approach the thermodynamic limit while preserving quantifiable error, it may be that quantum computers provide a decisive advantage. While we do not expect to be able to efficiently prepare the ground state of arbitrary Hamiltonians on quantum computers, there is good reason to expect that many physical Hamiltonians possess ground states that can be efficiently prepared. In addition to numerical calculations on classical and quantum computers, the Fermi-Hubbard model can also be studied using analog quantum simulation. The two major platforms for analog quantum simulation of the Fermi-Hubbard model are cold atoms in optical lattices [3, 18, 26, 35] and semiconductor quantum dot arrays [8, 12, 14, 19, 20].

In this work, we formulate feedback-based quantum algorithms that utilize the HVA to prepare ground states of the Fermi-Hubbard model in a manner that is optimization free. The paper is organized as follows: in Sec. 2, we describe the necessary preliminary theory behind our proposed method. This includes an explanation of quantum Lyapunov control, FALQON, the Fermi-Hubbard model, the HVA, and Jordan-Wigner (JW) fermion-to-qubit encoding. Next, in Sec. 3, we combine these preliminaries and introduce our feedback-based quantum algorithm for ground state preparation. We then show several numerical results on different instances of the Fermi-Hubbard model in Sec. 4, and conclude with a discussion of our results and possible future avenues of research in Sec. 5.

2. Preliminaries.

In this section, we provide technical details behind the key ideas utilized in this work.

2.1. Quantum Lyapunov Control.

The inspiration for this is quantum Lyapunov control theory, which is a framework for designing one or multiple time-dependent controls to drive a quantum dynamical system towards a desired objective in an asymptotic manner [6, 11]. To describe the method, we begin by considering a quantum system whose dynamics are dictated by the time-dependent Schrödinger equation,

$$i\frac{d}{dt}|\psi(t)\rangle = \big(H_p + H_d\beta(t)\big)|\psi(t)\rangle, \tag{2.1}$$

where H_p denotes the time-independent portion of the Hamiltonian and H_d denotes the control Hamiltonian that couples a time-dependent control function, $\beta(t)$, to the system. We consider the control problem of designing $\beta(t)$ to drive $|\psi(t)\rangle$ to a state that minimizes an objective function J. Quantum Lyapunov control has been formulated for a variety of different objective functions, such as capturing the distance from a target state, the target state error, or the expectation value of a target observable. Here, we define our objective function to be the expected value of H_p:

$$J(|\psi(t)\rangle) = \langle\psi(t)|H_p|\psi(t)\rangle. \tag{2.2}$$

To solve this control problem, we demand that $\beta(t)$ is defined in a manner that satisfies

$$\frac{dJ}{dt} \leq 0, \forall t \tag{2.3}$$

such that the objective function decreases monotonically over time (ideally, to the global minimum of J).

Evaluating the left-hand-side of Eq. (2.3) using Eqs. (2.1) and (2.2), we find

$$\begin{aligned}\frac{dJ}{dt} &= \frac{d}{dt}\langle\psi(t)|H_p|\psi(t)\rangle \\ &= \left(\frac{d}{dt}\langle\psi(t)|\right)H_p|\psi(t)\rangle + \langle\psi(t)|H_p\left(\frac{d}{dt}|\psi(t)\rangle\right) \\ &= \Big(\langle\psi(t)|i(H_p + H_d\beta(t))\Big)H_p|\psi(t)\rangle + \langle\psi(t)|H_p\Big(-i(H_p + H_d\beta(t))|\psi(t)\rangle\Big) \\ &= \langle\psi(t)|i[H_d, H_p]|\psi(t)\rangle\beta(t) \\ &\equiv A(t)\beta(t),\end{aligned} \tag{2.4}$$

where $[C, D] = CD - DC$ denotes the commutator of C and D and we have introduced the abbreviated notation $A(t) \equiv \langle\psi(t)|i[H_d, H_p]|\psi(t)\rangle$. Given the form of Eq. (2.4), the condition in Eq. (2.3) is satisfied for $\beta(t) = -wf(t, A(t))$, where $w > 0$ is a positive weight and $f(t, A(t))$ is a function selected such that $f(t, 0) = 0$ and $A(t)f(t, A(t)) > 0$ for all $A(t) \neq 0$.

One formulation is to choose $\beta(t)$ according to the following simple control law:

$$\beta(t) = -A(t) = -\langle\psi(t)|i[H_d, H_p]|\psi(t)\rangle, \tag{2.5}$$

so that the condition given by Eq. (2.3) is satisfied, i.e., $\frac{dJ}{dt} = -\big(A(t)\big)^2 \leq 0$.

2.2. FALQON. FALQON is a quantum algorithm developed specifically to solve combinatorial optimization problems on quantum computers in a manner that is optimization-free, thus distinguishing it from other candidate strategies such as the QAOA. This is accomplished by first mapping the combinatorial optimization problem under consideration to an Ising Hamiltonian, H_p, such that the ground state of H_p encodes the solution to the combinatorial optimization problem [23]. Then, the task is to minimize an objective function $J = \langle\psi|H_p|\psi\rangle$ over a set of parameters $\vec{\beta}$ that influence the state $|\psi\rangle$. FALQON is a feedback-based quantum algorithm that assigns values to each element in $\vec{\beta}$ according to a feedback law derived from quantum Lyapunov control theory, as outlined in Sec. 2.1. We begin this section by describing how the controlled quantum time evolution described by Eq. (2.1) could be simulated on a quantum computer. We then go on to discuss FALQON implementation details.

The solution to Eq. (2.1) for time-dependent Hamiltonians is given by

$$|\psi(t)\rangle = \mathcal{T} e^{-i\int_0^t (H_p + H_d\beta(t'))dt'}|\psi(0)\rangle, \tag{2.6}$$

where $\mathcal{T}$ denotes the time-ordering operator. We first digitize this evolution into a sequence of ℓ discrete time steps and approximate the Hamiltonian as time-independent over each step, such that

$$|\psi(t+\Delta t)\rangle = e^{-i(H_p+\beta(t)H_d)\Delta t}|\psi(t)\rangle. \tag{2.7}$$

Then, we approximate the evolution over each time step using Trotterization, i.e.,

$$e^{-i(H_p+\beta_k H_d)\Delta t} \approx e^{-i\beta_k H_d \Delta t} e^{-iH_p\Delta t}. \tag{2.8}$$

which can be justified for small Δt using Trotter's formula,

$$e^{i(C+D)t} = \lim_{n\to\infty} (e^{iCt/n} e^{iDt/n})^n. \tag{2.9}$$

In total, this sequence of approximations serves to break the complicated unitary governing the time evolution in Eq. (2.6) into a product of much simpler exponentials, i.e.,

$$\begin{aligned} |\psi_\ell\rangle &= e^{-i\beta_\ell H_d\Delta t} e^{-iH_p\Delta t} \cdots e^{-i\beta_1 H_d \Delta t} e^{-iH_p\Delta t}|\psi_0\rangle \\ &= U_d(\beta_\ell)U_p ... U_d(\beta_1)U_p|\psi_0\rangle. \end{aligned} \tag{2.10}$$

In Eq. (2.10), we have adopted the abbreviated notation $\beta_k \equiv \beta(2k\Delta t)$; $|\psi_k\rangle \equiv |\psi(2k\Delta t)\rangle$; $U_p \equiv e^{-iH_p\Delta t}$; and $U_d(\beta_k) \equiv e^{-i\beta_k H_d \Delta t}$, for $k = 1, 2, ..., \ell$.

We now describe the mechanics of implementing FALQON. In the following, we use the simplified notation $A_k \equiv \langle\psi_k|i[H_d, H_p]|\psi_k\rangle$. The initial step is to select an appropriate initial state $|\psi_0\rangle$ and to seed the algorithm with $\beta_1 = 0$. Then, FALQON utilizes the following feedback law at each step $1 < k \leq \ell$,

$$\beta_k = -A_{k-1} = -\langle\psi_{k-1}|i[H_d, H_p]|\psi_{k-1}\rangle \tag{2.11}$$

which is a digitized version of the control law given earlier in Eq. (2.5). We call Eq. (2.11) a feedback law because at each step, A_{k-1} is "fed back" to set the value of the next β_k. In general, the values of each A_k can be estimated via repeated measurements of the observable $i[H_d, H_p]$, decomposed into a linear combination of multiqubit Pauli operators, in the state $|\psi_k\rangle$. The free parameters in FALQON are the choice of the time step Δt, initial state $|\psi_0\rangle$, and number of layers ℓ. For further implementation details, see Fig. 2.2, as well as [24, 25].

2.3. Hamiltonian Variational Ansatz. The Hamiltonian variational ansatz (HVA) proposed in [36] is going to be our ansatz of choice for generalizing the FALQON feedback law for ground state preparation. The HVA is motivated by QAOA and, more broadly, the premise of quantum annealing. For the HVA, instead of separating out our problem Hamiltonian and our driver Hamiltonian, we can instead decompose our problem Hamiltonian as a sum of different terms,

$$H_p = \sum_{j=1}^{m} H_j. \tag{2.12}$$

From here, we can use Trotter's formula, Eq. (2.9), to approximate the time evolution under this Hamiltonian as a product of individual exponentials, which we may additionally parameterize by $\{\beta_k^j\}_{j=1}^m$, where j labels the corresponding term in the Hamiltonian and k labels the associated layer of HVA. This gives us that the kth layer of our Trotterized evolution is as follows:

$$|\psi_{k+1}\rangle = e^{-i\beta_k^1 H_1 \Delta t} e^{-i\beta_k^2 H_2 \Delta t} ... e^{-i\beta_k^m H_m \Delta t} |\psi_k\rangle. \tag{2.13}$$

This ansatz is discussed in greater detail in [29].

We can see that for appropriate choices of $\{\beta_k^j\}_{j=1}^m$, this method can be made very similar to a digitized implementation of quantum annealing. That is, if H_x is a Hamiltonian with a known and easily prepared ground state, we could let $\beta_0^x = 1$ and $\beta_0^{j \neq x} = 0$. As time increases, we could slowly turn on the other Hamiltonians, $H_{j \neq x}$, such that $\beta^{j \neq x} \to 1$, in order to generate an adiabatic transition to our new, sought-after ground state of the full Hamiltonian H_p in Eq. (2.12).

2.4. The Fermi-Hubbard Model. The Fermi-Hubbard model describes a lattice of fermions. As opposed to bosons, fermions obey the Pauli-exclusion principle, meaning no two fermions can occupy the same state at the same time. Throughout this section, we use hats to distinguish fermionic operators from their eigenvalues.

First, we define the Hilbert space associated with a lattice site x:

$$\mathcal{H}_x = \text{span}\{|n_{x\uparrow} n_{x\downarrow}\rangle : n_{x\sigma} \in \mathbb{Z}_2\}, \tag{2.14}$$

where σ denotes the spin of the fermion, and can take the value of either spin up, $\uparrow$, or spin down, $\downarrow$. This notation is an occupation number representation, where each $n_{x\sigma}$ is referred to as a spin orbital. Specifically, when $n_{x\sigma} = 1$, the lattice site x is occupied by one fermion of spin σ. For example, $|11\rangle$ indicates full occupation of site x by fermions of both spins and $|00\rangle$ indicates the site is unoccupied. Considering a system of N lattice sites, we get the following Hilbert space:

$$\mathcal{H}_{lattice} = \bigotimes_{x=1}^{N} \mathcal{H}_x \tag{2.15}$$

$$= \text{span}\{|n_{1\uparrow} n_{1\downarrow} ... n_{N\uparrow} n_{N\downarrow}\rangle : n_{x\sigma} \in \mathbb{Z}_2\}. \tag{2.16}$$

From this, we can see that each vector in the Hilbert space is going to be a normalized vector in $\mathbb{C}^{2^{2N}}$, with each element of the vector corresponding to the amplitude of a certain arrangement of the fermions across the lattice.

Prior to formalizing the Fermi-Hubbard Hamiltonian, we need to define several operators on the Hilbert space of the lattice. First, we will define the annihilation operator:

$$\hat{a}_{x\sigma}|...n_{x\sigma}...\rangle = \begin{cases} p_{x\sigma}|...0...\rangle & \text{if } n_{x\sigma} = 1 \\ 0 & \text{if } n_{x\sigma} = 0 \end{cases}, \tag{2.17}$$

and its Hermitian conjugate, the creation operator:

$$\hat{a}_{x\sigma}^\dagger|...n_{x\sigma}...\rangle = \begin{cases} 0 & \text{if } n_{x\sigma} = 1 \\ p_{x\sigma}|...1...\rangle & \text{if } n_{x\sigma} = 0 \end{cases}. \tag{2.18}$$

The variable $p_{x\sigma}$ corresponds to a factor of 1 or -1 that depends on the "parity" of the state, where a factor of -1 represents an odd number of fermions in the system. Note that neither of the above operators are Hermitian, but combining them gives us the number operator, which is Hermitian:

$$\hat{a}^\dagger_{x\sigma}\hat{a}_{x\sigma}|...n_{x\sigma}...\rangle \equiv \hat{n}_{x\sigma}|...n_{x\sigma}...\rangle = n_{x\sigma}|...n_{x\sigma}...\rangle. \tag{2.19}$$

The number operator satisfies an eigenvector relation with eigenvalue zero or one. We can use this number operator in the construction of our Hamiltonian to count the number of fermions in each site of the lattice.

With these definitions in place, we now introduce the Fermi-Hubbard Hamiltonian, which we denote by H_{FH}. We note that general fermionic Hamiltonians are described in detail in Section 3.1 of [5]. The Fermi-Hubbard Hamiltonian contains two terms:

$$H_{FH} = T + V. \tag{2.20}$$

The T term represents the kinetic energy associated with fermions moving, or "hopping," between sites on the lattice and forms the noninteracting portion of the Hamiltonian:

$$T = -\sum_{\langle m,n\rangle} t_{m,n}(\hat{a}^\dagger_{m\uparrow}\hat{a}_{n\uparrow} + \hat{a}^\dagger_{n\uparrow}\hat{a}_{m\uparrow} + \hat{a}^\dagger_{m\downarrow}\hat{a}_{n\downarrow} + \hat{a}^\dagger_{n\downarrow}\hat{a}_{m\downarrow}), \tag{2.21}$$

where $\langle m,n\rangle$ labels adjacent sites in the lattice and $t_{m,n}$ is the kinetic energy associated with hopping from site m to site n, also referred to as the tunneling amplitude. In our case, for all $\langle m,n\rangle$-pairs, we set $t_{m,n}$ to be equal to the same value, τ. We remark that the Fermi-Hubbard model can be studied with or without periodic boundary conditions, and in this work, we opt for the latter.

Meanwhile, V in Eq. (2.20) represents the local potential energy at each site that is due to the on-site interaction between fermions:

$$V = U\sum_{k=1}^{N} \hat{n}_{k\uparrow}\hat{n}_{k\downarrow}, \tag{2.22}$$

with U representing the on-site repulsive interaction strength.

The Fermi-Hubbard model can be solved analytically in the tight-binding limit where $U/\tau \to 0$ and also in the atomic limit where $\tau/U \to 0$. However, in the general case, an analytical solution is only known for the 1D case, which utilizes the Bethe ansatz [22]. Different regimes of U/τ are studied in materials science for analyzing different properties. For example, [18] looked specifically at $U/\tau \approx 14$.

2.4.1. HVA formulation for the Fermi-Hubbard model. We now consider ways to formulate the HVA for the Fermi-Hubbard Hamiltonian. One possibility is for the H_j's in Eq. (2.12) to correspond to $H_1 = T$, the kinetic energy associated with tunneling from one site to another, and $H_2 = V$, the potential energy associated with local repulsion at each site, as given in Eq. (2.20). As proposed in [4], another possibility is to further divide the Hamiltonian by separating T into sets of mutually commuting terms. For example, a 1D model would have two sets of terms: H_{h1} composed of terms associated with even-numbered sites and H_{h2} for odd sites. A 2D model would have two additional sets of terms, H_{v1} and H_{v2}, taking into account the vertical dimension. This procedure splits the 2D Fermi-Hubbard Hamiltonian into five pieces total. Each of the five pieces can then be parameterized, giving us that layer k of the HVA has the form:

$$U(\beta_k^{v2}, \beta_k^{h2}, \beta_k^{v1}, \beta_k^{h1}, \beta_k^{V}) = e^{-i\beta_k^{v2}H_{v2}}e^{-i\beta_k^{h2}H_{h2}}e^{-i\beta_k^{v1}H_{v1}}e^{-i\beta_k^{h1}H_{h1}}e^{-i\beta_k^{V}V}, \tag{2.23}$$

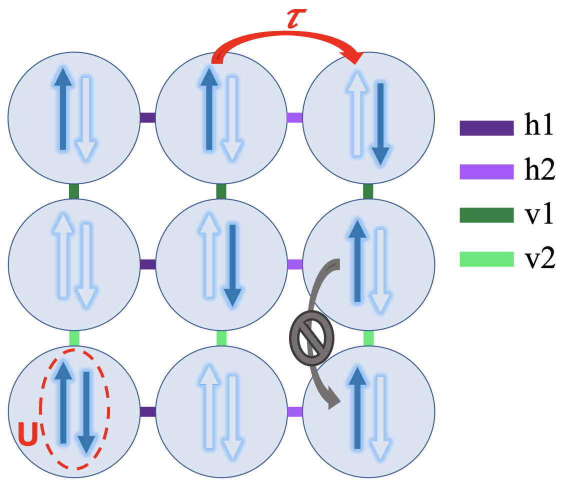


FIG. 2.1. *Configuration of 5 $\uparrow$ and 3 $\downarrow$ fermions in a 3×3 Fermi-Hubbard lattice. The potential energy associated with on-site interactions between fermions has strength U, and the tunneling amplitude associated with fermions hopping between adjacent lattice sites is given by τ, as shown in red. Two fermions with the same spin cannot occupy the same state, and thus, fermions are prohibited from hopping to adjacent states that are already occupied, which we illustrate in gray. In addition, the HVA decomposition of the hopping terms in T is shown in different shades of purple (horizontal terms $h1$ and $h2$) and green (vertical terms $v1$ and $v2$).*

where we note the similarities between Eq. (2.13) and Eq. (2.23). In Sec. 3, we take advantage of these similarities to formulate a feedback-based quantum algorithm for the Fermi-Hubbard model with the HVA, thereby extending the premise of FALQON to a new application.

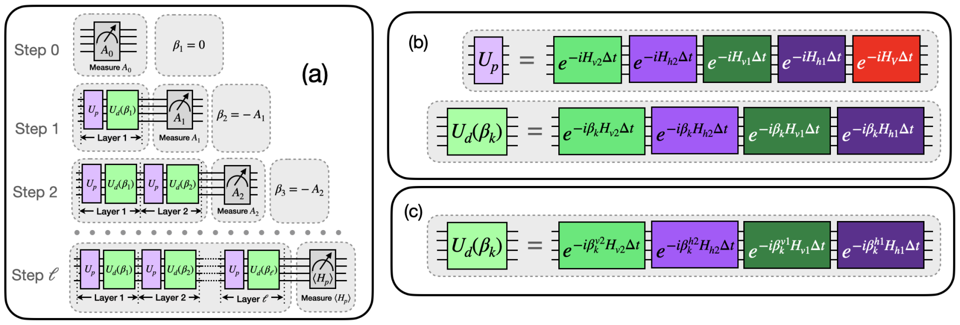

Step 0
Measure A_0
$\beta_1 = 0$
(a)
Step 1
U_p $U_d(\beta_1)$
Layer 1
Measure A_1
$\beta_2 = -A_1$
Step 2
Layer 1 Layer 2
Measure A_2
$\beta_3 = -A_2$
Step ℓ
Layer ℓ
Measure $\langle H_p \rangle$
(b)
$U_p = e^{-iH_{v2}\Delta t}\, e^{-iH_{h2}\Delta t}\, e^{-iH_{v1}\Delta t}\, e^{-iH_{h1}\Delta t}\, e^{-iH_{V}\Delta t}$
$U_d(\beta_k) = e^{-i\beta_k H_{v2}\Delta t}\, e^{-i\beta_k H_{h2}\Delta t}\, e^{-i\beta_k H_{v1}\Delta t}\, e^{-i\beta_k H_{h1}\Delta t}$
(c)
$U_d(\beta_k) = e^{-i\beta_k^{v2} H_{v2}\Delta t}\, e^{-i\beta_k^{h2} H_{h2}\Delta t}\, e^{-i\beta_k^{v1} H_{v1}\Delta t}\, e^{-i\beta_k^{h1} H_{h1}\Delta t}$

FIG. 2.2. *Feedback-based quantum algorithm for the Fermi-Hubbard model, based on the premise of FALQON. (a) The steps for implementing the algorithm. Each step k involves extending the quantum circuit by one layer, i.e., concatenating the quantum circuit from the prior step $k-1$ with $U_d(\beta_k)U_p$. The value of β_k at each layer k is set using the results from the prior step $k-1$ according to the feedback law $\beta_k = -A_{k-1}$. (b) The specific decompositions for U_p and U_d based on the HVA that are used in this work, shown here for the case of a single control function, with colors loosely corresponding to Fig. 2.1. (c) Our decomposition of U_d in the case of multiple control functions.*

2.5. Jordan-Wigner Encoding. In our Fermi-Hubbard model description, we represented whether or not a particular spin orbital was occupied with $|1\rangle$ or $|0\rangle$, with the state $|1\rangle$ corresponding to occupation of the spin orbital and the state $|0\rangle$ corresponding to the spin orbital being unoccupied. In this section, we consider how to map this fermion occupation number representation to a qubit representation using the Jordan-Wigner (JW) encoding scheme. JW encoding relies on properties of operators to maintain antisymmetry conditions associated with fermions, and is arguably more intuitive than other encoding methods such as Bravyi-Kitaev or parity encoding. The primary difference between the encoding schemes is the locality of storing parity versus storing occupation number. In JW encoding, the occupation number is stored locally and the parity is stored through design of the associated operators. Other encoding schemes alter this paradigm by storing a mix of the parity and occupation number locally, see [27] for a more in depth description of this. In this section, we adopt the notation $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ to denote single-qubit Pauli operators.

We can encode the creation and annihilation operators using JW encoding as follows [15, 27]:

$$\hat{a}_x \mapsto Q_x \otimes Z_{x-1} \otimes ... \otimes Z_0, \tag{2.24}$$

$$\hat{a}_x^\dagger \mapsto Q_x^\dagger \otimes Z_{x-1} \otimes ... \otimes Z_0. \tag{2.25}$$

For notational convenience, we now use the subscript x to denote each different spin orbital instead of indexing spin up and spin down separately. We encode Q_x as $\frac{1}{2}(X + iY)$, and as such $Q_x^\dagger = \frac{1}{2}(X - iY)$. The string of Z operators multiplies the result by the proper parity factor p_x, where $p_x = (-1)^{\sum_{k=0}^{x-1} n_k}$. The benefit of using this encoding is that we can write our Hamiltonian from the Fermi-Hubbard model entirely in terms of strings of Pauli matrices, which are easy to work with on a quantum computer.

We can additionally derive the encoding for each $\hat{a}_m^\dagger \hat{a}_n + \hat{a}_n^\dagger \hat{a}_m$ term from the Fermi-Hubbard Hamilton in Eq. (2.21) with JW encoding as follows:

$$\begin{aligned} \hat{a}_m^\dagger \hat{a}_n + \hat{a}_n^\dagger \hat{a}_m &\mapsto \frac{1}{4}[(X_m - iY_m)(X_n + iY_n) + (X_n - iY_n)(X_m + iY_m)]Z_{m+1}...Z_{n-1} && (2.26)\\ &= \frac{1}{4}(X_m X_n + Y_m Y_n + X_n X_m + Y_n Y_m)Z_{m+1}...Z_{n-1} && (2.27)\\ &= \frac{1}{2}(X_m X_n + Y_m Y_n)Z_{m+1}...Z_{n-1}. && (2.28)\end{aligned}$$

Note that Eq. (2.28) is exactly the encoding scheme provided by [4].

3. Feedback-based Quantum Algorithm for the Fermi-Hubbard Model. We now detail how the FALQON framework can be adapted to obtain a feedback-based quantum algorithm for preparing ground states of the Fermi-Hubbard model.

3.1. Single-Driver Hamiltonian. The simplest generalization of FALQON for the Fermi-Hubbard model is to define the driver Hamiltonian in Eq. (2.10) as $H_d = T$ and the problem Hamiltonian in Eq. (2.10) as $H_p = H_{FH} = T + V$. In this manner, we will have one β-parameter per layer of our quantum circuit, and the complete evolution will take the form of Eq. (2.10).

We select our initial state $|\psi_0\rangle$ as the ground state (associated with the appropriate fermion number and spin sector) of T which maintains an analogy to quantum annealing. This follows a standard practice in the literature with Fermi-Hubbard ground state preparation protocols, and the state can be prepared using quantum circuits based on Givens rotations [15, 17]. Since we start in this ground state, we let $H_p = H_{FH}$ and $H_d = T$. With these choices of problem and driver Hamiltonian, our feedback law from Eq. (2.11) is given by

$$\beta_{k+1} = -\langle\psi_k|i[T, H_{FH}]|\psi_k\rangle = -\langle\psi_k|i[T, V]|\psi_k\rangle, \tag{3.1}$$

where $|\psi_k\rangle$ is the state after k layers, as given by Eq. (2.10). This feedback law for β_k guarantees a monotonic decrease in the value of our objective function, J. Numerically, we observe an associated increase in ground state occupation, further discussed in Sec. 4.

To form the layers of our quantum circuits, we use the HVA to decompose the U_p and $U_d(\beta_k)$ terms as

$$\begin{aligned} U_p &= e^{-iH_{FH}\Delta t} && (3.2)\\ &= e^{-i(H_{v2}+H_{h2}+H_{v1}+H_{h1}+V)\Delta t} && (3.3)\\ &\approx e^{-iH_{v2}\Delta t}e^{-iH_{h2}\Delta t}e^{-iH_{v1}\Delta t}e^{-iH_{h1}\Delta t}e^{-iV\Delta t}. && (3.4)\end{aligned}$$

and

$$\begin{aligned} U_d(\beta_k) &= e^{-i\beta_k T\Delta t} && (3.5)\\ &= e^{-i\beta_k(H_{v2}+H_{h2}+H_{v1}+H_{h1})\Delta t} && (3.6)\\ &\approx e^{-i\beta_k H_{v2}\Delta t}e^{-i\beta_k H_{h2}\Delta t}e^{-i\beta_k H_{v1}\Delta t}e^{-i\beta_k H_{h1}\Delta t}. && (3.7)\end{aligned}$$

Notice the similarities between Eqs. (3.4), (3.7) and Eq. (2.23). In the next section, we further build on this decomposition.

3.2. Multiple Control Functions. The decomposition in Eq. (3.7) provides a natural extension for using multiple control functions as outlined in [25]. In particular, it enables us to treat each of the four noninteracting terms of the HVA as separately controlled terms, giving us the following modification to Eq. (2.4):

$$\frac{dJ}{dt} = \sum_{j\in\{v2,h2,v1,h1\}} \langle\psi(t)|i[H_j, H_{FH}]|\psi(t)\rangle\beta^j(t), \tag{3.8}$$

where β^j denotes the control function associated with H_j.

To ensure Eq. (3.8) is nonpositive, we can set our feedback law in the digitized picture according to a generalization of Eq. (2.11), yielding:

$$\beta^j_{k+1} = -\langle\psi_k|i[H_j, H]|\psi_k\rangle, \tag{3.9}$$

where $j \in \{v2, h2, v1, h1\}$. This yields the following modified formulation for each layer:

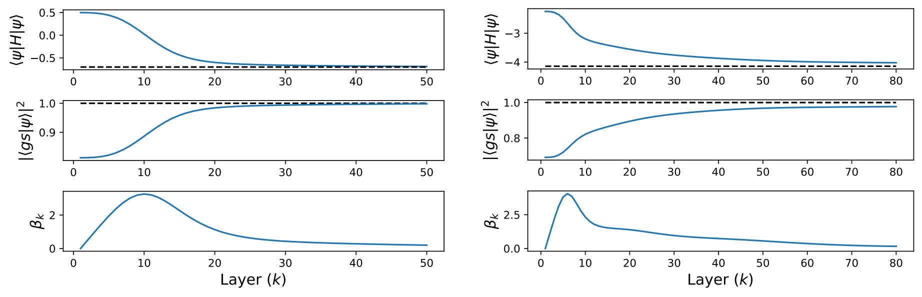
$$|\psi_{k+1}\rangle = U_d(\beta^{v2}_k, \beta^{h2}_k, \beta^{v1}_k, \beta^{h1}_k)U_p|\psi_k\rangle, \tag{3.10}$$

where U_p retains the same form as in Eq. (3.4), and

$$U_d(\beta^{v2}_k, \beta^{h2}_k, \beta^{v1}_k, \beta^{h1}_k) = e^{-i\beta^{v2}_k H_{v2}\Delta t}e^{-i\beta^{h2}_k H_{h2}\Delta t}e^{-i\beta^{v1}_k H_{v1}\Delta t}e^{-i\beta^{h1}_k H_{h1}\Delta t}. \tag{3.11}$$

This method takes advantage of the additional separation of terms in the HVA seen in Eq. (2.23). Note that this further decomposition does not necessarily guarantee better performance than the single driver case, but still requires the measurement of more observables in the execution of the algorithm.

4. Results. In this section, we explore the performance of our feedback-based quantum algorithm for the Fermi-Hubbard model through numerical simulations at various lattice sizes. We examine results for both the single driver Hamiltonian case and the multiple control functions case, as described in Sec. 3.1 and 3.2 respectively.



(a) 1×2 lattice (4 qubits), $1\uparrow 1\downarrow$ fermion (b) 2×3 lattice (12 qubits), $3\uparrow 2\downarrow$ fermions

FIG. 4.1. *Simulations of our feedback-based ground state preparation method with two different lattice configurations, utilizing one parameter per layer, $\Delta t = 0.01$, and acting in the $U/\tau = 5$ regime. In the top plots, the dashed lines show the actual ground state energy the algorithm should be converging to. In the middle plots, there is a dashed line at 1 to provide a reference for the algorithm's performance.*

4.1. Single Control Function. We begin by considering the simplest interesting case of the Fermi-Hubbard model, i.e., a 2-site lattice. The results for this lattice configuration at half-filling are presented in Fig. 4.1(a) for $\Delta t = 0.01$ and $U/\tau = 5$. Using the same choice of Δt and U/τ, we show like results for a 2×3 lattice with just over half-filling in Fig. 4.1(b). In the top panels, we see the value of the objective function is monotonically decreasing with respect to layer, as expected given our choice of feedback law from Eq. (3.1). Each layer k evolves the system by U_p and $U_d(\beta_k)$, as given in Eqs. (3.4) and (3.7). The middle panels show that as the value of the objective function decreases, the fidelity of preparing the ground state approaches $|\langle gs|\psi\rangle|^2 = 1$, a property that should be the case with any successful ground state preparation algorithm. The bottom panels show the actual β parameter values used in the circuit, as obtained by the feedback law in Eq. (3.1).

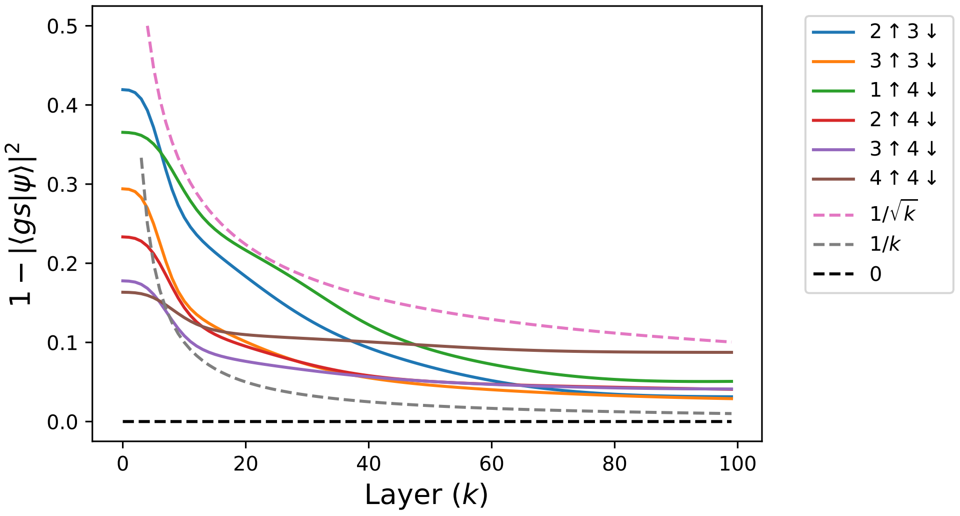


FIG. 4.2. *Comparing ground state infidelity for different numbers of up and down fermions for a 1×5 Fermi-Hubbard lattice (10 qubits), with $U/\tau = 5$ and $\Delta t = 0.01$.*

Fig. 4.2 visualizes the infidelity of the ground state with respect to layer for various fillings of a 1×5 lattice. Plots of $1/\sqrt{k}$ and $1/k$ are provided for reference, helping show the asymptotic performance and rate of convergence. We note that each of the different fillings exhibit slightly different rates of convergence, corresponding to differences in the underlying problem structures.

In addition to the choice of feedback law for setting the β_k parameters, this method also requires a choice for Δt. In most of our simulations, we found that $\Delta t = 0.01$ was sufficiently small for convergence to a state that decently overlaps the actual ground state. However, we also explored the algorithm's performance with varying timesteps. As shown

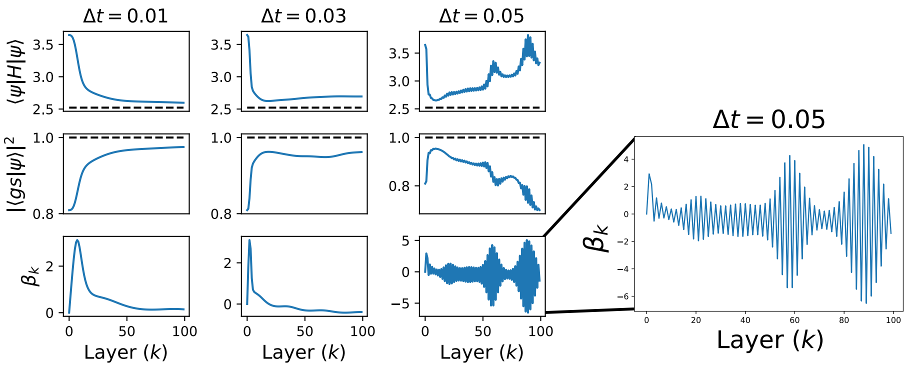
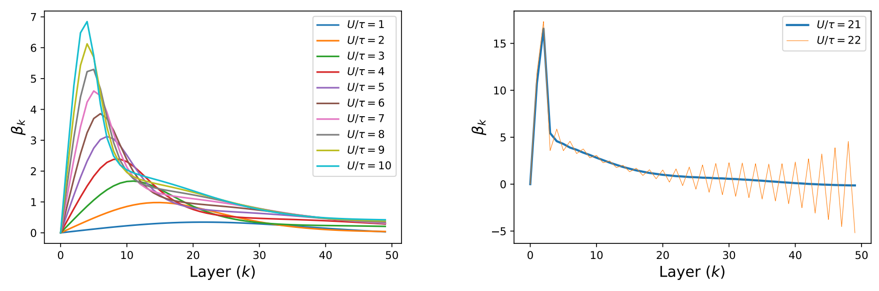


Fig. 4.3. *Exploring effect of timestep on feedback-based quantum algorithm for* 1×4 *lattice (8 qubits) with* $3\uparrow$ *and* $2\downarrow$ *fermions.*

in Fig. (4.3), the algorithm fails to converge when the timestep is too large. In this regime, the β_k parameters oscillate dramatically and the objective function fails to monotonically decrease. We remark that in practice, feedback-based quantum algorithms do not require objective function evaluations at each step, only measurement-based feedback to set each β_k value. As such, signs of oscillatory behavior in β_k should serve as the first warning that the timestep is too large and the algorithm is not converging.

We also explore how β_k parameter curves vary for different regimes of U/τ. Recall that our initial ground state corresponds to the ground state of our noninteracting Hamiltonian, T. Increasing U/τ would in some sense be providing more relative weight to the portion of the Hamiltonian not represented with our choice of initial state. To compensate for this, intuition suggests that a sharper increase in our β_k parameters should take place. This intuition is matched by the results of our numerical experimentation, as shown in Fig. 4.4(a). We see that as U/τ increases, the β_k parameter curves grow taller and thinner. We can see an oscillatory behavior in Fig. 4.4(b) similar to the oscillatory behavior of Fig. 4.3, showing that our timestep is likely too large. Hence, the steepness of the β_k-curves required for the $U/\tau \geq 22$ regime demands a timestep $\Delta t \leq 0.01$.

4.2. Multiple Control Functions. We now explore the formulation with multiple control functions, per Sec. 3.2. In 1D, only horizontal hopping terms are present. This yields two control functions, β_k^{h1} and β_k^{h2}, in general. In 2D, we consider a 2×3 lattice that additionally yields nonzero β_k^{v1}-parameters. The results of this simulation are shown in Fig. 4.5; since our vertical dimension is 2, the β_k^{v2}-parameters are zero at every layer. We remark that a 3×3 lattice would make all four parameters nonzero throughout the evolution and would be an interesting instance to explore in future work. We additionally note that in Fig. 4.5, $\beta_k^{h1} = \beta_k^{h2}$ at each layer.



(a) Variation of parameters with U/τ ratio (b) Regime requiring smaller timestep

FIG. 4.4. *Exploring effect of* U/τ *regime on* β *parameter values for* 1×4 *grid (8 qubits) with* $3 \uparrow 2 \downarrow$ *fermions and* $\Delta t = 0.01$.

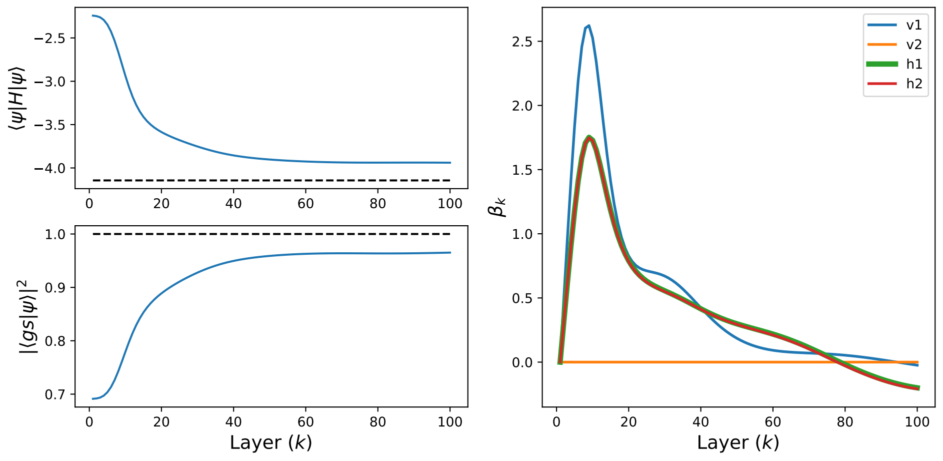


FIG. 4.5. *Simulation of feedback-based ground state preparation method for a* 2×3 *Fermi-Hubbard lattice (12 qubits) with* $3 \uparrow$ *fermions and* $2 \downarrow$ *fermions, utilizing multiple control functions,* $\Delta t = 0.01$*, and acting in the* $U/\tau = 5$ *regime.*

Even though there are additional parameters introduced for the multiple control functions setting, comparing its performance with the case of a single control function shows that these additional parameters do not guarantee improved performance. In fact, comparing the results for the same 2×3 lattice instance in Figs. 4.1(b) and 4.5, we see that a single control function performs slightly better. We believe this comes from the fact that Eq. (3.8) is overconstrained, i.e., in order for $\frac{dJ}{dt}$ to be nonpositive, there is no requirement that each of the terms in the sum in Eq. (3.8) are negative. Perhaps there is a better feedback law that would take this additional structure into account.

4.3. Algorithm Modifications. As shown in Fig. 4.6, we also implemented several improvements upon the base algorithm, inspired by the work in [25]. We tested an iterative method and added a reference field perturbation to see if our convergence could be improved.

For the iterative method, we first ran the algorithm through all ℓ layers with the standard feedback law. After acquiring the β_k parameters of each layer for the first iteration, we ran the algorithm a second time, but this time added the original β_k values to the new parameter values. This resulted in more extreme parameter values after each successive

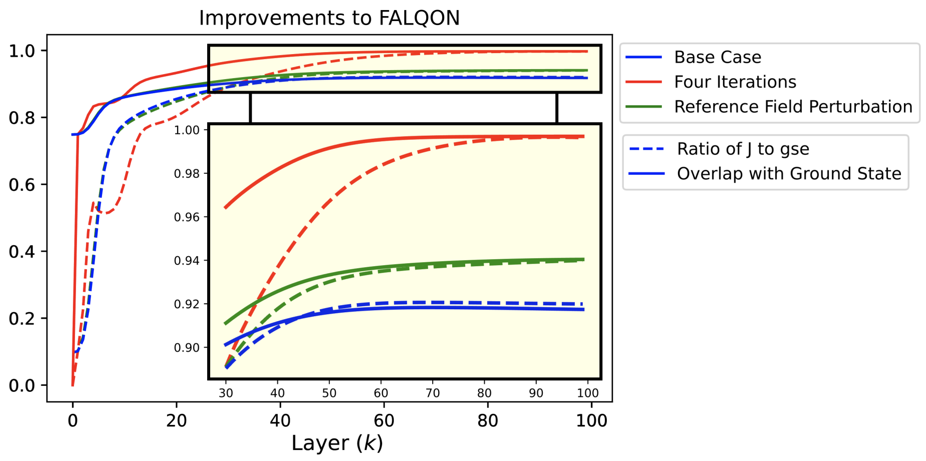


Fig. 4.6. *Simulation of modifications to base algorithm for a* 1×4 *Fermi-Hubbard lattice (8 qubits) with* $1 \uparrow$ *fermion and* $1 \downarrow$ *fermion,* $\Delta t = 0.01$*, in the* $U/\tau = 10$ *regime. The blue curves provide the base case performance using the standard feedback law. The green curves show the new convergence with a reference field perturbation added to the* β *parameter curves. The red curves show the new convergence after four iterations, summing over the* β *parameters at each iteration. The dashed lines show the overlap of the prepared state with the actual ground state,* $|\langle gs|\psi\rangle|^2$*, and the solid lines show the ratio of the objective function to the actual ground state energy. From this figure, we see that the iterative method provided the most significant improvement in this case, while the reference field perturbation did also positively impact convergence.*

iteration, summing over all previous parameter values. Fig. (4.6) only shows the result after the fourth iteration, summing over the previous three iterations of parameter values.

We also explored adding a reference field perturbation to our calculated β parameters to see if we could improve the performance of the algorithm. This practice is inspired by common practices in quantum control. For this exploration, we used a simple linear ramp going from 0.5 at the first layer down to 0 at the last layer, specifically $\beta'_k = \beta_k + 0.5 \cdot (1 - k/\ell)$. Both this modification and the iterative method resulted in slight improvements to convergence with nontrivial instances of the Fermi-Hubbard model.

5. Conclusion. In this work, we have introduced a novel method for ground state preparation of the Fermi-Hubbard model that utilizes an optimization-free feedback law inspired by quantum Lyapunov control. We have presented simulation results exploring its performance on different problem instances, and with enough layers, we seem to get consistently good convergence. The algorithm at times still seems to converge to suboptimal solutions (see Fig. 4.2), but we have tested strategies to modify the base algorithm to improve this performance (see Fig. 4.6) This method for ground state preparation has more general prospect beyond the NISQ era since it eliminates the need for the prohibitively large search spaces of variational quantum algorithms, shifting the burden of energy minimization to the quantum computer.

This work motivates a variety of further avenues for research. This feedback-based framework for preparing Fermi-Hubbard ground states could be implemented on actual NISQ hardware and the impact of the associated noise could be analyzed. This framework could also be used to seed other ground state preparation algorithms, such as VQE. There is also additional research to be done exploring how FALQON for the Fermi-Hubbard model scales with lattice-size or number of fermions, and how measurement cost of these algorithms compare with other algorithms.

Another potential direction for future study would be to adapt this framework for other Hamiltonian models, broadening the applicability of feedback-based quantum algorithms. Future work on feedback-based algorithms could also verify convergence properties of these methods, explore adaptations to fault-tolerant architectures, or apply FALQON to problems with nonlinear cost functions such as those found in quantum machine learning. This ansatz could also be compared to other methods for state preparation, benchmarking its performance against other common practices. It might be interesting to compare the performance of this feedback-based algorithm to quantum annealing, another method for ground state preparation without classical optimization.

REFERENCES

[1] D. P. Arovas, E. Berg, S. A. Kivelson, and S. Raghu, *The hubbard model*, Annual review of condensed matter physics, 13 (2022), pp. 239–274.

[2] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, *Quantum algorithms for quantum chemistry and quantum materials science*, Chemical Reviews, 120 (2020), pp. 12685–12717.

[3] A. Bohrdt, L. Homeier, C. Reinmoser, E. Demler, and F. Grusdt, *Exploration of doped quantum magnets with ultracold atoms*, Annals of Physics, 435 (2021), p. 168651.

[4] C. Cade, L. Mineh, A. Montanaro, and S. Stanisic, *Strategies for solving the fermi-hubbard model on near-term quantum computers*, Phys. Rev. B, 102 (2020), p. 235122.

[5] L. Clinton, T. Cubitt, B. Flynn, F. M. Gambetta, J. Klassen, A. Montanaro, S. Piddock, R. A. Santos, and E. Sheridan, *Towards near-term quantum simulation of materials*, arXiv:2205.15256, (2022).

[6] S. Cong and F. Meng, *A survey of quantum lyapunov control methods*, Sci. World J., 2013 (2013).

[7] M. Dobšíček, G. Johansson, V. Shumeiko, and G. Wendin, *Arbitrary accuracy iterative quantum phase estimation algorithm using a single ancillary qubit: A two-qubit benchmark*, Physical Review A, 76 (2007), p. 030306.

[8] A. Dusko, A. Delgado, A. Saraiva, and B. Koiller, *Adequacy of si: P chains as fermi–hubbard simulators*, npj Quantum Information, 4 (2018), pp. 1–5.

[9] E. Farhi, J. Goldstone, and S. Gutmann, *A Quantum Approximate Optimization Algorithm*, arXiv:1411.4028, (2014).

[10] N. Gomes, F. Zhang, N. F. Berthusen, C.-Z. Wang, K.-M. Ho, P. P. Orth, and Y. Yao, *Efficient step-merged quantum imaginary time evolution algorithm for quantum chemistry*, Journal of Chemical Theory and Computation, 16 (2020), pp. 6256–6266.

[11] S. Grivopoulos and B. Bamieh, *Lyapunov-based control of quantum systems*, in 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475), vol. 1, Dec 2003, pp. 434–438 Vol.1.

[12] T. Hensgens, T. Fujita, L. Janssen, X. Li, C. Van Diepen, C. Reichl, W. Wegscheider, S. D. Sarma, and L. M. Vandersypen, *Quantum simulation of a fermi–hubbard model using a semiconductor quantum dot array*, Nature, 548 (2017), pp. 70–73.

[13] J. Hubbard, *Electron correlations in narrow energy bands*, Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 276 (1963), pp. 238 – 257.

[14] J. A. Ivie, Q. Campbell, J. C. Koepke, M. I. Brickson, P. A. Schultz, R. P. Muller, A. M. Mounce, D. R. Ward, M. S. Carroll, E. Bussmann, et al., *Impact of incorporation kinetics on device fabrication with atomic precision*, Physical Review Applied, 16 (2021), p. 054037.

[15] Z. Jiang, K. J. Sung, K. Kechedzhi, V. N. Smelyanskiy, and S. Boixo, *Quantum algorithms to simulate many-body physics of correlated fermions*, Phys. Rev. Applied, 9 (2018), p. 044036.

[16] A. Y. Kitaev, *Quantum measurements and the abelian stabilizer problem*, arXiv preprint quant-ph/9511026, (1995).

[17] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, *Quantum simulation of electronic structure with linear depth and connectivity*, Physical Review Letters, 120 (2018).

[18] J. Koepsell, J. Vijayan, P. Sompet, F. Grusdt, T. A. Hilker, E. Demler, G. Salomon, I. Bloch, and C. Gross, *Imaging magnetic polarons in the doped fermi–hubbard model*, Nature, 572 (2019), pp. 358–362.

[19] N. H. Le, A. J. Fisher, N. J. Curson, and E. Ginossar, *Topological phases of a dimerized fermi–hubbard model for semiconductor nano-lattices*, npj Quantum Information, 6 (2020), pp. 1–10.

[20] N. H. Le, A. J. Fisher, and E. Ginossar, *Extended hubbard model for mesoscopic transport in donor arrays in silicon*, Physical Review B, 96 (2017), p. 245406.

[21] J. P. LeBlanc, A. E. Antipov, F. Becca, I. W. Bulik, G. K.-L. Chan, C.-M. Chung, Y. Deng, M. Ferrero, T. M. Henderson, C. A. Jiménez-Hoyos, et al., *Solutions of the two-dimensional hubbard model: Benchmarks and results from a wide range of numerical algorithms*, Physical Review X, 5 (2015), p. 041041.

[22] E. H. Lieb and F. Wu, *The one-dimensional hubbard model: a reminiscence*, Physica A: statistical mechanics and its applications, 321 (2003), pp. 1–27.

[23] A. Lucas, *Ising formulations of many NP problems*, Frontiers in Physics, 2 (2014).

[24] A. B. Magann, K. M. Rudinger, M. D. Grace, and M. Sarovar, *Feedback-based quantum optimization*, arXiv, (2021), p. 2103.08619.

[25] ———, *Lyapunov control-inspired strategies for quantum combinatorial optimization*, arXiv:2018.05945, (2021).

[26] A. Mazurenko, C. S. Chiu, G. Ji, M. F. Parsons, M. Kanász-Nagy, R. Schmidt, F. Grusdt, E. Demler, D. Greif, and M. Greiner, *A cold-atom fermi–hubbard antiferromagnet*, Nature, 545 (2017), pp. 462–466.

[27] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, *Quantum computational chemistry*, Reviews of Modern Physics, 92 (2020).

[28] S. McArdle, T. Jones, S. Endo, Y. Li, S. C. Benjamin, and X. Yuan, *Variational ansatz-based quantum simulation of imaginary time evolution*, npj Quantum Information, 5 (2019), pp. 1–6.

[29] A. A. Mele, G. B. Mbeng, G. E. Santoro, M. Collura, and P. Torta, *Avoiding barren plateaus via transferability of smooth solutions in hamiltonian variational ansatz*, arXiv:2206.01982, (2022).

[30] M. Motta, C. Sun, A. T. Tan, M. J. O'Rourke, E. Ye, A. J. Minnich, F. G. Brandão, and G. K.-L. Chan, *Determining eigenstates and thermal states on a quantum computer using quantum imaginary time evolution*, Nature Physics, 16 (2020), pp. 205–210.

[31] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, *A variational eigenvalue solver on a photonic quantum processor*, Nature communications, 5 (2014), pp. 1–7.

[32] J. Preskill, *Quantum computing in the NISQ era and beyond*, Quantum, 2 (2018), p. 79.

[33] M. Qin, T. Schäfer, S. Andergassen, P. Corboz, and E. Gull, *The hubbard model: A computational perspective*, Annual Review of Condensed Matter Physics, 13 (2022), pp. 275–302.

[34] C. S. Punla, https://orcid.org/ 0000-0002-1094-0018, cspunla@bpsu.edu.ph, R. C. Farro, https://orcid.org/0000-0002-3571-2716, rcfarro@bpsu.edu.ph, and Bataan Peninsula State University Dinalupihan, Bataan, Philippines, *Are we there yet?: An analysis of the competencies of BEED graduates of BPSU-DC*, International Multidisciplinary Research Journal, 4 (2022), pp. 50–59.

[35] J. Vijayan, P. Sompet, G. Salomon, J. Koepsell, S. Hirthe, A. Bohrdt, F. Grusdt, I. Bloch, and C. Gross, *Time-resolved observation of spin-charge deconfinement in fermionic hubbard chains*, Science, 367 (2020), pp. 186–189.

[36] D. Wecker, M. B. Hastings, and M. Troyer, *Progress towards practical quantum variational algorithms*, Physical Review A, 92 (2015).

[37] S. Yamada, T. Imamura, and M. Machida, *16.447 tflops and 159-billion-dimensional exact-diagonalization for trapped fermion-hubbard model on the earth simulator*, in SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, IEEE, 2005, pp. 44–44.

LEARNING QUANTUM MECHANICAL DENSITIES OF HELLMAN-FEYNMAN OPTIMIZED BASIS SETS USING EQUIVARIANT NEURAL NETWORKS

ALEXANDER R. MUNOZ*, SHIVESH PATHAK†, AND JOSH A. RACKERS‡

Abstract. The simulation of quantum mechanical properties has been a long-standing problem in the field of quantum chemistry. While traditional quantum chemistry techniques have found great success, they scale poorly with system size. Recent advances have been made using equivariant neural networks, allowing for the use of smaller system sizes to effectively simulate the properties of larger systems. In tandem with high accuracy basis sets designed to accurately simulate forces, Hellman-Feynman (HF) bases, our work is focused on obtaining high accuracy densities for the HF bases from equivariant neural networks. In this manuscript, we show that the form of the HF basis functions causes a significant difference in learning as compared to a well-established basis. By introducing a multipole expansion inspired scaling factor to the neural network architecture, we decreased the error in the HF basis' electron density by a factor of four.

1. Introduction.

The simulation of quantum mechanical systems has been an active field of research for over 70 years. The development of density functional theory (DFT) has been an exceptionally fruitful research program with advances made in the study of magnetism [2], topological systems [21], and molecular dynamics [15]. However, the scaling of these calculations with system size can make calculations of forces on large molecules prohibitively expensive [10]. Recently, new methods for circumventing the quantum scaling limit have emerged from the synthesis of physical concepts and machine learning [14].

While machine learning techniques have found success in this field, it is still problematic to create a sufficiently large training set given the scaling problem in DFT. To train some neural networks, it takes thousands of DFT samples to learn material phases and properties[14]. Equivariant neural networks are a particular form of neural network that exploit the symmetries of a data set to increase the efficiency of learning. For DFT, the most important symmetries to encode are the symmetries of 3D space, that are capture by a class of neural networks known as Euclidean neural networks [7].

A framework of Euclidean neural network, e3nn, has been used to learn DFT densities[14]. The ability to learn densities beyond the size of the training data will enable the use of these densities to efficiently compute the forces on the ions in the system, in spite of the quantum scaling limit. The computation of forces from densities can be done through the use of the Hellman-Feynman theorem[17]. The downside of this computation is that the Pulay force creates an error in which the Hellman-Feynman theorem is no longer exact [13].

In this study, we examine recently developed basis sets that minimize the Pulay force [16, 11]. Particularly, we contrast e3nn's efficacy in computing the density on a simple basis set, def2, with training on a Hellman-Feynman optimized basis set, with the density fitting basis generated automatically as in Stoychev, Auer, and Neese [20, 18]. By studying the training on both bases, we determine that the Hellman-Feynman optimized basis set's geometry causes an error in the density that makes it unsuitable for use in computing forces. We introduce a new layer to the neural network, inspired by the multipole expansion, that rescales the input data as a function of angular momentum. Other data normalization schemes have found success in machine learning, and with this physically motivated data normalization, we were able to reduce the density error on the Hellman-Feynman optimized basis set by a factor of four [9]. We then turn to other physically motivated training

*Department of Physics; University of Illinois at Urbana-Champaign, armunoz3@illinois.edu
†Sandia National Laboratories, sapatha@sandia.gov
‡Sandia National Laboratories,jracker@sandia.gov

techniques that will lead to further improvements, enabling the use of HF basis densities for the computation of forces.

2. Methods.

2.1. Density functional theory.

Quantum mechanical systems are described by the *ab initio* many-body Schrodinger equation. The Hamiltonian for the problem of interest here is,

$$\hat{H} = -\frac{1}{2}\sum_{i}^{N}\nabla_i^2 + \sum_{i\neq j}^{N}\frac{1}{r_{ij}} - \sum_{Ii}\frac{Z_I}{|r_I - r_i|} + \sum_{IJ}\frac{Z_I Z_J}{|r_I - r_J|}, \tag{2.1}$$

where each term represents the electron kinetic energy, the electron-electron potential, electron-ion potential and the ion-ion potential, respectively. Obtaining solutions to the Schrodinger equation is an exponentially hard problem with respect to the size of the system. Given a basis, the dimension of the Hilbert space grows combinatorially with the system size. As a consequence, the treatment of realistic solids and liquids often falls under the purview of density functional theory (DFT).

The framework of DFT provides a way of computing noninteracting single-particle orbitals. In the Kohn-Sham approach, they determine a noninteracting auxiliary potential such that the electronic density equals the electronic density for the interacting system. Kohn-Sham DFT circumvents the problem of directly solving the Schrodinger equation by computing the energy of a state from a density instead of the positions of each electron and ion in a system [10].

With a non-interacting system, we can write the solutions as Slater determinants built from single-particle orbitals,

$$E_i\phi_i(\vec{r}) = (-\frac{1}{2}\nabla^2 + v_{eff})\phi_i(\vec{r}) \tag{2.2}$$

where v_{eff} is the auxiliary potential, and ϕ_i is one of the single-particle orbitals. This equation is known as the Kohn-Sham equation. With v_{eff}, Kohn-Sham can be self-consistently solved for the full set of orbitals, ϕ_i. Using ϕ_i, a density is computed,

$$n(\vec{r}) = \sum_{i}^{N}|\phi_i(\vec{r})|^2. \tag{2.3}$$

Solving the Kohn-Sham equation for each orbital independently is far more tractable than the direct solution of the Schrodinger equation. However, the solution to the equation still scales like N_e^3 where N_e is the number of electrons in the system. While DFT's scaling is effective for many large systems, the use of DFT in generating starting points for large molecular dynamics simulations is often still prohibitively expensive, especially when considering the size and composition of an orbital basis.

In this work, we use Gaussian type Orbitals, GTOs [6]. These basis functions are written with the radial-angular decomposition,

$$\phi(\vec{r}) = R_l(r)Y_{lm}(\theta, \Phi) \tag{2.4}$$

where Y_{lm} is a spherical harmonic, l is the angular momentum, and m is the z-component of the angular momentum. The radial part of the function is written as,

$$R_l(r) = N(l, \alpha)r^l e^{-\alpha r^2} \tag{2.5}$$

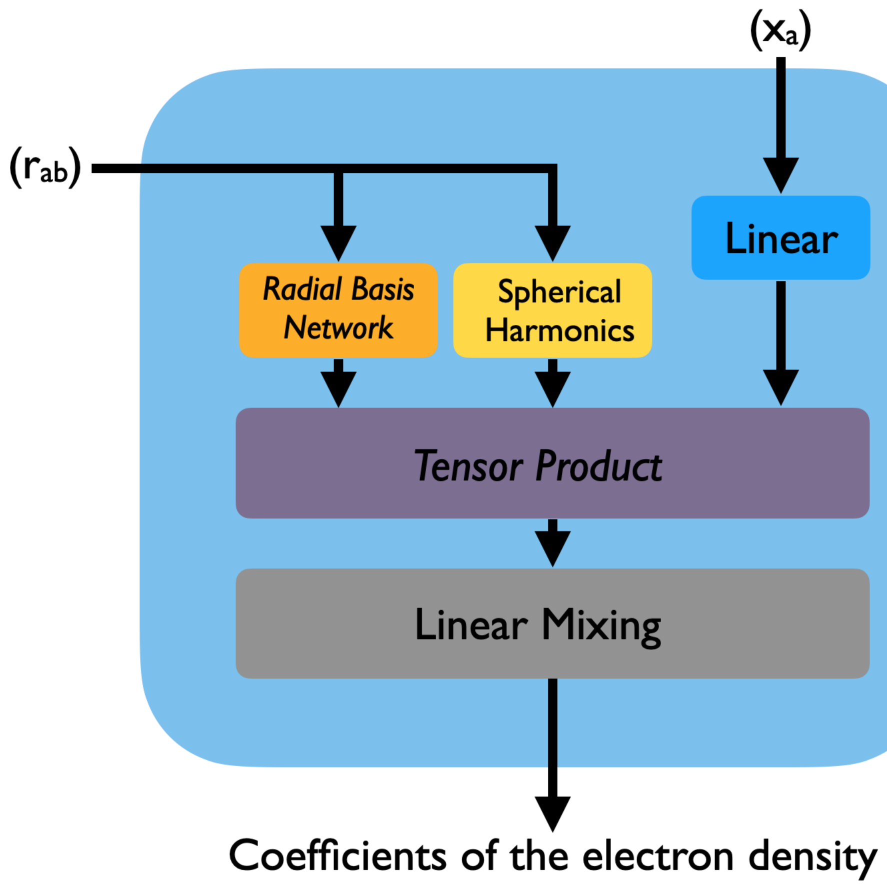


FIG. 2.1. *Internal construction of the convolutional layer in our model. Features r_{ab} and x_a are inputs to the network and represent the geometry of the graph and a one-hot encoding of the atom types, respectively. The majority of the trainable weights are in the italicized sections. The radial basis network is represented with a fully-connected multilayer perceptron. In practice, we use several convolutional layers connected with gated nonlinearities.*

where α is a parameter, and $N(l, \alpha)$ is the normalization determined by,

$$\int_0^\infty dr\ r^2\ |R_l(r)|^2 = 1. \tag{2.6}$$

These bases can be written in a general way with an arbitrary number of functions such that the density is written as,

$$n(r) = \sum_i^{N_{atoms}} \sum_k^{N_{basis}} \sum_l^{l_{max}} \sum_{m=-l}^{l} C_{iklm} Y_{lm} e^{-\alpha_{ikl}(r-r_i)^2}, \tag{2.7}$$

where C_{iklm} are the coefficients for each basis function [14]. For our purposes, we will focus on two bases, def2 and a Hellman-Feynman optimized basis [11], distinguished by the size of the bases and the schemes used to produce them [20, 18].

Hellman-Feynman basis sets are specifically designed for the calculation of the Hellman-

Feynman force,

$$F_I^E = -\frac{\partial E}{\partial \vec{R}_I} = -\frac{\partial \langle \Psi(\vec{R}_i)|\hat{H}|\Psi(\vec{R}_i)\rangle}{\partial \vec{R}_I} = \vec{F}_I^{HF} + \vec{F}_I^{Pulay}, \tag{2.8}$$

where, from the chain rule, F_I^{HF} is the derivative of the Hamiltonian with respect to the ionic coordinates and the Pulay term is the derivative of the wave function with respect to the ionic coordinates. In order for the Hellman-Feynman theorem to hold, the Pulay term must be suppressed. Hellman-Feynman bases are optimized to suppress the Pulay terms such that the HF force can be computed from the electron density. With a sufficiently accurate density for a large system, we should be able to compute the force with a similar accuracy for less computational cost than starting from DFT. If our goal is to obtain densities for large molecular dynamics calculations, DFT can still be restrictive, so we turn to the use of neural networks to circumvent the scaling problem of DFT [4].

2.2. Equivariant Neural Networks.

In learning the density of GTO DFT calculations, the goal is to build a model that accurately predicts C_{iklm} using a loss function like,

$$Loss_{density} = \frac{1}{n}\sum_{i}^{n}(C_i^{ML} - C_i^{target})^2 \tag{2.9}$$

where n is the number of functions in the basis set. Invariant machine learning models, models that operate only on scalars between layers, can be restrictive in how they learn data with underlying symmetries like those present in 3-dimensional geometry [1]. On the other hand, equivariant neural networks are symmetry-aware, meaning they can encode the rotation and translation of 3-dimensional geometric objects like C_{iklm}. Specifically, equivariance means that if an operation is performed on the inputs to a function, it is equivalent to applying the same operation to the outputs, of the function,

$$f(\Gamma(x)) = \Gamma(f(x)), \tag{2.10}$$

where x is the input, f is the function, and Γ is an arbitrary symmetry operation[7]. In this work we use a machine learning framework called Euclidean Neural Networks (e3nn). For cases with 3-dimensional data, e3nn has been shown to reduce the necessary training data by a factor of 1000 with respect to invariant models.

The e3nn framework constructs graph convolutional neural networks with nodes defined by the atoms in the system [7, 8]. For the convolution, shown in Figure 2.1, we input two parameters x_a and r_{ab}, a one-hot encoding of the atomic species and the geometry of the graph. The one-hot encoding gives each node in the graph an input irreducible representation. The hidden layer is then a gated convolution relying on the outputs of the preceding layer $x^{(i)}$. Practically, we use a series of convolutions akin to depth-wise convolutional layers to reduce the computation time [5].

The transformation of the convolution with a nonlinear gate between layers is

$$x_a^{(i+1)} = \sigma_{gated}(Lin_2(\sum_{b\in n(a)} C(Lin_1(x_b^{(i)}), \vec{r}_{ab}) + SC(x_a^i))) \tag{2.11}$$

where σ is an equivariant gated nonlinearity, SC is the self-connection mixing channels of the same irreducible representation, Lin_1 and Lin_2 are equivariant linear layers, $n(a)$ are the neighbors within a specified radius of a, $\vec{r}_{ab}$ is the vector between atom b and the convolution center, and C is the convolution tensor product operation [14]. The convolution tensor product operation is core to e3nn's functionality, as it permits all tensor operations that yield the desired irreducible representations set as the output of the network by the user. The convolutional tensor product is defined as

$$C(x_b, \vec{r}_{ab}) = x_b \otimes R(|r_{ab}|)Y(\vec{r}_{ab}) \tag{2.12}$$

where Y is a set of spherical harmonics up to a certain l, and R is the radial basis function. The radial basis function contains most of the learnable parameters in the convolutional layer and is constructed as a simple multilayer perceptron (MLP) defined as

$$R(|r_{ab}|) = W_2(\sigma_{MLP}(W_1(B(|r_{ab}|)))) \tag{2.13}$$

where the $W_{1,2}$ are the weights of the MLP layers, σ is the sigmoid nonlinearity, and B are Gaussian radial basis functions. In total, this operation brings us to the form of the DFT basis functions we are attempting to learn.

3. Results.

3.1. DFT calculations.

For the experiments in this paper, we use Psi4 and the PBE0 density functional to produce densities on 100 geometries of three water molecules [12, 19]. These calculations were performed on the conventional def2 basis and the Hellman-Feynman optimized basis. In addition, we performed calculations on isolated atoms that were be subtracted away from the molecular clusters to normalize the data for learning. For the purposes of validating our dataset, we use the conventional split of 85-15 for training and testing, respectively [3].

3.2. Neural Network Architecture. The network used for our experiments is composed of three convolutional layers combined depth-wise with the radial basis function being represented by a single-layer fully connected network [5]. The internal tensor product uses spherical harmonics up to l=5. The hidden layer features in our network admit functions with a maximum angular momentum of four. We use 125 functions for l=0, 45 functions for l=1, 20 functions for l=2, 15 functions for l=3, and 7 functions for l=4. The use of the higher angular momentum functions increases the data efficiency of the network. To train the network, we utilize the Adam optimizer built into PyTorch. With an initial learning rate 0.01, we found this set of parameters optimally trains on both the def2 and Hellman-Feynman bases.

3.3. Learning on the def2 basis.

As an example, we demonstrate the effectiveness of our network architecture in learning the densities of def2 DFT calculations in Figure 3.1. Both the training and test errors on the DFT coefficients reach total losses of 10^{-6} before 200 epochs have passed. The training and test losses also appear to be converging at similar rates, indicating that the network is not overfitting to the training data. The percent density error on the test set, ϵ_{test} is computed as,

$$\epsilon_{test} = 100 \times \frac{\int d\vec{r}|\rho_{QM}(\vec{r}) - \rho_{ML}(\vec{r})|}{\int d\vec{r}\rho_{QM}(\vec{r})} \tag{3.1}$$

where the integrals are performed over the entire simulation cell. In def2's case, we observe that ϵ_{test} can be trained to below one percent and even arbitrarily lower with a higher number of epochs. For purposes like computing forces, this training is ideal as the density error should be well-correlated with the force errors.

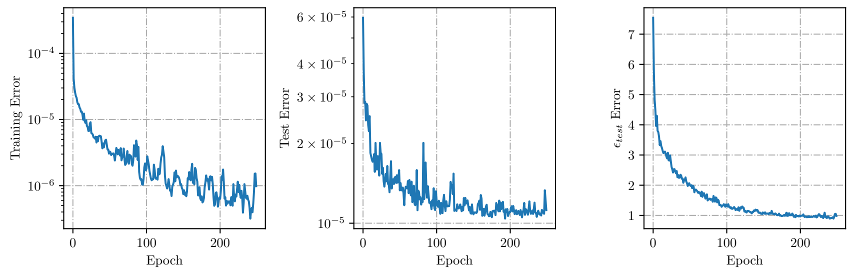


FIG. 3.1. *Example of training on the def2 basis. The training set contained 85 geometries of three water molecules and the test set contained another 15 geometries. The difference between the training error and test error is due to a residual amount of overfitting that is systematically controlled by the learning rate and size of the batch used for training. By training on this data set, we were able to compute the density error, ϵ, on the test set. The percent error with respect to the DFT reference density is converged to less than one percent, our target density error.*

3.4. Learning on the Hellman-Feynman basis.

In contrast to the ideal training shown in Figure 3.1, we tracked the training of an identical neural network on the Hellman-Feynman optimized basis set, as shown in Figure 3.2. Unlike training on def2, the training error nearly converges around 10^{-5}. The growing oscillation at more than 100 epochs is controllable with batch training, but due to the small dataset used in this study, we used an optimized batch size of three during training. Likewise, the test error only reaches 10^{-3} even after 250 epochs, although the network reaches this level after only 50 epochs. While the training error continues to decrease with each epoch, the test error plateaus. After 150 epochs, the network begins learning on the test data at a very slow rate that is infeasible for larger systems.

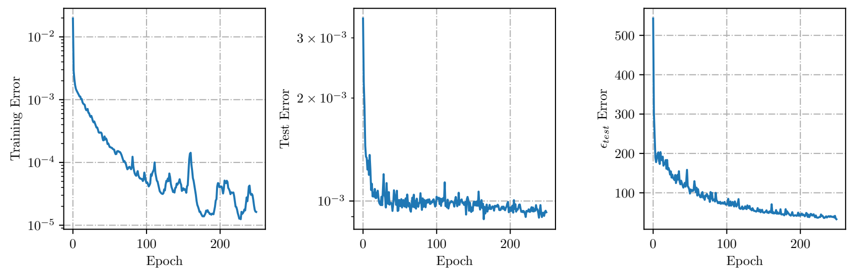


FIG. 3.2. *Initial training on an HF optimized basis. For training and testing, we used the same 85-15 split in geometries that was used for def2. In comparison to def2, the errors are higher by orders of magnitude, and are not converging to the same level with the same number of epochs. Particularly, we observe an initial density error of over 500 percent that converges to 30 percent after 250 epochs. The converged 30 percent error is persistent across many random initializations of our network architecture, and across hyperparameter sweeps.*

In Figure 3.2, we plotted the ϵ_{test} error for the Hellman-Feynman optimized basis set. Strikingly, the initial error on the density is two orders of magnitude larger than the initial density error on def2, starting around 525 percent. Inaccurate initial guesses can cause the optimization algorithm to become caught in local minima that may not generalize to the test data set [3]. While the density error does fall rapidly with the first few epochs, the density error begins to plateau with the test error. As a consequence, the error on the density reaches 35 percent after 250 epochs. With further iterations, the density error converges around 30 percent regardless of the seed given to the randomized initial weights. It is important then, to turn to the differences in the basis sets to determine why the Hellman-Feynman basis initializes poorly and suffers from poor training.

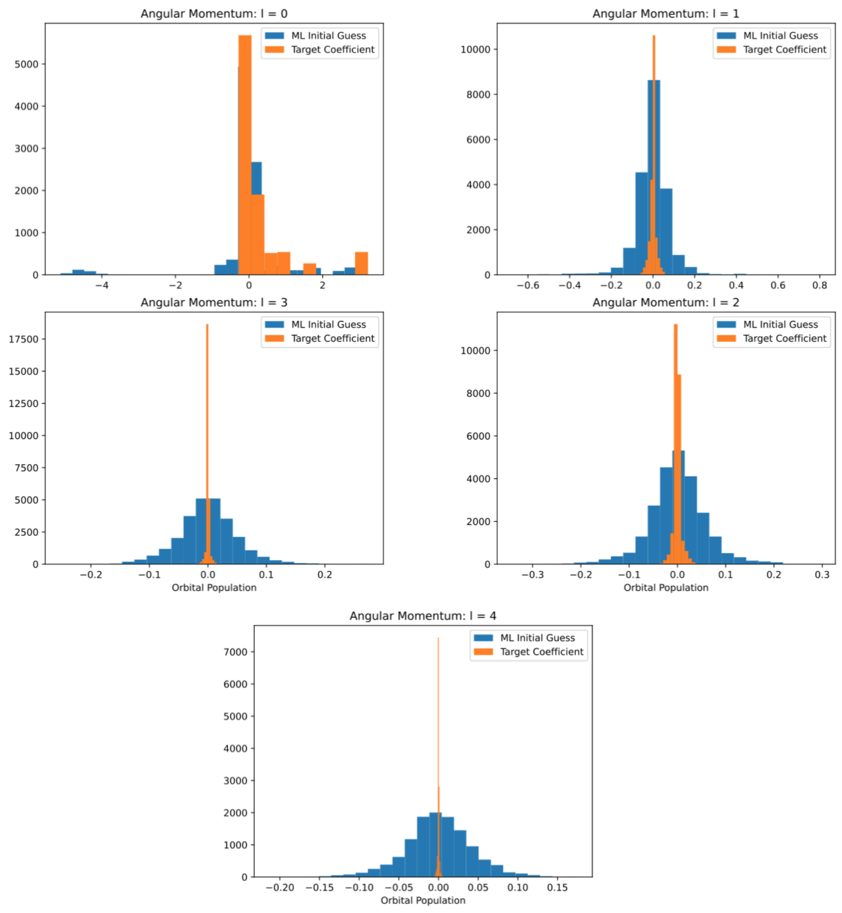


Fig. 3.3. *Distribution of the DFT target orbital populations for the HF basis in comparison to the distribution of the neural network initial guesses for the orbital populations. As the angular momentum is increased, we observe that the network's initialization has an increasingly high variance with respect to the target distribution variance. Large differences between the distributions at high angular momentum lead to larger density errors than differences at low angular momentum. The initial errors in the density are then tied directly to the initialization of the neural network at high angular momentum.*

3.5. Distributions of Initial Guesses.

On initialization, it is clear from the ϵ_{test} error that the machine learning initial coefficients are not close to the initial distribution of the DFT target coefficients. In Figure 3.3, we show histograms with the distributions of the machine learning coefficients after one epoch in comparison to the target coefficients as a function of the angular momentum channel. As we progress higher in angular momentum, the target populations have increasingly small variances. However, the initialization of our network does not inherently alter the variance of the model with angular momentum. The mismatch in orbital population distributions

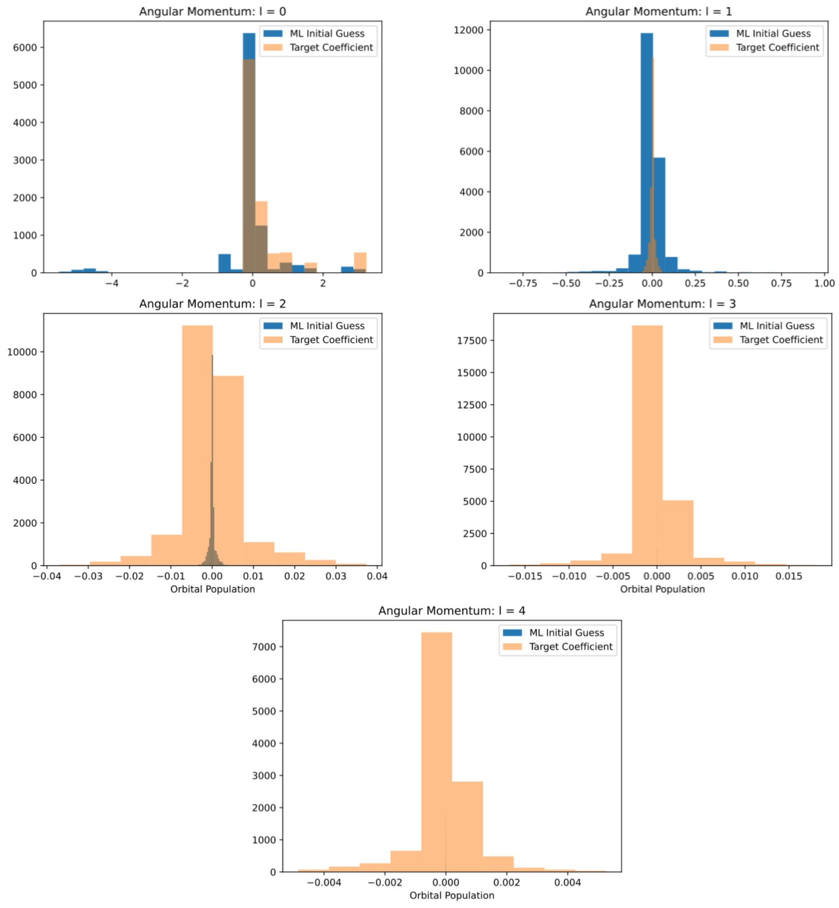


FIG. 3.4. *Distribution of the DFT target orbital populations in comparison to the neural network initialized orbital populations after rescaling the outputs for the HF basis in accordance with the multipole expansion. By applying successively smaller prefactors to the angular momentum, the neural network initialization of the orbital populations is brought within an order of magnitude of the target populations. The initialization can be further improved by tuning the prefactors; however, the architecture performs similarly in reducing the error on the density as long as the initialization is reasonable.*

leads directly to a large error in the density. The density from an orbital is computed as,

$$\rho_i = P_i \times \mathcal{N}_i \tag{3.2}$$

where P$_i$ is the population of electrons in orbital i, and N$_i$ is the normalization of the orbital. Small differences in the population of an orbital can lead to large differences in the density if the normalization of the orbital is large. This is of particular importance when considering the GTOs of the Hellman-Feynman optimized basis set at high angular momentum which have both high angular momentum and larger exponents than the def2 basis. Both of these factors lead to a larger normalization as we can tell from the radial part of the GTO.

3.6. Scaling coefficients with respect to the Multipole Expansion. In order to resolve the initialization issue, we augmented the neural network to include a rescaling of the machine learning orbital populations in each epoch. For water molecules, we expect that the density is almost entirely contained in the s-orbitals, as seen in Figure

3.3, and the total density should converge in accordance with the multipole expansion,

$$\rho(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} C_l^m Y_l^m(\theta, \phi) \tag{3.3}$$

where C_l^m are the coefficients and Y_l^m are the spherical harmonics. Given the diminishing target populations at high angular momentum, we scale the machine learning populations as a function of angular momentum. Starting from one we decrease the scaling factor by a factor of ten for each angular momentum channel. The new machine learned coefficients after a single epoch of learning are shown in Figure 3.4. At high angular momentum, we have removed the overestimates of the target populations by factors of 10 or 100. The initial density error is also reduced by a factor of ten as shown in the first epoch of Figure 3.5.

We show the effect the rescaling has on the training in Figure 3.5. The test loss now maintains a steady decrease from epoch 100 to epoch 250, but we maintain the oscillatory behavior in the training loss. However, the oscillations can be systematically controlled with batch training, and learning rate optimization. The ϵ_{test} graph shows some initial fluctuations in training, but has a near-monotonic behavior after 20 epochs. Eventually, the training reduces the error on the density to seven percent with indications it could continue.

While the multipole rescaling has improved the training on the Hellman-Feynman optimized basis by a factor of four, it is an insufficient improvement for extension to larger datasets. To this point, however, further optimization can be performed in our learning procedure. This includes learning rate hyperparameter sweeps on an angular momentum dependent basis, allowing us to potentially learn the high population orbitals to the precision necessary to push the density error below one percent.

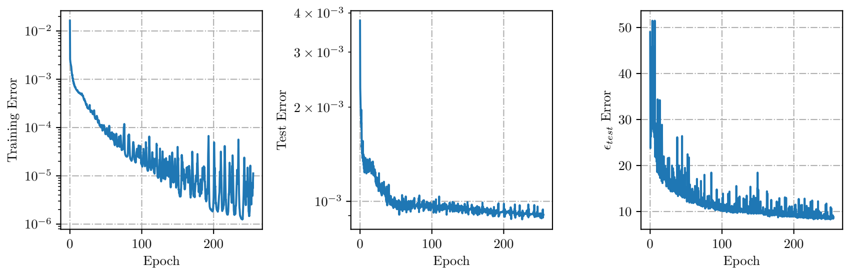


FIG. 3.5. *Training on the HF basis with the scaling procedure applied within the neural network. In comparison to the density error without the scaling, we find the initial state has a density error an order of magnitude smaller. With this improvement, the reaches density error is a factor of four smaller than the converged density error without scaling, reaching an eight percent total density error. Inspecting both the test error and the density error, they do not appear to have converged after 250 epochs, but the rate of training is impractical for density improvements to the level necessary for computing forces.*

4. Conclusions. The problem of extending *ab initio* quantum mechanical calculations to larger systems has been long-standing. Advances in equivariant neural networks like e3nn have reduced the cost of training densities for large systems with the ability to reach one percent error with traditional basis sets like def2. An attractive prospect remains in the ability to extend the use of equivariant neural networks to the computation of forces using the Hellman-Feynman theorem and basis sets optimized to remove the Pulay force. In this manuscript, we have improved the machine learned density estimates from Hellman-Feynman basis sets by a factor of four by introducing a multipole expansion inspired scaling to the neural network architecture. However, these errors still need to be reduced to be viable for the computation of forces. Further work in correcting the machine learned density continues through investigations into the ideal learning rates for the different angular momentum channels, and applying an angular momentum dependent learning to the appropriate functions.

REFERENCES

[1] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, *E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials*, Nature Communications, 13 (2022), p. 2453.

[2] G. Bihlmayer, *Density Functional Theory for Magnetism and Magnetic Anisotropy*, Springer International Publishing, Cham, 2018, pp. 1–23.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Berlin, Heidelberg, 2006.

[4] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, *Bypassing the kohn-sham equations with machine learning*, Nature Communications, 8 (2017), p. 872.

[5] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807.

[6] J. Fernández Rico, R. López, I. Ema, and G. Ramírez, *Accuracy of the electrostatic theorem for high-quality slater and gaussian basis sets*, International Journal of Quantum Chemistry, 100 (2004), pp. 221–230.

[7] M. Geiger and T. Smidt, *e3nn: Euclidean neural networks*, 2022.

[8] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, *Neural message passing for quantum chemistry*, 2017.

[9] J. Jun-Mo, *Effectiveness of normalization pre-processing of big data to the machine learning performance*, The Journal of the Korea institute of electronic communication sciences, 14 (2019), pp. 547–552.

[10] W. Kohn, *Density functional and density matrix method scaling linearly with the number of atoms*, Phys. Rev. Lett., 76 (1996), pp. 3168–3171.

[11] S. Pathak, J. A. Rackers, I. E. López, R. L. Fernández, A. J. Lee, W. P. Bricker, and S. Lehtola, *Accurate hellmann-feynman forces with optimized atom-centered gaussian basis sets*, 2022.

[12] J. P. Perdew, M. Ernzerhof, and K. Burke, *Rationale for mixing exact exchange with density functional approximations*, The Journal of Chemical Physics, 105 (1996), pp. 9982–9985.

[13] P. Pulay, *Ab initio calculation of force constants and equilibrium geometries in polyatomic molecules. I. Theory*, Molecular Physics, 17 (1969), pp. 197–204.

[14] J. A. Rackers, L. Tecot, M. Geiger, and T. E. Smidt, *Cracking the quantum scaling limit with machine learned electron densities*, 2022.

[15] P. C. Rathi, R. F. Ludlow, and M. L. Verdonk, *Practical high-quality electrostatic potential surfaces for drug discovery using a graph-convolutional deep neural network*, Journal of Medicinal Chemistry, 63 (2020), pp. 8778–8790.

[16] J. F. Rico, R. López, I. Ema, and G. Ramírez, *Generation of basis sets with high degree of fulfillment of the hellmann-feynman theorem.*, J Comput Chem, 28 (2007), pp. 748–758.

[17] L. Salem and E. B. Wilson, *Reliability of the hellmann—feynman theorem for approximate charge densities*, The Journal of Chemical Physics, 36 (1962), pp. 3421–3427.

[18] G. L. Stoychev, A. A. Auer, and F. Neese, *Automatic generation of auxiliary basis sets*, Journal of Chemical Theory and Computation, 13 (2017), pp. 554–562.

[19] J. M. Turney, A. C. Simmonett, R. M. Parrish, E. G. Hohenstein, F. A. Evangelista, J. T. Fermann, B. J. Mintz, L. A. Burns, J. J. Wilke, M. L. Abrams, N. J. Russ, M. L. Leininger, C. L. Janssen, E. T. Seidl, W. D. Allen, H. F. Schaefer, R. A. King, E. F. Valeev, C. D. Sherrill, and T. D. Crawford, *Psi4: an open-source ab initio electronic structure program*, WIREs Computational Molecular Science, 2 (2012), pp. 556–565.

[20] F. Weigend, *Hartree–fock exchange fitting basis sets for h to rn †*, Journal of Computational Chemistry, 29 (2008), pp. 167–175.

[21] R. Zhang, B. Singh, C. Lane, J. Kidd, Y. Zhang, B. Barbiellini, R. S. Markiewicz, A. Bansil, and J. Sun, *Critical role of magnetic moments in heavy-fermion materials: Revisiting* smb$_6$, Phys. Rev. B, 105 (2022), p. 195134.

A CHARACTER-BASED APPROACH TO DUPLICATE IDENTIFICATION

KILLIAN MUOLLO[*], ELAINE M. RAYBOURN[†], V. GREGORY WEIRS[‡], REED MILEWICZ[§], JEFFREY A. MAULDIN[¶], AND THOMAS J. OTAHAL[‖]

Abstract. Noisy, unstructured data — particularly natural language data — are abundantly available, yet inherently difficult to parse, organize, and analyze. Identifying duplicate information is crucial to compiling and disseminating information, but the process of doing so can be tedious and often requires a significant degree of human processing. However, natural language processing techniques can be used to minimize the amount of manual oversight needed for duplicate identification within unstructured textual data. Using a dataset consisting of 310 phrases of varying lengths pulled from automatically-generated video transcripts, we apply a character-based approach to measuring semantic similarity. We compute pairwise normalized Levenshtein distances, then identify groups of duplicate statements by generating a graph from these metrics and isolating all connected subgraphs. We find that a character-based methodology helps account for irregularities in the data, and could potentially be applied to similar natural language data with great success.

1. Introduction. An important component of understanding natural language data of any form is the ability to identify and disambiguate similar pieces of information. In doing so, developers can track bugs that spread unwittingly with the cutting and pasting of existing code into new products, and community forums can detect potential for plagiarism and develop usage statistics [6]. Duplicate information, and by extension its identification, is also key to the compilation and dissemination of knowledge. For example, Bettenburg et al. [1] found that software developers actually tend to find duplicate issue trackers helpful — being able to read about the same problem from slightly different perspectives often helped them build a more robust understanding of the situation. A wealth of knowledge can similarly be found in videos, especially those regarding software documentation and walkthroughs [7]. Automatically-generated transcription services provided by companies like Zoom, YouTube, and Microsoft Teams make it easier to apply duplicate-identification techniques to such sources for the purpose of aggregating related information, over transcribing by hand. However, the manner in which the natural language is encoded (e.g. how prosody or the patterns of stress, intonation, or pauses are handled in the text transcription) can also introduce errors. Some human oversight is still necessary.

The manual process of picking out and sorting duplicates can be tedious and time-consuming. However, automating this process is possible with the aid of existing natural language processing techniques. Ellmann [2] classifies automated approaches to duplicate identification into five categories of analysis: characteristic, syntactic, semantic, classification, and prediction. Characteristic analysis looks at the qualities of a document outside its actual textual content, like ID numbers and creation and closing timestamps; syntactic and semantic analyses focus on the words used in a document and their relation to each other; and classification and prediction center on feature extraction, often facilitated by some sort of machine learning tool or algorithm.

Existing literature on duplicate identification in technical fields specifically discusses the issue of human error and variation in text generation. While attempting to pick out duplicate bug reports, Sureka et al. [8] found typos, shorthand, punctuation, and other natural language nuances to vary wildly across users; similarly, Wang et al. [9] noticed

[*]University of Central Florida, kkmuollo@knights.ucf.edu
[†]Sandia National Laboratories, emraybo@sandia.gov
[‡]Sandia National Laboratories, vgweirs@sandia.gov
[§]Sandia National Laboratories, rmilewi@sandia.gov
[¶]Sandia National Laboratories, jamauld@sandia.gov
[‖]Sandia National Laboratories, tjotaha@sandia.gov

that developers used a great deal of paraphrasing when describing duplicate tasks. In order to address this issue, Ellmann [2] suggests following the approach of researchers who use character-based approaches to syntactic and semantic comparison. For example, Sureka et al. used a character n-gram (as opposed to word n-gram) algorithm to look at portions of words for comparison, leading them to correctly identify parts of text like "Add Java exception" and "AddExceptionDialog" as having a similar meaning.

In the following sections of this paper, we present our interpretation of a character-based approach to duplicate identification. We utilize the Levenshtein distance as a character-based measure of string similarity, and then use a graphical representation of the generated relationships between datapoints to isolate groups of duplicate text. We measure the accuracy of this technique by comparing our results to a manual classification of our entire dataset. Finally, we discuss the limitations of our investigation and potential expansions of our research.

2. Methodology.

2.1. Dataset.

For the present effort, we used a test dataset consisting of 310 natural language entries. Each entry was created by one of six individuals (henceforth referred to as "coders") whose goal was to independently identify salient phrases from recorded walkthroughs of two different software use cases. The walkthroughs encompassed three separate videos, lasting between 60 and 90 minutes each, with synced audio and video, as well as text transcripts generated automatically by Microsoft Teams. The phrases identified by the six coders were sometimes cut directly from the transcripts, sometimes cut from the transcripts and then edited based on the audio, and sometimes pulled by ear from just the audio. This resulted in a great deal of variation in formatting, length, human errors like typos, and the inclusion of stopwords like "uh" or "um." While each selected phrase was also labeled by the coders with a timestamp, there was a similar level of variation in how these were entered; like with the phrases themselves, some coders pulled their timestamps from the transcripts, while others identified them by matching the audio with the scrub bar on the video. Included below is an example of the same phrase pulled from the same video, but recorded differently by two separate coders:

	Timestamp	**Phrase**
Coder 3	0:38:57	So all I'm getting out of this are the images, there is no other burden on the file system?
Coder 5	00:38:55.770 - -> 00:39:11.480	So all that I'm getting out of this is the images, right? There's no other. Burden on the file system. Yeah. Right, right.

Our goal was to identify and group all duplicate phrases that refer to the same point in the same video walkthrough, but had been selected and entered by at least two different coders.

2.2. Identifying Duplicates.

In order to avoid incorrectly identifying "duplicates" that actually occur in different videos, we started by splitting all phrases into three groups based on which walkthrough they were extracted from. Therefore, each phrase was only compared to phrases that originate from the same video. The only natural language preprocessing we applied to these phrases is conversion to lower case; common practices like removing punctuation and lemmatizing words are forgone for the sake of preserving inherent syntactic information. Finally, for each of the 310 phrases in our dataset, we limited our search for duplicates to a small time frame nearby. Since there was some significant variation in the timestamping of phrases, we made this window twice as large as the longest identified phrase, which was recorded as lasting 70 seconds. So, as an example using timestamps formatted as HH:MM:SS, we would have only checked for duplicates of a phrase with a starting timestamp of 00:38:55 and an ending timestamp of 00:39:11 between the times 00:36:35 and 00:41:31.

Like Sureka et al. [8], we wanted to use a character-based similarity metric to compare phrases. One such well-known measure of string similarity is the Levenshtein distance. Given two sequences of characters pulled from a finite alphabet, their Levenshtein distance is calculated as the minimum number of insertions, deletions, and substitutions required to match them exactly. Since the length of phrases in our dataset was not consistent, we could not use this metric without first normalizing it, so we turned instead to the normalized Levenshtein distance metric introduced by Yujian et al. [10]. This metric outputs values between zero and one, with measures closer to one corresponding to lower Levenshtein distances and, thus, stronger string similarity.

Using the Python package `Levenshtein` [3], we computed pairwise normalized Levenshtein distances between all phrases that fell within the aforementioned 140-second window of each other. For the same reason we did not lemmatize or remove punctuation, we calculated the Levenshtein distance without sorting words or characters beforehand. Phrases that were measured to have a partial Levenshtein distance greater than or equal to 0.9 were considered duplicates of each other. This threshold was selected through a trial and error process. As would be expected, higher thresholds resulted in more statements that were actually duplicates being overlooked by our tool; for example, increasing the threshold from 0.9 to 0.95 resulted in 52.94% more of these false negatives. Lowering the threshold too far also presented problems, albeit in a less obvious way. When lowering the threshold from 0.9 to 0.85, only one additional statement was falsely identified as a duplicate. However, as a result of the duplicate grouping method described in the next section, there were four instances wherein our tool falsely merged distinct sets of duplicates.

2.3. Grouping Duplicates. With the Levenshtein calculations, we obtained a list for each phrase of its duplicates, but encountered an issue with false negatives. If there was a group of more than two true duplicates, our method often linked them in a chain to each other, but did not identify all pairwise connections. For example, consider four true duplicates, phrases A, B, C, and D. Phrase A may have both B and C in its duplicate list, B may have just A, and C may have A and D. With human oversight, we could have followed this connection from A to C to D using the transitive property, i.e. if A=C and C=D, we must also have A=D. In terms of duplicate identification, we conclude that if statement C is a duplicate of statement A, and statement D is a duplicate of statement C, then statement D must also be a duplicate of statement A. In order to account for this, we constructed a graph in which each phrase in our dataset represented a single node. We created edges between two nodes if those phrases were identified as duplicates using the method described above. Then, we selected the final grouping of duplicates as the connected subgraphs of this graph. Figure 2.1 illustrates an example of this. Here, phrases A, B, C, and D would all be identified as duplicate group 1; E, F, and G would be duplicate group 2; H would not be a duplicate at all; and I and J would be duplicate group 3.

3. Results. For the purpose of measuring accuracy, we reviewed the grouped phrases manually and labeled them according to one of four categories: false positive (FP), false negative (FN), true positive/negative (TPN), and grouped duplicate (GD). The first three labels are intuitive, with the label "FP" indicating that a phrase was *not* actually a duplicate of another phrase, but was labeled as such by our tool; "FN" indicating that a phrase *was* a true duplicate but was not identified as one; and "TPN" indicating that our tool correctly identified the phrase as a duplicate or not a duplicate. The last label, "GD," was created to account for a particular idiosyncrasy of our dataset that was revealed through

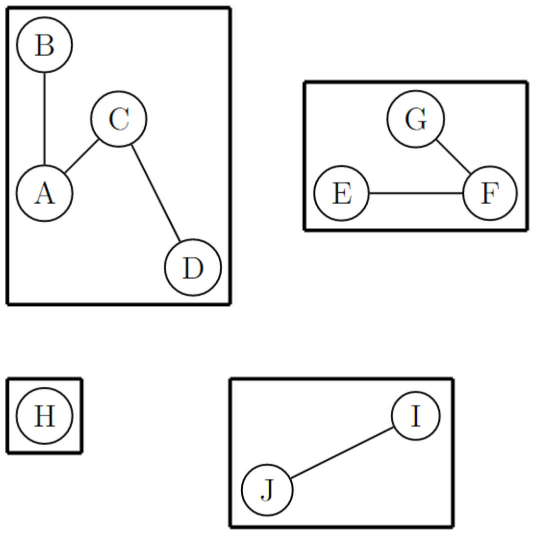


Fig. 2.1. *Toy example of using graphs to identify groups of duplicate statements: Each individual statement is represented by a node, nodes are connected with an edge if the statements they represent are identified as duplicates based on character similarity, and duplicate groupings are established by connected subgraphs*

the grouping process.

As mentioned in our description of the dataset, the length of each phrase varied greatly, and contiguous sentences were entered by some coders as a single phrase, but as multiple phrases by others. For example, the transcript excerpt, "It seems...pretty usable at the moment. I'm sure there's some nuance that I won't really understand until I start using it" was split in half by some coders, but left whole by others. As a result of the subgraph grouping we applied, this also meant that the two sentences, even when entered separately, were identified as duplicates because of their connection to the larger phrase. Some applications of duplicate identification that require a stricter definition of a duplicate might consider such a grouping a failure. However, for many research questions involving duplicate identification, units of text falling under this category of "grouped duplicates" still provide a great deal of information and minimize the need for human intervention in data processing. In our test case, having these duplicate groups allowed us to easily identify what needed additional human oversight, thus facilitating an efficient expenditure of our resources. We were able to able to focus on a smaller subset of the data — less than a quarter — for more detailed analysis, which may not have fit the exact definition of duplicate identification but certainly provided a helpful and meaningful transformation of otherwise unstructured textual data. Although we separated statements like this into their own group to distinguish them from exact duplicates that required no additional processing, we considered these GDs to be successful duplicate identifications for the goal of minimizing human oversight. With this standard in mind, we were able to correctly group 93.55% of phrases with their appropriate duplicates. More specifically, as visualized in Figure 3.1, we ended up labeling 3 phrases as FPs (0.97%), 17 as FNs (5.48%), 71 as GDs (22.90%), and 219 as TPNs (70.65%).

4. Conclusions and Future Work. The work presented in this paper attempts to identify duplicates in a rather literal sense, and this allows us to see our findings very clearly and extrapolate them in support of similar conclusions in the current literature.

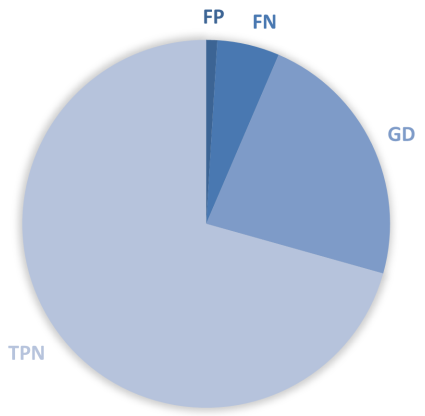


Fig. 3.1. *Percent of phrases labeled as false positive (FP), false negative (FN), grouped duplicate (GD), and true positive/negative (TPN)*

Perhaps most obviously, we can reiterate the importance of context to identifying a duplicate. Before calculating the Levenshtein similarity between phrases, we first had to separate the videos and impose limited timeframes in which to search for duplicates. Just as Sureka et al. [8] found increased accuracy by limiting their search for duplicate issue trackers to specific projects, we see here a need to artificially differentiate the separate walkthroughs before applying our tool. Without taking such precautions, we found more false positives. For example, when removing just the timeframe restriction but still separating the videos, we idenitifed sixteen false positives instead of three — an increase of more than 400%. Manual inspection revealed that simple comments like "yeah" and "right," which appeared often throughout the transcripts but were spoken by different individuals at different times, conflated duplicate detection. We can thus see that such unspecific language relies on the context in which it is given for meaning and can impede duplicate identification if considered in isolation.

The high accuracy of our method, despite the differences in how coders gathered and entered their data, supports another conclusion drawn by Sureka et al. [8]: that a character-based similarity measure accounts for human variation better than a word-tokenizing method might. Using the Levenshtein distance allowed us to successfully connect words and phrases that sounded the same when spoken, but were recorded very differently in text, like "par-aview" versus "pair of view." Typos similarly had little effect on the performance of our method, since we did not have to rely on any sort of pre-constructed dictionary or lexicon. This natural language "noise" that we were able to account for bodes well not just for this investigation, but for all forms of natural language processing. The specific types of errors we encountered in our dataset mirror issues experienced when trying to draw conclusions from many other forms of unstructured textual data. In qualitative analyses of corpora, coding schemas are commonly used to extract key themes and features from a body of text, but attempting to consolidate them is an arduous task for many of the reasons we discussed in this paper [4]. Furthermore, any research that relies on automated transcription services to generate textual data for study will experience varying levels of accuracy and, in turn, varying levels of "noise" depending on the tool they choose for transcription [5]. Being able to filter out some level of variation before turning to manual analysis would be invaluable in saving time and resources in all fields of study.

The biggest limitation of the work presented in this paper is that it was performed on a relatively small dataset. We observed very strong accuracy as detailed in the "Results" section, but Sureka et al. [8] used a dataset consisting of over 200,000 bug reports and found that the percentage of false positives increased with the size of their data. Although this would very likely be a limitation if we applied our work as-is to higher volume data, the application of duplicate identification to larger datasets also presents an excellent opportunity for future work. With a larger dataset, it becomes possible to train a machine learning model that could potentially make up for whatever deficits arise in the character-based approach we employed in this paper. A classification-based, black box machine learning model could also help provide more nuance in the "grouped duplicate" problem we encountered, as it might be able to gauge some features of the text that we were unable to explicitly define.

A similar limitation that also presents an opportunity for future work is the type of data we used in this paper. While there was some slight variation in the lengths of each statement comprising our dataset, unstructured textual data can vary far more than what we analyzed here. In our data, some statements were only one word long and others were several sentences, but natural language often comes in entire paragraphs, pages, or even books worth of text. So, the method examined in this paper could likely be implemented with little adjustment on short-form text like product reviews, tweets, or bug reports, but would require at least some degree of pre-processing before use with longer text corpora like full, unfragmented video transcripts.

REFERENCES

[1] N. Bettenburg, R. Premraj, T. Zimmermann, and . Sunghun Kim, *Duplicate bug reports considered harmful … really?*, in 2008 IEEE International Conference on Software Maintenance, 2008, pp. 337–345.

[2] M. Ellmann, *Natural language processing (nlp) applied on issue trackers*, in Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering, NL4SE 2018, New York, NY, USA, 2018, Association for Computing Machinery, p. 38–41.

[3] A. Haapala, *python-levenshtein 0.12.2: Python extension for computing string edit distances and similarities.* https://pypi.org/project/python-Levenshtein/, 2021.

[4] J. Hocker, T. Bipat, M. Zachry, and D. W. McDonald, *Sharing your coding schemas: Developing a platform to fit within the qualitative research workflow*, in Proceedings of the 16th International Symposium on Open Collaboration, OpenSym 2020, New York, NY, USA, 2020, Association for Computing Machinery.

[5] J. Y. Kim, C. Liu, R. A. Calvo, K. McCabe, S. C. R. Taylor, B. W. Schuller, and K. Wu, *A comparison of online automatic speech recognition systems and the nonverbal responses to unintelligible speech*, CoRR, abs/1904.12403 (2019).

[6] E. Kodhai, S. Kanmani, A. Kamatchi, R. Radhika, and B. V. Saranya, *Detection of type-1 and type-2 code clones using textual analysis and metrics*, in 2010 International Conference on Recent Trends in Information, Telecommunication and Computing, 2010, pp. 241–243.

[7] L. MacLeod, M.-A. Storey, and A. Bergen, *Code, camera, action: How software developers document and share program knowledge using youtube*, in 2015 IEEE 23rd International Conference on Program Comprehension, 2015, pp. 104–114.

[8] A. Sureka and P. Jalote, *Detecting duplicate bug report using character n-gram-based features*, in 2010 Asia Pacific Software Engineering Conference, 2010, pp. 366–374.

[9] X. Wang, D. Lo, J. Jiang, L. Zhang, and H. Mei, *Extracting paraphrases of technical terms from noisy parallel software corpora*, in Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, Suntec, Singapore, Aug. 2009, Association for Computational Linguistics, pp. 197–200.

[10] L. Yujian and L. Bo, *A normalized levenshtein distance metric*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1091–1095.

EVENT SENSOR DATA PIPELINE FOR TESTING SPIKING MOTION SEGMENTATION ALGORITHM

WALKER A. RICKORD*, SHANE POLDERVAART†, FELIX WANG‡, RYAN DELLANA§, AND SUMA GEORGE CARDWELL¶

Abstract. Neuromorphic event sensors are inspired by the human retina and measure per-pixel brightness changes asynchronously. These cameras have high temporal resolution (μs) with reduced motion blur, high dynamic range (140 dB), and low power consumption (10 mW–10 μW). However, novel algorithms are required to process the sparse data (i.e. motion segmentation, target detection) generated by event sensors. Spiking neural networks are ideal for processing event sensor data. Event-based motion segmentation algorithms cluster the events in a given time snapshot of a scene. This can be leveraged for background subtraction and thus accurate target detection. In this work, we investigate SpikeMS, a deep spiking neural network for motion segmentation. We detail the construction of an event-based data pipeline to prepare raw event data from event-based vision sensors for use in model development. Additionally, the pipeline functions to convert event streams to videos for visualization. We hope this work will come to enable the execution of motion segmentation algorithms at the edge with better energy efficiency through the combination of spiking neural networks, event-based dynamic vision sensors (DVS), and neuromorphic processors.

1. Introduction. Neuromorphic computing has seen a rise in interest due to its ability to distribute both computation and memory along the same "neurons" with high energy efficiency [22]. This architecture overcomes the Von Neumann bottleneck found in classical computing, associated with limited computational speeds and low energy efficiency in conventional devices [23]. Utilizing artificial neurons and synapses, neuromorphic computing's objective is to mimic key computational blocks of the brain for computational efficiency. Based on asynchronous, event-based spikes, neuromorphic systems have the ability to perform complex calculations with a drastic decrease in energy consumption and footprint due to their sparse nature [6]. These systems include both neuromorphic event sensors, which replicate the process of translating sensory information to spikes, and neuromorphic processors, which aim to function similar to that of the biological brain [19].

Neuromorphic platforms are promising since they can meet the size, weight, and power (SWaP) constraints desirable for edge applications. With the commercial production of event-based cameras [29], this is increasingly a viable solution for edge computing systems and applications [12]. Furthermore, as neuroscientists continue to better understand the inner-workings of the brain, researchers will continue to advance neuromorphic algorithms for which neuromorphic hardware is the ideal candidate to achieve peak SWaP performance. In this paper, we provide an overview of neuromorphic event sensors, spiking deep learning based neuromorphic algorithms, and neuromorphic hardware. Then, we discuss the insights obtained from the development of an event sensor data pipeline, followed by the results of our implementation and testing of SpikeMS [24], a deep spiking neural network for motion segmentation on traditional computer hardware [24].

Neuromorphic computing consists of both bio-inspired hardware and algorithms. To better understand the required components for deep learning-based neuromorphic motion segmentation, we will cover a brief explanation of event sensors in Section 1.1, spiking neural networks (SNNs) in Section 1.2, a technique to allow for model training/learning using non-differentiable spikes (i.e. spike layer error reassignment in time – SLAYER) in Section 1.3, and neuromorphic processors in Section 1.4.

*University of Illinois at Urbana-Champaign, walkerr2@illinois.edu
†Sandia National Laboratories, srpolde@sandia.gov
‡Sandia National Laboratories, felwang@sandia.gov
§Sandia National Laboratories, rdellan@sandia.gov
¶Sandia National Laboratories, sgcardw@sandia.gov

1.1. Event Sensors. Event-based sensors (EBS) generate events/spikes due to changes in the sensed quantity and include vision [18], auditory [2] and tactile sensors [3]. Event-based vision sensors are inspired by the human retina. They asynchronously measure the per-pixel brightness change in the scene. Event cameras output the time, pixel location and the polarity of brightness change. These biologically inspired cameras have been designed in order to overcome the limitations of traditional cameras, such as frame-based imaging with low temporal resolution, capturing images at a fixed rate, low dynamic range and subjectivity to motion blur. Many designs exist within the event sensor realm [14, 26], including dynamic vision sensors (DVS) [18], asynchronous time based image sensors (ATIS) [25], dynamic and active pixel vision sensors (DAVIS) [5], and contrast detectors (CD) [27]. DVS event cameras are responsible for reading in brightness changes, whereas ATIS cameras account for brightness changes and assign an additional sub-pixel to account for "absolute" intensity. DAVIS cameras combine an active pixel sensor (APS) with a DVS to achieve smaller pixel sizes in its readout and to enable the sensors to output frames in addition to events [14]. CD sensors stem from ATIS cameras and feature high resolution, low latency, and high dynamic range among other benefits. Since each camera type includes a DVS, we will be primarily focusing on the constructs of this sensor, and specifically Prophesee's implementation.

DVS cameras read in changes in light intensity independently and asynchronously at each pixel. When a pixel is fired, its output contains the coordinates (x, y), timestamp (t), and polarity (p) of the brightness change. The polarity indicates if the change was an increase or decrease in brightness. This enables the analysis of visual information based off of sparse arrays, therefore coming at a fraction of the energy and data cost when compared to traditional frame-based devices [14]. A comparison of event-based and frame-based technologies can be seen in Figure 1.1 and Figure 1.2. Due to the promises of low power computing that spiking affords, there is interest in further work assessing end-to-end neuromorphic systems comprised of event sensors and neuromorphic processors.

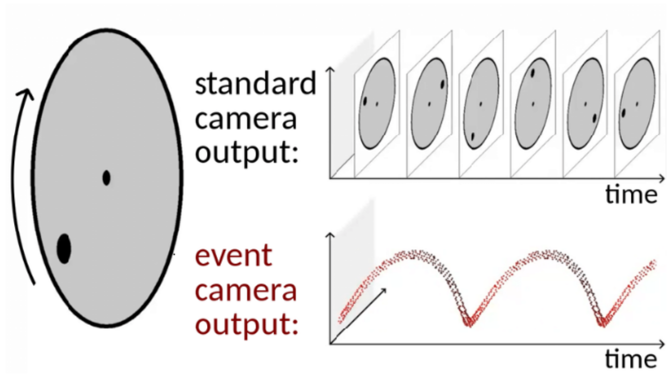


Fig. 1.1. *Output comparison for frame-based and event-based sensors [15]. Image reproduced with permission from [15].*

1.2. Spiking Neural Networks (SNNs). With rising applications of deep learning, there is an increased amount of focus on understanding artificial neural networks (ANNs) and the various architectures associated with them, especially efficient implementation in hardware. ANNs, which are loosely inspired by biological neural networks, are comprised of node layers with biases, weights, transfer functions, activation functions, and specific

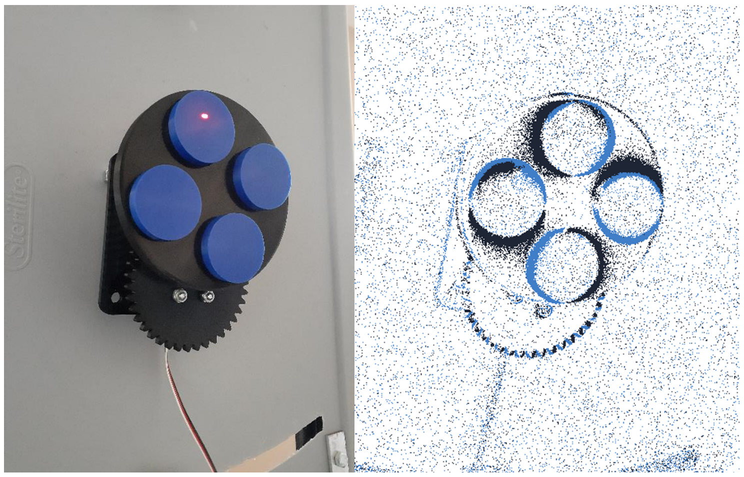


Fig. 1.2. *Image comparison for frame-based image and binned event sensor capture using a Prophesee Gen 4. Blue represents positive pixel events where it got brighter and black represents negative events where the pixel darkened.*

connectivity. Current research is centered on pushing these model architectures to their limits, as increasing parameter size may result in better performance. However, as seen in Table 1.1, this also escalates power consumption and model training time causing high compute costs. Training bigger ANNs is expensive and does not emulate the energy efficiency seen in biological systems. For this reason, there are challenges in deploying deep learning models at the edge in real-time.

Table 1.1

Various state-of-the-art ANN estimated model training costs [34]. Table reproduced with permission from [34].

Model	Hardware	Power(W)	Hours	kWh · PUE	CO_2e	Cloud Compute Cost
Transformer$_{base}$	P100x8	1415.78	12	27	26	$41-$140
Transformer$_{big}$	P100x8	1515.43	84	201	196	$289-$981
ELMo	P100x8	517.66	366	275	262	$433-$1472
BERT$_{base}$	V100x64	12,041.51	79	1507	1438	$3751-$12571
BERT$_{base}$	TPUv2x16	—	96	—	—	$2074-$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	$942,973-$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055-$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902-$43,008

On the other hand, SNNs may alleviate the high costs of traditional neural networks while fitting perfectly into the neuromorphic ecosystem. SNNs take greater biological inspiration than their ANN counterparts. Rather than being based off of an abstract neuron, SNN neurons produce discrete events (distinct from events as defined for event sensors) which exist at specific points in time [8]. Similar to biological neurons, which fire after their membrane potential reaches a threshold, SNN neuron membrane potentials must reach an activation threshold for the corresponding neuron to spike. As described in Section 1.4, a neuron's spike is sent to all its downstream neurons, whose membrane potentials will then be positively or negatively effected depending on established weights. After firing, each neuron resets below its resting potential to mimic the refractory period seen in biology.

Although more fundamental approaches to SNNs have been developed (i.e. spike time interval computational kernel — STICK [16]), we will fixate on deep SNNs. A point of focus for deep SNNs is their training algorithms. Spikes are non-differentiable, which inhibits the use of traditional gradient-descent-based error backpropagation in training [17]. Methods have been presented to overcome this (i.e. [30]). Loihi 2, in particular, has two approaches to neuromorphic error backpropagation: online and offline. Online refers to the ability to deploy SNNs locally and allow approximations of error backpropagation to train the model parameters in real-time. Offline, on the other hand, refers to translating the SNN to an ANN, applying error backpropagation to the ANN, and using those parameters to train the SNN [10].

To encourage researchers to develop on Loihi 2, Intel created their neuromorphic development package, Lava, as a framework for mapping neuro-inspired applications to neuromorphic hardware [9]. Their framework contains both online and offline training mechanisms. An implementation of SLAYER, which will be described in Section 1.3, is utilized for online training in Lava. The offline ANN-SNN training approach, Bootstrap, is not relevant for our work.

1.3. Spike Layer Error Reassignment in Time (SLAYER). SLAYER is a technique to learn synaptic weights and axonal delays using a temporal credit assignment policy for error backpropagation in preceding layers. Importantly, SLAYER has proven to achieve state-of-the-art performance for an SNN on numerous widely used datasets [33].

SLAYER's training mechanism allows for networks to adapt to event-based data directly. It overcomes non-differentiability by using a surrogate gradient based off of the probability density function (PDF) for change of state of a spiking neuron. Error is then assigned to previous layers. SLAYER enables traditional optimization to occur in SNNs, while avoiding the gradient vanishing and explosion problem commonly associated with recurrent neural networks [35].

Lava's implementation of SLAYER supports recurrent network structures and a wider variety of neuron models and synaptic connections among other features [32]. In future works, we hope to use Lava's SLAYER as a method for training SpikeMS in an end-to-end neuromorphic fashion.

1.4. Neuromorphic Processors. The human brain uses approximately 20 W of power for computation [11], making mimicking its energy efficiency highly advantageous. As a result, neuromorphic processors were devised with the aim to emulate the brain and its biological processes in silicon. To design neuromorphic processors, interdisciplinary research has been conducted to develop analog and digital architectures based on the spiking nature found in biological neurons.

Analog processors have the potential to offer greater advantages in energy and latency when compared to an optimized digital application specific integrated circuit (ASIC) [1]. Despite their increased complexity, analog systems are not readily available, are noisy, and require target algorithm development [6]. In the future, if scaled, these systems would be more computationally efficient. Our work focuses on digital neuromorphic processors.

Currently, a state-of-the-art digital neuromorphic processor combines many neuromorphic cores with a few traditional cores. Neuromorphic cores rely on networks-on-chip to route their communications. Rather than communication being dependent on time-steps (clock-driven), communication and computation only happens when a spike occurs (event-driven) [4]. They utilize these spike events to communicate between downstream neurons, with associated weights and delays present for each synaptic connection [31]. One function of traditional cores is to perform on-board translations of data from the dense, synchronous domain to the neuromorphic domain. These features allow for fully integrated memory and computation, sparse connectivity, massive parallelism, network fan-outs, and on-chip learning, all features strongly desired within the neuromorphic algorithms development community [10].

One chip of note is Intel's Loihi 2 [10]. Loihi 2 was developed with vast improvements from its predecessor by focusing on expanding its application space. Specifically, Loihi 2's hardware is primed for easier neuromorphic algorithm development. The processor features the ability to carry integers within spikes, rather than binary values, and can integrate various spiking neural network model types [10]. SpiNNAKER [13] and IBM's TrueNorth [20] are additional chips designed with spiking neural network architectures, however, we will focus on Loihi 2 and its algorithmic benefits throughout the remainder of this discussion.

2. Methods. When examining various neuromorphic implementations of motion segmentation algorithms, it is important to account for the information being passed into each network. We tested SpikeMS for its ability to perform incremental predictions. Specifically, testing on edge cases such as extreme camera motion and rotation to verify the network's ability to generalize outside of what it was trained on. We found it necessary to develop a seamless method for ingesting new event sensor data into the existing SpikeMS model to view the results. In Section 2.1 we detail the process of efficiently converting event data into workable formats. In Section 2.2 we describe the SpikeMS implementation and provide results from testing it on new datasets.

2.1. Event Data Pipeline. The two-element event data pipeline is designed with the purpose of creating a simple interface for event sensor data manipulation. The first element creates a streamlined process of extracting the necessary event information from the event sensor output. This implies converting the RAW event sensor format to the desired file type (i.e. HDF5, NPZ, etc). The second element provides visualization of Prophesee event sensor data.

Prophesee's Metavision [28] is utilized to read in Prophesee event sensor RAW files. Three necessary components must be extracted from the event sensor files to complete these tasks. The first component is the event stream, which consists of time, x-coordinate, y-coordinate, and polarity information. The second component is an array of each events's starting index. Logic for retrieving each event as a bin is built into Metavision's EventsIterator [28]. The final component is the sensor dimensions. Although the pipeline can convert file formats beyond RAW event sensor files (e.g., NPZ, CSV, and TXT), we demonstrate pseudo code for RAW file conversion in Algorithm 1.

To store event information, numerous formats were tested. Hierarchical Data Format version 5 (HDF5) is found to be the most efficient for reading/writing event data when compared to NPZ, CSV, and TXT. NPZ files are formatted similarly to HDF5, but result in slower read/write times when converting RAW data. Although CSV and TXT files are human readable and directly editable, they result in vastly slower read/write times during conversion. For this reason, SpikeMS's prediction feature (an addition to the network described in Section 2.2) is based off of information extracted from HDF5 files using H5PY [7]. Event information is stored in HDF5 files as shown in Table 2.1.

Testing determined NPZ format to be the most effective for video conversion (converting event stream data into a video). HDF5 files perform the same task marginally slower, whereas CSV and TXT files face significantly longer video conversion times. The pipeline's converter enables NPZ files to be generated containing the same information depicted in Table 2.1. The information is quantized into frames based on timestamp and polarity and output as MP4 files. Examples of these outputs are shown in Section 2.2.

Overall, the pipeline is capable of converting a Prophesee RAW file or event stream file into an HDF5 suitable for SpikeMS predictions. Additionally, the pipeline's video converter is a useful tool to visualize original input event streams and SpikeMS predicted event streams

Algorithm 1: Event Sensor Data Pipeline

```
$input = rawFilePath$
$output = outputFilePath$
$outputType$ = Extract output type from path.

$eventsFromRaw$ = Extract events using Prophesee's Metavision [28].
$totalEvents$ = Extract spike count from $eventsFromRaw$.
$dimensions$ = Extract sensor size from $eventsFromRaw$.

$idx$ = Index counter.
$events$, $events\_idx$ = Empty arrays to store events information.
for $eventBin$ in $eventsFromRaw$ do
    $shape$ = Extract length of event bin from $eventBin$.
    $nextIdx \leftarrow idx + shape$
    $events[idx : nextIdx] \leftarrow [t, x, y, p]$ from $eventBin$
    $events\_idx \leftarrow [idx]$
    $idx \leftarrow nextIdx$
end
if $outputType$ is ".hdf5" or ".npz" then
    $events\_dictionary \leftarrow events,\ events\_idx,\ dimensions$
    Open $outputFile$
    $outputFile \leftarrow events\_dictionary$
end
if $outputType$ is ".csv" or ".txt" then
    Open $outputFile\_events$, $outputFile\_events\_idx$, $outputFile\_dimensions$
    $outputFile \leftarrow events,\ events\_idx,\ dimensions$ accordingly.
end
```

Table 2.1
Event data contained in exported files.

Key	Size	Value
"events"	(# of Events, 4)	[t, x, y, p]
"events_idx"	(# of Bins, 1)	[starting index for bin]
"dimensions"	(2)	[height, width]

for direct comparison of results. This enables interoperability within the system. The ideal workflow is shown in Figure 2.1.

2.2. SpikeMS Implementation. SpikeMS was designed by the University of Maryland Dept. of Computer Science's Perception and Robotics Group [24]. The algorithm is a spiking neural network implementation of an encoder-decoder architecture. For motion segmentation, this hourglass-shaped architecture is highly appealing due to its ability to identify objects of interest and reduce background noise. It uses a modified version of SLAYER to train. SpikeMS is available with a pre-trained model on an event-based motion segmentation learning set, EV-IMO [21]. EV-IMO contains pixel-wise motion masks, egomotion (sensor movement), and ground truth depth to enable results such as those seen in Figure 2.2.

To further test its implementation, we incorporated a "predict only" feature to SpikeMS.

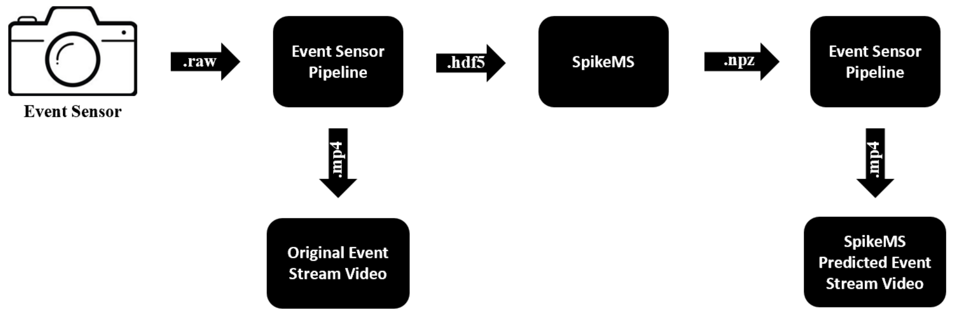


Fig. 2.1. *Event sensor pipeline SpikeMS workflow for visualizing original input and SpikeMS predicted event streams.*

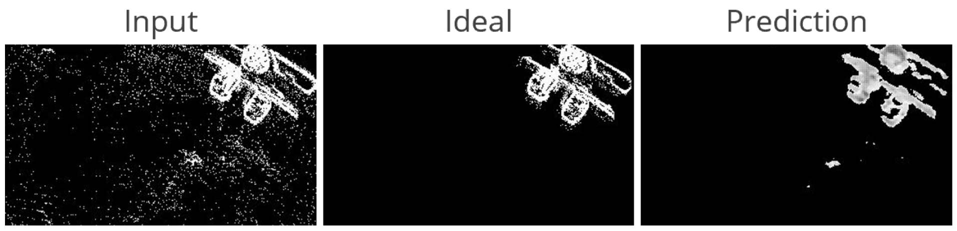


Fig. 2.2. *SpikeMS prediction images compared to input and ideal output for EV-IMO dataset. Predictions generated from our native implementation of SpikeMS.*

The feature does not require mask information, but rather uses event streams and the event indices to make its predictions. As such, the output does not contain ideal images. We allocated the predicted tensors into event streams, similar to the function of an event sensor, to enable visualization. We performed further analysis of SpikeMS's predictions on Prophesee event sensor datasets [29] and scenes captured at the Neuromorphic Research Laboratory (NERL) at Sandia National Laboratories. These datasets include driving scenes, a pendulum swinging, and fan spinning. The results are demonstrated in Figures 2.3 and 2.4 using the visualization tool mentioned in Section 2.1.

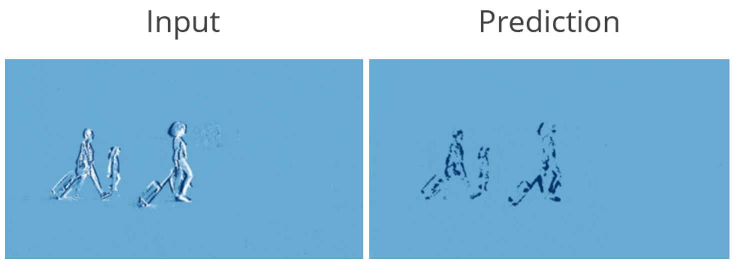


Fig. 2.3. *SpikeMS prediction images on Prophesee driving dataset. SpikeMS correctly identifies multiple moving objects of interest. Predictions generated from our native implementation of SpikeMS and our event sensor data pipeline.*

SpikeMS accurately completes motion segmentation and background subtraction tasks. These scenes as shown in Figure 2.3 and Figure 2.4 introduce egomotion, multiple objects of focus, and background objects to test various situations with SpikeMS.

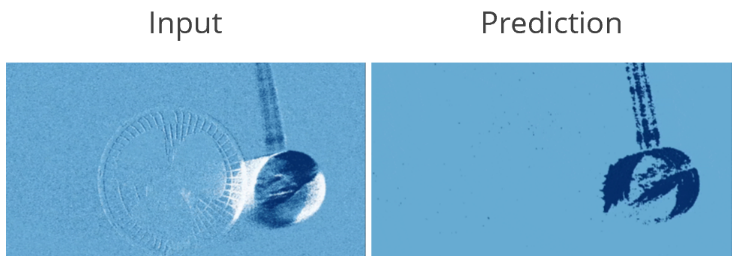


Fig. 2.4. *SpikeMS prediction images on NERL pendulum dataset. Sensor is undergoing egomotion and SpikeMS was correctly able to identify the foreground object. Predictions generated from our native implementation of SpikeMS and our event sensor data pipeline.*

3. Conclusions. We have demonstrated a pipeline that enables flexible visualization of event sensor data and integration of this data into neural networks. Additionally, we have shown results from SpikeMS tested on data external to its training set. We verified that SpikeMS is capable of incremental predictions (predictions from smaller amounts of test data than the dataset it was trained on). Our preliminary results indicate that SpikeMS is a great candidate to implement on a neuromorphic processor due to its success with complex scenarios and edge cases that were not introduced during training. Therefore, in future work, we plan to explore implementing this system on a neuromorphic processor, Intel's Loihi 2, using Lava to enable end-to-end neuromorphic motion segmentation at the edge. With this hardware implementation we can analyze performance along with potential energy benefits without the need for simulation.

4. Acknowledgments. Special thanks to Park Hays for his work related to handling Prophesee data for Algorithm 1.

REFERENCES

[1] S. Agarwal, A. H. Hsia, R. B. Jacobs-Gedrim, D. R. Hughart, S. J. Plimpton, C. D. James, and M. J. Marinella, *Designing an analog crossbar based neuromorphic accelerator*, 2017 Fifth Berkeley Symposium on Energy Efficient Electronic Systems & Steep Transistors Workshop (E3S), (2017), pp. 1–3.

[2] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, *Feature representations for neuromorphic audio spike streams*, Frontiers in neuroscience, 12 (2018), p. 23.

[3] T. Birkoben, H. Winterfeld, S. Fichtner, A. Petraru, and H. Kohlstedt, *A spiking and adapting tactile sensor for neuromorphic applications*, Nature Scientific reports, 10 (2020), pp. 1–11.

[4] K. Boahen, *Ee-iv shannon room #54-134*, May 2019.

[5] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, *A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor*, IEEE Journal of Solid-State Circuits, 49 (2014), pp. 2333–2341.

[6] S. G. Cardwell, C. Vineyard, W. Severa, F. S. Chance, F. Rothganger, F. Wang, S. Musuvathy, C. Teeter, and J. B. Aimone, *Truly heterogeneous HPC: Co-design to achieve what science needs from HPC*, in Communications in Computer and Information Science, Communications in computer and information science, Springer International Publishing, Cham, 2020, pp. 349–365.

[7] A. Collette, *Python and HDF5*, O'Reilly, 2013.

[8] S. Davidson and S. B. Furber, *Comparison of artificial and spiking neural networks on digital hardware*, Front. Neurosci., 15 (2021), p. 651141.

[9] M. Davies, *Taking neuromorphic computing to the next level with loihi 2 technology brief*, tech. rep., Intel, 2021.

[10] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, *Advancing neuromorphic computing with loihi: A survey of results and outlook*, Proceedings of the IEEE, 109 (2021), pp. 911–934.

[11] D. Drubach, *The Brain Explained*, Pearson, Upper Saddle River, NJ, Sept. 1999.

[12] K. Eng, *Understanding the performance of neuromorphic event-based ... - inivation*, May 2020.

[13] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, *The spinnaker project*, Proceedings of the IEEE, 102 (2014), pp. 652–665.

[14] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, *Event-based vision: A survey*, CoRR, abs/1904.08405 (2019).

[15] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, *Asynchronous, photometric feature tracking using events and frames*, ECCV, (2018).

[16] X. Lagorce and R. Benosman, *STICK: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony*, Neural Computation, 27 (2015), pp. 2261–2317.

[17] J. H. Lee, T. Delbruck, and M. Pfeiffer, *Training deep spiking neural networks using backpropagation*, Frontiers in Neuroscience, 10 (2016).

[18] P. Lichtsteiner, C. Posch, and T. Delbruck, *A 128$\times$128 120 dB 15 μs latency asynchronous temporal contrast vision sensor*, IEEE Journal of Solid-State Circuits, 43 (2008), pp. 566–576.

[19] M. A. Mahowald and C. Mead, *The silicon retina*, Scientific American, 264 (1991), pp. 76–82.

[20] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al., *A million spiking-neuron integrated circuit with a scalable communication network and interface*, Science, 345 (2014), pp. 668–673.

[21] A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, *Ev-imo: Motion segmentation dataset and learning pipeline for event cameras*, in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov 2019, pp. 6105–6112.

[22] D. Monroe, *Neuromorphic computing gets ready for the (really) big time*, Commun. ACM, 57 (2014), p. 13–15.

[23] Q.-F. Ou, B.-S. Xiong, L. Yu, J. Wen, L. Wang, and Y. Tong, *In-memory logic operations and neuromorphic computing in non-volatile random access memory*, Materials, 13 (2020), p. 3532.

[24] C. M. Parameshwara, S. Li, C. Fermüller, N. J. Sanket, M. Evanusa, and Y. Aloimonos, *Spikems: Deep spiking neural network for motion segmentation*, CoRR, abs/2105.06562 (2021).

[25] C. Posch, D. Matolin, and R. Wohlgenannt, *A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS*, IEEE Journal of Solid-State Circuits, 46 (2011), pp. 259–275.

[26] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, *Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output*, Proceedings of the IEEE, 102 (2014), pp. 1470–1484.

[27] Prophesee, *Event Based Concepts*. `https://docs.prophesee.ai/stable/concepts.html?highlight=contrast%20detector`, 2022.

[28] ———, *Metavision Intelligence Docs*. `https://docs.prophesee.ai/stable/index.html`, 2022.

[29] ———, *Metavision Intelligence Recordings and Datasets*. `https://docs.prophesee.ai/stable/datasets.html`, 2022. 2022-08-05.

[30] A. Renner, F. Sheldon, A. Zlotnik, L. Tao, and A. Sornborger, *Implementing backpropagation for learning on neuromorphic spiking hardware*, in Proceedings of the Neuro-inspired Computational Elements Workshop, ACM, Mar. 2020.

[31] G. S. Rose, M. S. A. Shawkat, A. Z. Foshie, J. J. Murray, and M. M. Adnan, *A system design perspective on neuromorphic computer processors*, Neuromorphic Computing and Engineering, 1 (2021), p. 022001.

[32] S. B. Shrestha, *Lava-dl slayer*, 2022.

[33] S. B. Shrestha and G. Orchard, *Slayer: Spike layer error reassignment in time*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018.

[34] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in NLP*, CoRR, abs/1906.02243 (2019).

[35] Y. Xing, G. Di Caterina, and J. Soraghan, *A new spiking convolutional recurrent neural network (SCRNN) with applications to event-based hand gesture recognition*, Front. Neurosci., 14 (2020), p. 590164.

WEIGHTED AND PENALIZED RESIDUAL MINIMIZATION FORMULATIONS FOR MODEL REDUCTION OF STEADY HYPERSONIC FLOWS

R. LOEK VAN HEYNINGEN*, PATRICK J. BLONIGAN†, AND ERIC J. PARISH‡

Abstract. Computational simulations of hypersonic flow play an important role in the design of hypersonic vehicles, for which experimental data are scarce. Reduced order models (ROMs) have the potential to make many-query problems, such as design optimization and uncertainty quantification, tractable for this domain. Recent work has shown that residual minimization ROMs which incorporate entropy principles into their optimization statement can possess improved accuracy and robustness for compressible flow. We incorporate this entropy weighting with other residual minimization techniques developed for convective problems, including ℓ^1 minimization and penalized formulations of a constrained minimization problem that enforces conservation. This project investigates the utility of these ROMs for steady hypersonic test cases. Particular attention is paid to accuracy for problems with strong shocks, which are common in hypersonic flow and challenging for projection-based ROMs.

1. Introduction.

The relevance of hypersonic flow to technologies with strategic and commercial interest has spurred the development of computational fluid dynamics (CFD) methods capable of simulating such flows at greater scales and higher levels of fidelity. Despite advances in computational methods, CFD for hypersonic flow is often computationally expensive. This poses a challenge when trying to integrate CFD into engineering studies that require repeated simulations at different operating conditions, such as design optimization or uncertainty quantification.

In these many-query workflows, cheap approximations of a numerical solver, achieved with some form of model order reduction, can be used in place of the high-fidelity solver. Examples of model reduction include the consideration of simplified physics, the construction of purely data-driven surrogates, and the use of projection-based reduced order models (ROMs). Projection-based ROMs tend to require more intrusive access to the full order model (FOM) than other surrogate approaches, but they use the structure of the governing equations to offer high accuracy with limited FOM data and can potentially provide rigorous error estimates [5].

Projection-based ROMs are usually introduced with Galerkin ROMs, which enforce orthogonality of the continuous dynamical system with a basis constructed from FOM data. In fluid dynamics problems for which Galerkin ROMs can lack robustness, they can be augmented with stabilization terms [4] or variable transformations that can enhance their stability [17, 22, 10]. A related class of ROMs are residual minimization ROMs, which aim to minimize the residual in a specified norm rather than achieving exact orthogonality. These include the Least-Squares Petrov Galerkin (LSPG) method [8], which has been demonstrated on large scale turbulent and hypersonic fluid problems [29, 6]

The features of hypersonic flow that make full order CFD simulations expensive also pose challenges for projection-based ROMs. These issues can be addressed through the construction of the reduced basis or the formulation of the ROM itself.

A significant issue is the hyperbolic-like nature of hypersonic flow and the appearance of shocks or sharp gradients. Convection-dominated problems possess a large Komogorov n-width, which means that the linear subspace often used in ROMs will be slow to converge [1]. This has been demonstrated explicitly in a number of common high-speed flow benchmarks [16, 28, 20]. Though a larger basis can always be considered, projection-based ROMs scale

*Massachusetts Institute of Technology, rloekvh@mit.edu
†Sandia National Laboratories, pjblonig@sandia.gov
‡Sandia National Laboratories, ejparis@sandia.gov

poorly with the size of the basis, which can lead to ROMs that are more expensive than the FOMs at moderate basis sizes [6]. This issue has spurred the development of local and adaptive linear bases [1, 23], nonlinear manifold approximations [18], and registration-based methods [27, 21]. For shock dominated flows, particularly effective ROMs can be developed if discontinuities can be aligned in the training data [20, 21, 11].

Other less intrusive modifications to the basis can also have advantages. Offline augmentations of the basis can provide substantial benefits for compressible flow ROMs [3]. Previous work has found that the introduction of simple positivity-enforcing functions can make ROMs that are more stable for hypersonic and chemically reacting flow [11, 13]. Other works have found that using different variable sets to construct the basis can have accuracy advantages. This can have a dramatic effect when using entropy variables to add symmetry to a Galerkin formulation [17, 22], while primitive and entropy variable bases can add accuracy for LSPG [11, 14].

This work is mostly a follow up to [11], which primarily focused on the above improvements to the basis. There are also potential benefits to be gained by reconsidering the construction of the ROM itself rather than the basis. This is the focus of this work, particularly in the context of residual minimization ROMs.

Previous works have pointed out that the usual ℓ^2 norm is dimensionally inconsistent for compressible flow problems [2]. In previous hypersonic studies this was avoided by dividing each component of the residual by a reference scale [6]. Later [22], showed that a norm defined by a so-called entropy-based inner product could lead to even more accurate ROMs for compressible flow. The preconditioned LSPG ROMS of [19] can also be considered as an adaptive modification of the standard ℓ^2 norm. Other works have proposed using an ℓ^1 norm minimization, motivated by its connection to the Total Variation Diminishing property, which is useful to simulations of hyperbolic problems [1]. Minimization of the ℓ^1 norm has been shown to provide improved accuracy over LSPG for some hyperbolic problems [2, 21].

Adding constraints to a residual minimization ROM's optimization statement is a way to reintroduce structure that is lost in the projection to a reduced basis. We focus on the addition of conservativity, which was first derived in [9]. Earlier work on projection-based ROMs for hypersonics showed that the conservativity constraint led to the fastest and most accurate ROMs [6], but the method ran into robustness issues when used on a problem with more limited data [11]. Both of these works used a strict equality constraint, but the conservation violation can also be incorporated as a penalty to the objective function.

This work tests out these variations of the optimization statement to a shock dominated problem from [11]. Despite the fact that [11] showed that mesh adaptivity resulted in the most effective ROMs, we test combinations of entropy weighting, ℓ^1 minimization, and the conservation penalty to both fixed mesh and adapted solutions. This is because not every CFD code will have adaptive capabilities and not every hypersonics problem will be amenable to grid tailoring. The ROM formulations and software used are discussed in Section 1. Results on fixed and adapted meshes for problems with sharp features are shown in Section 3, and conclusions are given in Section 4.

2. Methods.

2.1. Software and Application.

2.1.1. Hypersonic CFD code. We build off the framework for hypersonic model reduction developed in [6, 11]. For the full order model, we use the Sandia Parallel Aerodynamics Reentry Code, or SPARC [12]. SPARC solves the Navier-Stokes equations using a cell-centered finite volume discretization on structured or unstructured meshes. While

SPARC has capabilities for turbulence models and thermochemical nonequilibrium, we restrict our attention to flows that remain laminar and in equilibrium. For steady flow, the FOM is expressed as

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\mu}) = 0. \tag{2.1}$$

The function $\boldsymbol{f}$ will be referred to as the residual and the solution of (2.1) is an evaluation of the FOM.

For a d-dimensional domain discretized with n cell centers, the total number of degrees of freedom is $N = (d+2)n$, with $\boldsymbol{f}(\cdot,\cdot) \in \mathbb{R}^N$, $\boldsymbol{x} \in \mathbb{R}^N$, and $\boldsymbol{\mu} \in \mathbb{R}^q$. The state variables are the conservative variables density ρ, momentum $\rho\boldsymbol{u}$ and energy ρE, while $\boldsymbol{\mu}$ is a vector of parameters. For the ROM basis, we also make use of the primitive variables $\boldsymbol{x}_{\text{prim}} = (\rho, \boldsymbol{u}, T)$. The expression for $\boldsymbol{f}$ can be found in [12].

For the steady problems considered in this work, SPARC uses pseudo-time stepping. SPARC also has a form of mesh adaptivity known as grid tailoring [26, 7]. This technique aligns the mesh with a bow shock, which can greatly improve heat flux computations for finite volume methods. Details of SPARC's grid tailoring implementation can be found in [11].

2.1.2. Reduced order modeling library. The ROM routines are implemented with Pressio, a header-only C++ library [25]. This lightweight interface allows the ROM methods to reuse the data structure and methods of the FOM, allowing for large-scale ROM studies.

Pressio requires the evaluation of $\boldsymbol{f}$ and the application of the Jacobian of $\boldsymbol{f}$ to a vector. As opposed to the previous Pressio and SPARC studies, this work uses exact Jacobian-vector products using automatic differentiation (AD) provided by the SACADO library [24]. The exact Jacobian vector product is slower than the previous finite difference approach but is expected to provide robustness. A full study of the tradeoffs for the AD capability is out of scope for this work.

2.2. Reduced order model techniques.

2.2.1. Basis generation. We assume that the solution to our PDE (2.1) can be represented as a composition of a nonlinear and affine mapping,

$$\boldsymbol{x}(\boldsymbol{\mu}) = \boldsymbol{h}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}) \tag{2.2}$$

where $\boldsymbol{h} : \mathbb{R}^N \to \mathbb{R}^N$, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times p}$ is a data-driven basis, and $\hat{\boldsymbol{x}} \in \mathbb{R}^p$ are the expansion coefficients in this basis. An offset vector, $\boldsymbol{x}^0$, is used to better center the data around the origin and is often set to zero or the mean of the training data. The nonlinear mapping $\boldsymbol{h}$ is a simple "clipper" function that prevents the solution from reaching nonphysical states by ensuring that the density and temperature remain positive.

A standard choice for generating the basis $\boldsymbol{\Phi}$ is proper-orthogonal decomposition (POD). With training snapshots $\boldsymbol{X}_{\text{train}}$, optionally shifted and scaled as shown in [6, 11], the POD basis consists of the p leading left singular vectors of the singular value decomposition of $\boldsymbol{X}_{\text{train}}$. The number of modes p can be can be chosen based off the decay of the singular values.

As mentioned in the introduction, global POD modes tend to do a poor job of capturing flows with discontinuities or strong gradients. As an alternative, we use the dictionary basis described in [11]. To query the basis for a new parameter $\boldsymbol{\mu}$, a basis is built up out of the displacements from the nearest training snapshot in $\boldsymbol{X}_{\text{train}}$. First, the training parameter closest to $\boldsymbol{\mu}$ is determined as $\boldsymbol{\mu}_n$ with a corresponding snapshot $\boldsymbol{x}_n$. Then, the p snapshots closest to $\boldsymbol{x}_n$, as measured by a metric in the $\boldsymbol{\mu}_n$ field, are loaded into a matrix $\boldsymbol{X}(\boldsymbol{\mu}_n(\boldsymbol{\mu}))$. The local basis is then defined as

$$\boldsymbol{\Phi}(\boldsymbol{\mu}) = \boldsymbol{X}(\boldsymbol{\mu}_n(\boldsymbol{\mu})) - \boldsymbol{x}_n(\boldsymbol{\mu}_n(\boldsymbol{\mu})) \tag{2.3}$$

where subtraction is performed column-wise.

When the FOMs use grid tailoring and $\boldsymbol{\mu}$ has quantities that can change the shock location, each snapshot is potentially solved for on a different grid. This means that for a new parameter set $\boldsymbol{\mu}$, we must first calculate a mesh for the ROM to use. In this case, a dictionary basis is also used for training data of grid displacements. Computing a POD basis for this training data is possible but was not attempted here. Note that the grid basis can have a different order than the basis for the fields themselves. The ROM grid is constructed before the residual minimization process begins rather than being updated at each iteration.

The basis is also constructed using primitive rather than conservative variables. Primitive variable bases were preferred for all tests except for one quantity of interest (QoI) in a single test case in [11]. Whether primitive variables are always more effective is an open question, and it is pointed out in [11] that the optimal choice of state variable may depend on test case and QoI[1]. For example, surface heat flux is often of interest for hypersonic flow, and it has a simpler dependence on the primitive variables than conservative variables. Another reason for the difference in accuracy could be that the nonlinear mapping $\boldsymbol{h}$ can act directly on the primitive variables but requires extra operations to act on the conservative variables [11].

2.2.2. Residual minimization: choice of norm. The LSPG ROM of order p finds a solution $\hat{\boldsymbol{x}} \in \mathbb{R}^p$ that minimizes the residual in the the ℓ^2 norm

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{z}} \in \mathbb{R}^p}{\arg\min} \|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_2. \tag{2.4}$$

Hyper-reduction [8, 30] is incorporated with the matrix $\boldsymbol{A}$, which we ignore in this work by letting $\boldsymbol{A} = \boldsymbol{I}$.

We solve (2.4) with Gauss-Newton iterations. Given an iterate $\hat{\boldsymbol{z}}^k$, the next iterate is determined by setting up and solving a linear least squares problem

$$\boldsymbol{f}_k = \boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}^k; \boldsymbol{\mu}), \quad \boldsymbol{J}_k = \boldsymbol{A}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{h}(\boldsymbol{x})} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}} \tag{2.5}$$

$$\delta\hat{\boldsymbol{z}}^k = \underset{\delta\hat{\boldsymbol{z}} \in \mathbb{R}^p}{\arg\min} \|\boldsymbol{J}_k \delta\hat{\boldsymbol{z}} + \boldsymbol{f}_k\|_2 \tag{2.6}$$

$$\hat{\boldsymbol{z}}^{k+1} = \hat{\boldsymbol{z}}^k + \alpha_k \delta\hat{\boldsymbol{z}}^k. \tag{2.7}$$

Note that while α_k can be a stepsize defined through a line-search or trust region algorithm, here we leave it equal to one.

As with any Newton-type method, the starting iterate has a substantial effect on convergence and efficiency properties. Based on the conclusions from [11], we use radial basis function (RBF) interpolation to construct an initial guess $\hat{\boldsymbol{z}}^0$. An effective initial guess allows the steady ROM to obtain susbstantial speedups over the hypersonic FOMs even without hyper-reduction, since only a moderate number of Newton iterations are typically required for convergence [6].

[1]Note that in [22], the use of entropy snapshots and have been shown was shown to be advantageous. We do not consider this here, as the advantage was somewhat minor and we expect the entropy snapshots to be less helpful for LSPG with a dictionary basis.

Many hypersonic codes including SPARC output quantities in dimensional variables. The ℓ^2 norm of the dimensional residual is dimensionally inconsistent and problematic numerically; the energy is typically up to five orders of magnitude greater than the density, for example. This can be avoided by considering a weighted LSPG statement

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{z}} \in \mathbb{R}^p}{\arg\min} \|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_{2,\boldsymbol{W}} \tag{2.8}$$

where $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ and $\|\boldsymbol{f}\|_{2,\boldsymbol{W}} = \boldsymbol{f}^T\boldsymbol{W}\boldsymbol{f}$. Previous works used a diagonally scaled ℓ^2 norm, where $\boldsymbol{W}$ is a matrix that nondimensionalizes each component of $\boldsymbol{f}$ by including the reciprocal of a reference state $\boldsymbol{x}_{\text{ref}}$.

An alternative ℓ^2 was derived in [22], using the entropy forms of compressible flow equations. Using a set of entropy variables $\boldsymbol{w}$ in the formulation of the FOM can result in equations that provably preserve an entropy functional and can provide stability by symmetrizing the equations [15]. While we do not use an entropy formulation of the FOM, we use the observation of [22] that for a single collection of conservative variables $\boldsymbol{x} \in \mathbb{R}^{d+2}$ and entropy variables $\boldsymbol{w} \in \mathbb{R}^{d+2}$, the following expression is dimensionally consistent:

$$\boldsymbol{x}^T \left(\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}}(\boldsymbol{x}_{\text{ref}})\right) \boldsymbol{x}. \tag{2.9}$$

Since the residual $\boldsymbol{f}$ has the same units as $\boldsymbol{x}$, the inner product $\boldsymbol{f}^T\boldsymbol{W}_{\text{ent}}\boldsymbol{f}$ will be dimensionally consistent if $\boldsymbol{W}_{\text{ent}} \in \mathbb{R}^{N \times N}$ is a block diagonal matrix where each block is $\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}}(\boldsymbol{x}_{\text{ref}}) \in \mathbb{R}^{(d+2)\times(d+2)}$.

Note that $\boldsymbol{x}_{\text{ref}}$ for the diagonal or entropy-based weights could be made to adapt for each degree of freedom or time step, but here it is kept constant through all ROM iterations as the mean of the training snapshots. An adaptively weighted norm would have similarities to the preconditioned LSPG ROMs of [19], which propose the use of norms adaptively weighted by an approximate Jacobian inverse.

Another alternative norm choice is proposed in [1]

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{z}} \in \mathbb{R}^p}{\arg\min} \|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_1. \tag{2.10}$$

The ℓ^1 minimization problem can be solved as a linear program or using iteratively reweighted least squares (IRLS). This method uses the fact that the ℓ^1 can be expressed as a weighted ℓ^2 where the weighting is a diagonal matrix with $|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{x}}; \boldsymbol{\mu})|^{-1}$ on the diagonal. This weighting implicitly depends on the solution of the minimization problem, so an iterative procedure is used to approach the optimum. For an iteration k of the IRLS algorithm, a weight matrix

$$\boldsymbol{W}^k = \operatorname{diag}\left(|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}^k; \boldsymbol{\mu})|^{-1}\right) \tag{2.11}$$

is formed and the minimization problem

$$\hat{\boldsymbol{z}}^{k+1} = \underset{\hat{\boldsymbol{z}} \in \mathbb{R}^p}{\arg\min} \|\boldsymbol{A}\boldsymbol{f}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_{2,\boldsymbol{W}^k} \tag{2.12}$$

is solved. This process is repeated until convergence.

In addition to being more expensive than the usual Gauss-Newton method, there are fundamental difficulties to solving (2.10). Two of these are described in [2]. First, the ℓ^1 norm is not convex, which can be addressed with a convex regularization term. The choice of regularization is an open question for this problem. Furthermore, the norm is not differentiable around 0. As suggested in [2], we use the Huber norm which smoothly switches between to the ℓ^2 norm close to the origin. The norm is defined as $\|\boldsymbol{f}\|_H = \sum_{i=1}^N H_\delta(f_i)$ where H is a function defined for some constant δ

$$H_\delta(f_i) = \begin{cases} f_i^2 & \text{if } |f_i| < \delta, \\ 2\delta(|f_i| - \delta) & \text{if } |f_i| \geq \delta. \end{cases} \tag{2.13}$$

In IRLS, this is accomplished by changing the weight matrix to

$$\boldsymbol{W}_{ii} = \begin{cases} 1 & \text{if } |f_i| < \delta, \\ 2\delta/(|f_i|) & \text{if } |f_i| \geq \delta. \end{cases} \tag{2.14}$$

As in [2], we let $\delta = 1 \times 10^{-6} \max(1.0, \|\boldsymbol{f}\|_\infty)$.

Finally, the ℓ^1 norm is also not dimensionally consistent for a dimensional residual. The ℓ^1 norm is not defined in terms of an inner product, but we can change the definition of the residual slightly to have a problem that is better scaled. The diagonal and entropy weight matrices defined so far are SPD, so we can take their Cholesky factorization as $\boldsymbol{W} = \boldsymbol{L}^T\boldsymbol{L}$. Then we can minimize the weighted residual $\boldsymbol{Lf}$. In practice, this is not much different than how the weighted ℓ^2 minimization is implemented.

2.2.3. Residual minimization: constraints. In addition to modifying the norm, we can add constraints to the optimization statement that can enforce structure within the ROM. Unlike the finite volume-based FOM on which they are based, the ROMs described so far do not inherently satisfy conservation laws.

This property can be enforced with the conservation constraint first derived in [9] with the introduction of conservative LSPG, or C-LSPG. It is expressed with an equality constraint that encodes conservation violations with a constant matrix $\boldsymbol{C} \in \mathbb{R}^{5\times N}$

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{z}}\in\mathbb{R}^p}{\arg\min} \|\boldsymbol{Af}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_q \tag{2.15}$$

$$\text{s.t. } \boldsymbol{Cf}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu}) = 0. \tag{2.16}$$

The use of C-LSPG resulted in fast and accurate ROMs in hypersonics in [6]. When C-LSPG was applied to problems with sparser training data however, it suffered from robustness issues [11]. We take an approach suggested in [9] to deal with potential infeasibility of the constraint by recasting the equality constraint as a penalty

$$\hat{\boldsymbol{x}} = \underset{\hat{\boldsymbol{z}}\in\mathbb{R}^p}{\arg\min} \|\boldsymbol{Af}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_{q_1} + \nu\|\boldsymbol{Cf}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})\|_{q_2} \tag{2.17}$$

for some constant ν. We will refer to this method as C(ν)-LSPG.

We note that q_1 and q_2 can be different of norms. Since $\boldsymbol{C}$ is a dimensionless matrix, the arguments regarding dimensional inconsistency also apply to the penalty term and q_2 should be weighted appropriately to ensure dimensional consistency. Without the weighted norm choice, the penalty method always diverges in the following examples due to large magnitudes in the energy component.

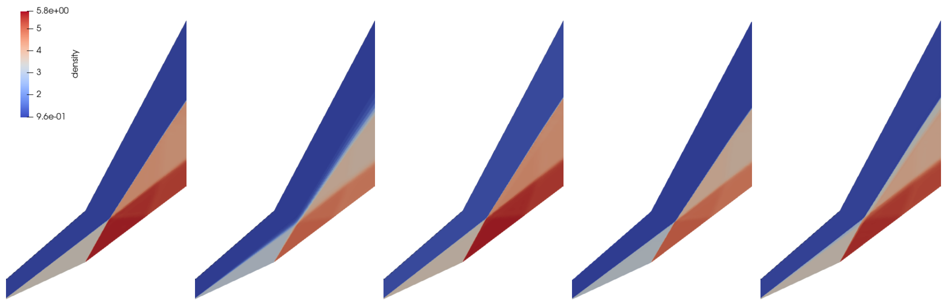
Since all of our considered norms can be expressed as weighted ℓ^2 norms, we can solve this problem with Gauss-Newton. If q_1 is a ℓ^2 norm weighted with $\boldsymbol{W}_1$ and q_2 is a ℓ^2 norm weighted with $\boldsymbol{W}_2$ and each weight matrix has a respective Cholesky factorization $\boldsymbol{W}_1 = \boldsymbol{L}_1^T\boldsymbol{L}_1$ and $\boldsymbol{W}_2 = \boldsymbol{L}_2^T\boldsymbol{L}_2$, then (2.17) can be written as

$$\hat{\boldsymbol{x}} = \min_{\hat{\boldsymbol{z}}\in\mathbb{R}^p} \left\| \begin{pmatrix} \boldsymbol{L}_1\boldsymbol{Af}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu}) \\ \boldsymbol{L}_2\sqrt{\nu}\boldsymbol{Cf}(\boldsymbol{x}^0(\boldsymbol{\mu}) + \boldsymbol{\Phi}\hat{\boldsymbol{z}}; \boldsymbol{\mu})) \end{pmatrix} \right\|_2 \tag{2.18}$$

Note that while the strict constraint provably restores conservativity, the penalty method has no such guarantees.

3. Results. We consider the accuracy of the above formulations on two test cases with parametric-dependent discontinuities in the flow field. The first example is used for visualizations of the different ROM methods while we perform an error analysis on the second case over a range of parameters. The ROMs considered are specifically LSPG with diagonal scaling and entropy scaling, ℓ^1 with diagonal and entropy scaling, and C(ν)-LSPG. The entropy weighting is constructed with the same entropy variables described in the appendix of [22]. A solver with ℓ^1 and a conservative penalty was implemented and tested, but the results are not shown here as they were more or less identical to the penalized LSPG method.

3.1. Double wedge. The first example is a double wedge geometry of the type often used to study shock-boundary layer interactions. The FOM uses a mesh with 122,880 cells. The parameter set $\boldsymbol{\mu}$ consists of a single quantity, the free-stream Mach number, Ma. We use a small parameter range running from $Ma = 4$ to $Ma = 5$ and use 6 equally-spaced training points with intervals of 0.2. The ROM uses a POD basis with $p = 6$ and is run until the residual decreases 6 orders of magnitude. Each ROM is evaluated at $Ma = 4.3$ and visualized in Figure 3.1. Even for this mild parameter variation, the minimization statement has a noticeable impact on the ROM solution.



(a) FOM (b) LSPG, $\boldsymbol{W}_{diag}$ (c) LSPG, $\boldsymbol{W}_{ent}$ (d) L1 (e) C(ν)-LSPG

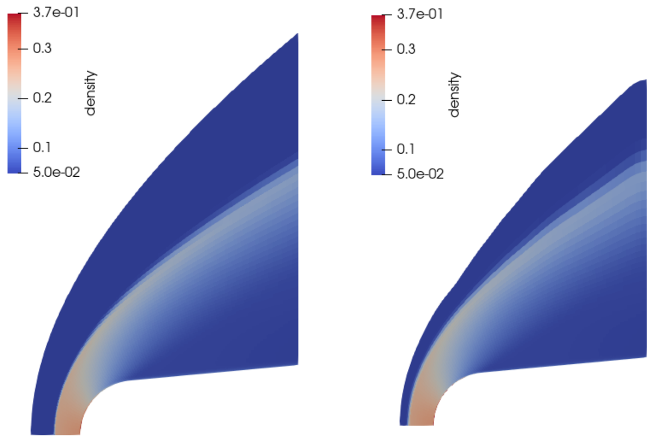
Fig. 3.1: Sample visualizations of a double wedge problem comapring the FOM to several minimization formulations. Minor changes in the minimization statement can have qualitative effects on the ROM solution

3.2. Single wedge. This is a repeat of the first example from [11], which considers laminar ideal gas flow over a 2D blunt wedge geometry. The FOM is run on a mesh with 100,000 cells and pseudo-time stepping is employed until the residual decreases by 7 orders of magnitude. The flow has a fixed free-stream temperature of $T_\infty = 216.66K$ and a baseline Reynolds number of $Re = 1.5e7$. Parametric variations occur in the free-stream density ρ_∞ and the free-stream velocity in order to vary the free stream Mach number. The free-stream density ranges from 0.03 to 0.09 while the Mach number goes from 3.0 to 10.0. See Figure 3.2 for sample visualizations of fixed mesh and grid tailored solutions.

To construct the ROM, we use training points at

$$\boldsymbol{\mu}_{\text{train}} = \rho_{\text{train}} \times Ma_{\text{train}} = [0.04, 0.06, 0.08] \times [3.0, 5.0, 7.0, 9.0].$$

Due to the poor performance of the POD basis shown in [11], the basis is a dictionary with $p = 8$ and RBF interpolation for the initial guess. Each ROM minimization routine is run



(a) Fixed mesh (b) Grid tailored

Fig. 3.2: Sample visualization of the blunt wedge with $\rho_\infty = 0.05$ and $\mathrm{Ma}_\infty = 5.0$ comparing fixed mesh and grid tailored solutions. The tailored solution iteratively deforms the mesh in order to have a mesh aligned with the bow shock.

for 100 iterations or until a relative residual tolerance of 1e-6 is reached. We consider the same three error metrics considered in [11]. First, the standard ℓ^2 state error:

$$\varepsilon_x(\boldsymbol{\mu}) = \frac{\|\boldsymbol{x}_{\mathrm{FOM}}(\boldsymbol{\mu}) - \boldsymbol{x}_{\mathrm{ROM}}(\boldsymbol{\mu})\|_2}{\|\boldsymbol{x}_{\mathrm{FOM}}(\boldsymbol{\mu})\|_2}. \tag{3.1}$$

We also check the convergence of two scalar quantities of interest (QoI): axial force F_x and wall heat flux Q_{wall}. Each ROM is also evaluated for relative errors in these two quantities, denoted as ε_{F_x} and $\varepsilon_{Q_{\mathrm{wall}}}$. It was demonstrated empirically in [11] that lower state errors do not always correspond to lower QoI errors. While ROMs can naturally lower the state error by increasing p, errors for particular QoIs are often of greater practical interest. Accuracy is calculated on a regular grid in parameter space. We exclude the training set and include small regions of extrapolation. When reporting the mean errors, we discard the cases with the lowest and highest errors to eliminate outliers.

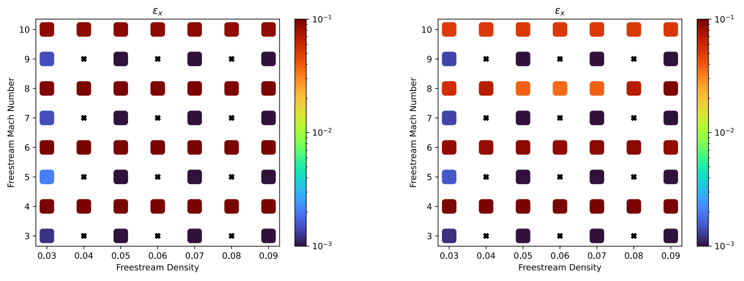
3.2.1. Fixed mesh. The average errors for state, axial force, and integrated heat flux are given in table 3.1. We note first that that the errors for all cases are quite large, especially for the integrated heat flux. This was observed in [11] and motivated the use of the grid tailored ROMs. The entropy-weighted LSPG ROMs on average result in very slightly improved state and integrated heat flux errors while the error in the axial force is increased marginally. A similar pattern occurs with ℓ^1 minimization, where slight improvements are seen with in state errors and integrated heat flux with the entropy weighting, but the axial force becomes less accurate. The lowest errors in all three metrics are achieved with the conservative penalty.

To get a better idea of where the errors are effected, we plot the errors for each tested parameter. Figure 3.3 shows the effect of the weight matrix on the state error for LSPG. The fixed mesh ROM struggles to capture the flow in between the Mach numbers used in the training set. The introduction of the entropy-based weight matrix slightly improves

Method	mean ε_x	mean ε_{F_x}	mean $\varepsilon_{Q_{\text{wall}}}$
LSPG, $\boldsymbol{W}_{diag}$	0.08859	0.03916	0.19221
LSPG, $\boldsymbol{W}_{ent}$	0.05248	0.04268	0.16729
ℓ^1, $\boldsymbol{W}_{diag}$	0.12001	0.03113	0.21810
ℓ^1, $\boldsymbol{W}_{ent}$	0.09648	0.04186	0.18696
C(ν)-LSPG	0.03979	0.01776	0.07308

Table 3.1: Average state and QoI errors for the fixed mesh ROMs

error in the higher Mach numbers, $Ma = 8$ and $Ma = 10$. Figure 3.4 shows that the there is a small improvement of the integrated heat flux in this region while the axial force is less accurate here.



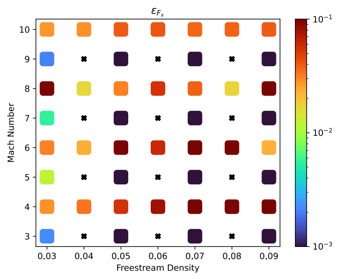
(a) LSPG, $\boldsymbol{W}_{diag}$ (b) LSPG, $\boldsymbol{W}_{ent}$

Fig. 3.3: State errors for LSPG with different choices of weight matrix on a fixed mesh. Training cases denoted by black X symbols.

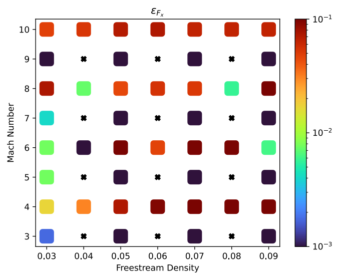
The ℓ^1 ROMs state errors are shown in Figure 3.5 and the QoI errors are shown in Figure 3.6. The ROMs still struggle to reach acceptable state errors in between the Mach numbers used for training. The ℓ^1 norm with diagonal scaling results in a noticeably more accurate prediction of axial force in the high Mach number range, though large errors are still observed when the Mach number is small.

The state errors for the C(ν)-LSPG are shown in Figure 3.7 while the QoIs are shown in Figure 3.8. Improvements are seen almost uniformly across the parameter range. Both weight matrix choices give very similar results, so only errors with the entropy weighting are shown. There are still regions of high error, particularly for low Mach numbers and some regions outside the convex hull of the parameter set, but the addition of the conservative penalty greatly expands the region where the ROMs can be considered reasonably accurate. This effect is particularly large the QoIs, with significant improvements seen in the heat flux measurements. We do note that the quality of the ROM does depend the choice of ν. See Figure 3.9 for a demonstration of the impact ν has on heat flux errors. In this case, larger values of ν tend to result in more accurate ROMs.

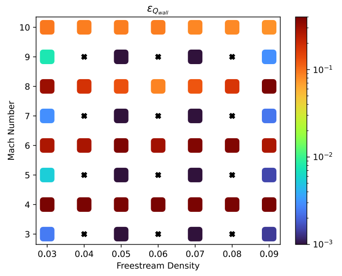
3.2.2. Grid tailoring. The same experiments are repeated with the grid tailored ROMs. We note that the FOMs themselves are more accurate with grid tailoring. The average errors are reported in table 3.2. Consistent with previous studies, the use of mesh adaption greatly increases the accuracy of the ROM. For both LSPG and ℓ^1 minimization,



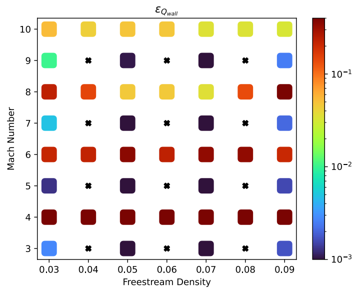
(a) LSPG, $\boldsymbol{W}_{diag}$, axial force



(b) LSPG, $\boldsymbol{W}_{ent}$, axial force

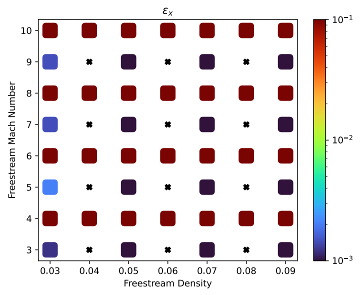


(c) LSPG, $\boldsymbol{W}_{diag}$, heat flux

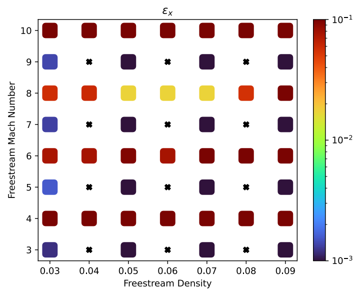


(d) LSPG, $\boldsymbol{W}_{ent}$, heat flux

Fig. 3.4: QoI errors for LSPG with different choices of weight matrix on a fixed mesh.



(a) $\ell^1, \boldsymbol{W}_{diag}$

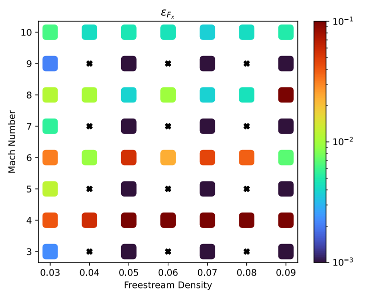


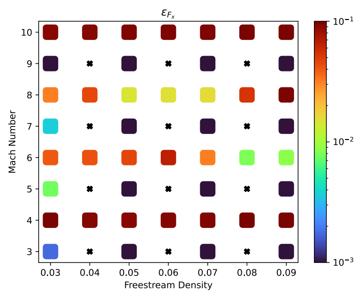
(b) $\ell^1, \boldsymbol{W}_{ent}$

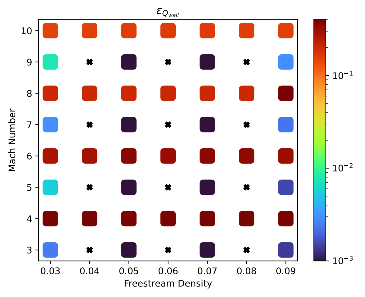
Fig. 3.5: State errors for ℓ^1 with different choices of weight matrix on a fixed mesh.

the entropy-based weighting results in greater accuracy for the state and axial force with minor loss of accuracy in the heat flux. The addition of the conservative penalty lowers the state and axial force errors. Interestingly, the average integrated heat flux error is slightly increased, while for fixed mesh C(ν)-LSPG resulted in dramatic improvements for integrated heat flux errors.

As before, we plot the pattern of the different errors across the range of parameters. All of the state errors are shown in Figure 3.10, the axial force errors in Figure 3.11, and the heat flux errors in Figure 3.12. There are fewer discernable patterns than in the fixed


(a) ℓ^1, $\boldsymbol{W}_{diag}$, axial force


(b) ℓ^1, $\boldsymbol{W}_{ent}$, axial force


(c) ℓ^1, $\boldsymbol{W}_{diag}$, heat flux

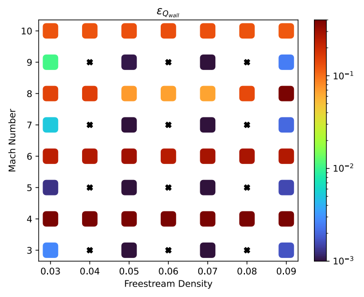

(d) ℓ^1, $\boldsymbol{W}_{ent}$, heat flux

Fig. 3.6: QoI errors for ℓ^1 with different choices of weight matrix on a fixed mesh.

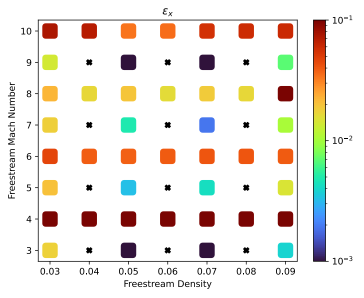
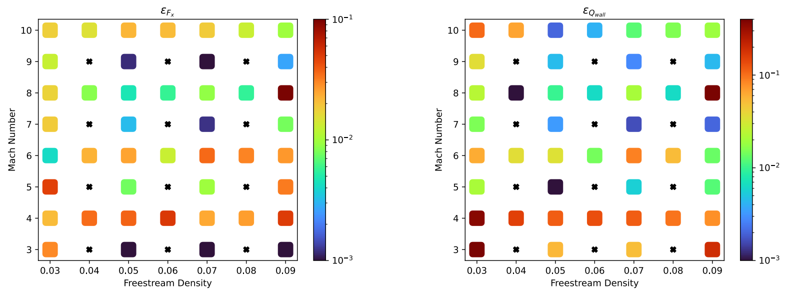


Fig. 3.7: State errors for C(ν)-LSPG with $\nu = 10$

mesh case. In general, C(ν)-LSPG provides the lowest errors around the edges of the graph, hinting at a potential for better extrapolation properties. Note that for grid tailoring we found that increasing ν did not increase the solution quality as it did for the fixed mesh case. This points to a sensitivity of this method to its penalty parameter. This method will require tuning of the penalty parameter, and more study is required to narrow down what order of magnitude the parameter should lie in.

The relative gain of using these alternative minimization statements seems greater for the fixed mesh case. While improvements in ROM accuracy can still be obtained, the



(a) Axial force error (b) Heat flux error

Fig. 3.8: QoI errors for C(ν)-LSPG with $\nu = 10$ on a fixed mesh

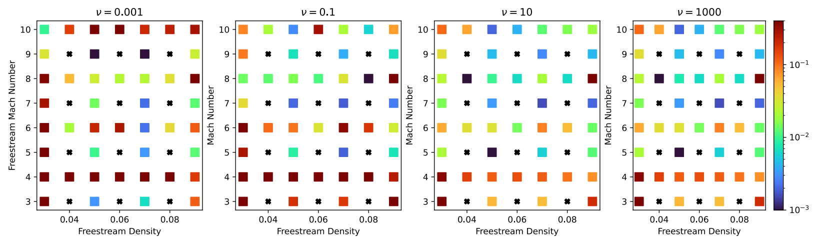


Fig. 3.9: Heat flux errors for the conservative penalty method with different choices of ν.

alignment of features is a powerful enough tool such that the details of the minimization statement are less relevant.

4. Conclusions. This work analyzed the importance of different ROM minimization statements on a hypersonic flow problem with sharp features and limited data. Different choices of norms and a penalty constraint were introduced and applied to problems that had challenged more standard ROM routines in past works. A few conclusions can be reached. First, defining a ROM in terms of the entropy-based norm can result in lower state errors with minimal computational effort. That said, in this case it did not lead the same level of improvement that was seen in the unsteady problems studied in [22]. This could be due to the differences in these hypersonic problems compared to those considered in that work, or it could be that the entropy norm has less potential impact in a steady problem. The impact on QoIs is not consistent. This is an issue across all of the ROMs considered, emphasizing the conclusion from [11] that the best choice of ROM may be problem and QoI dependent.

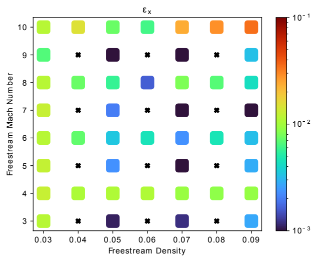
The ℓ^1 solver used here, while effective for certain QoIs in certain parameter ranges, seems to provide less of an advantage than in previous works like [2, 21], where the ℓ^1 ROM is deemed consistently better. The implementation can likely be improved by examining the choice of δ for the Huber norm or by considering different regularizations.

The most accurate ROM studied here was the penalized variant of C-LSPG. In particular on the fixed mesh, C(ν)-LSPG was more accurate in a large section of parameter space. It is important to determine how to choose the parameter, or whether the penalized or strictly enforced constraint should be used.

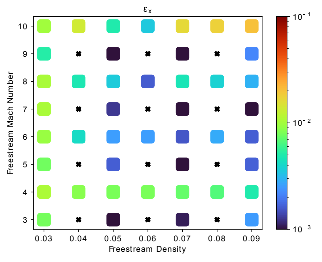
Since the goal of these ROMs are to speed up computational studies, a definitive con-

Method	mean ε_x	mean ε_{F_x}	mean $\varepsilon_{Q_{\text{wall}}}$
LSPG, $\boldsymbol{W}_{diag}$	0.00723	0.01632	0.07467
LSPG, $\boldsymbol{W}_{ent}$	0.00519	0.01317	0.07557
ℓ^1, $\boldsymbol{W}_{\text{diag}}$	0.00697	0.01369	0.07455
ℓ^1, $\boldsymbol{W}_{ent}$	0.00476	0.01007	0.07710
C(ν)-LSPG	0.00397	0.00564	0.08580

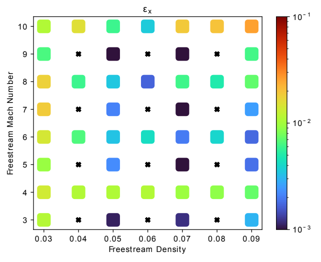
Table 3.2: Average state and QoI errors for grid tailored ROMs



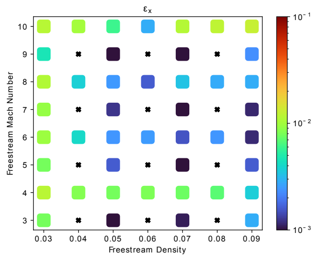
(a) LSPG, $\boldsymbol{W}_{diag}$



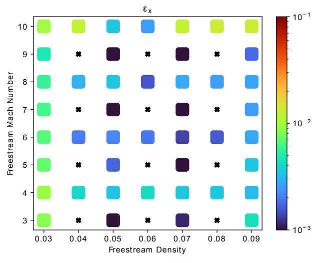
(b) LSPG, Went, $\boldsymbol{W}_{ent}$



(c) ℓ^1 diag, $\boldsymbol{W}_{diag}$



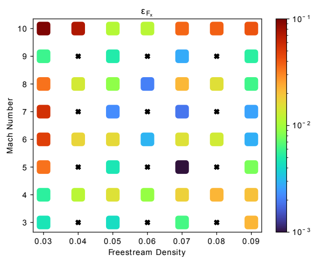
(d) ℓ^1 entropy, $\boldsymbol{W}_{ent}$



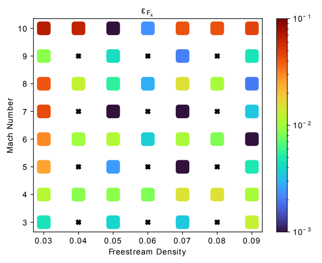
(e) C(ν)-LSPG, $\nu = 0.001$

Fig. 3.10: State errors for all ROM variants with grid tailoring

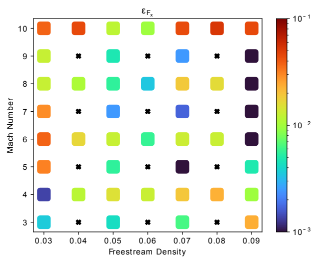
clusion on the overall effectiveness of each ROM must consider computational cost. Furthermore, all of the ROMs considered here should be tested on different hypersonic test cases. It could be that this flow, with a single bow shock and not many secondary effects, is particularly prone to causing conservation violations when expanded in a reduced ba-



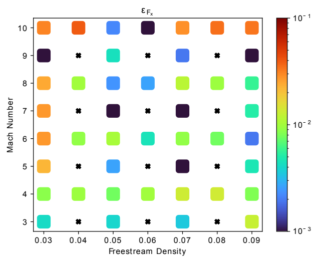
(a) LSPG, $\boldsymbol{W}_{diag}$



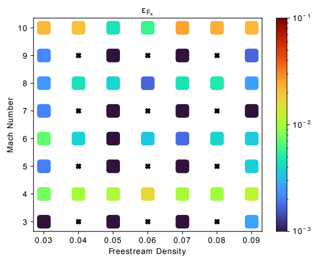
(b) LSPG, $\boldsymbol{W}_{ent}$



(c) ℓ^1, $\boldsymbol{W}_{diag}$



(d) ℓ^1, $\boldsymbol{W}_{ent}$



(e) C(ν)-LSPG, $\nu = 0.001$

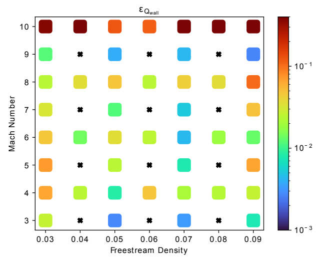
Fig. 3.11: Axial force errors for all ROM variants with grid tailoring

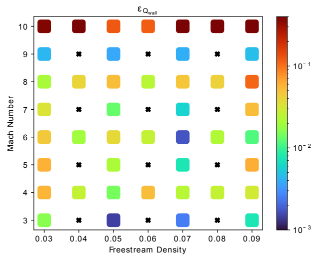
sis. It is worth reevaluating these methods with more complicated flow-fields, such as flows with interacting shocks or nonequilibrium effects. While speedups can be achieved without hyper-reduction for steady ROMs, the extension to unsteady hypersonic test cases will likely require combining these ROMs with hyper-reduction.

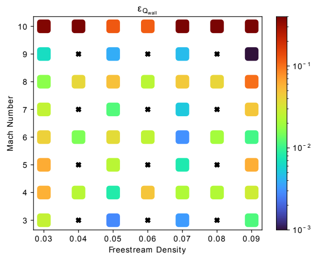
REFERENCES

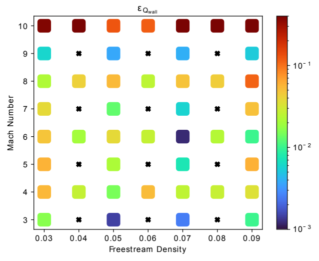
[1] R. Abgrall, D. Amsallem, and R. Crisovan, *Robust model reduction by l1-norm minimization and approximation via dictionaries: application to nonlinear hyperbolic problems*, Advanced Modeling and Simulation in Engineering Sciences, 3 (2016), pp. 1–16.

[2] R. Abgrall and R. Crisovan, *Model reduction using l 1-norm minimization as an application to nonlinear hyperbolic problems*, International Journal for Numerical Methods in Fluids, 87 (2018), pp. 628–651.


(a) LSPG, $\boldsymbol{W}_{diag}$


(b) LSPG, $\boldsymbol{W}_{ent}$


(c) ℓ^1, $\boldsymbol{W}_{diag}$


(d) ℓ^1, $\boldsymbol{W}_{ent}$

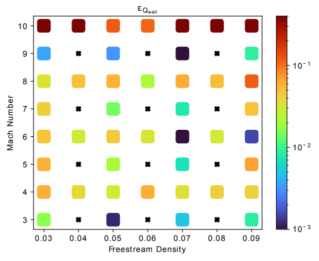

(e) C(ν)-LSPG, $\nu = 0.001$

Fig. 3.12: Heat flux errors for all ROM variants with grid tailoring

[3] M. Balajewicz, I. Tezaur, and E. Dowell, *Minimal subspace rotation on the stiefel manifold for stabilization and enhancement of projection-based reduced order models for the compressible navier–stokes equations*, Journal of Computational Physics, 321 (2016), pp. 224–241.

[4] M. F. Barone, I. Kalashnikova, D. J. Segalman, and H. K. Thornquist, *Stable galerkin reduced order models for linearized compressible flow*, Journal of Computational Physics, 228 (2009), pp. 1932–1946.

[5] P. Benner, S. Gugercin, and K. Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM review, 57 (2015), pp. 483–531.

[6] P. J. Blonigan, F. Rizzi, M. Howard, J. A. Fike, and K. T. Carlberg, *Model reduction for steady hypersonic aerodynamics via conservative manifold least-squares petrov–galerkin projection*, AIAA Journal, 59 (2021), pp. 1296–1312.

[7] G. V. Candler, H. B. Johnson, I. Nompelis, V. M. Gidzak, P. K. Subbareddy, and M. Barnhardt, *Development of the us3d code for advanced compressible and reacting flow simulations*, in 53rd AIAA Aerospace Sciences Meeting, 2015, p. 1893.

[8] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations*, International Journal for numerical methods in engineering, 86 (2011), pp. 155–181.

[9] K. Carlberg, Y. Choi, and S. Sargsyan, *Conservative model reduction for finite-volume models*, Journal of Computational Physics, 371 (2018), pp. 280–314.

[10] J. Chan, *Entropy stable reduced order modeling of nonlinear conservation laws*, Journal of Computational Physics, 423 (2020), p. 109789.

[11] D. S. Ching, P. J. Blonigan, F. Rizzi, and J. A. Fike, *Model reduction of hypersonic aerodynamics with residual minimization techniques*, in AIAA SCITECH 2022 Forum, 2022, p. 1247.

[12] M. Howard, A. Bradley, S. W. Bova, J. Overfelt, R. Wagnild, D. Dinzl, M. Hoemmen, and A. Klinvex, *Towards performance portability in a compressible cfd code*, in 23rd AIAA Computational Fluid Dynamics Conference, 2017, p. 4407.

[13] C. Huang, K. Duraisamy, and C. L. Merkle, *Investigations and improvement of robustness of reduced-order models of reacting flow*, AIAA Journal, 57 (2019), pp. 5377–5389.

[14] C. Huang, C. R. Wentland, K. Duraisamy, and C. Merkle, *Model reduction for multi-scale transport problems using model-form preserving least-squares projections with variable transformation*, Journal of Computational Physics, 448 (2022), p. 110742.

[15] T. J. Hughes, L. P. Franca, and M. Mallet, *A new finite element formulation for computational fluid dynamics: I. symmetric forms of the compressible euler and navier-stokes equations and the second law of thermodynamics*, Computer methods in applied mechanics and engineering, 54 (1986), pp. 223–234.

[16] S. S. Joshi, R. Prasad, and S. Choi, *Discrete empirical interpolation based hyper-reduced order model for steady hypersonic flows*, in AIAA SCITECH 2022 Forum, 2022, p. 0079.

[17] I. Kalashnikova and M. Barone, *Stable and efficient galerkin reduced order models for non-linear fluid flow*, in 6th AIAA Theoretical Fluid Mechanics Conference, 2011, p. 3110.

[18] K. Lee and K. T. Carlberg, *Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders*, Journal of Computational Physics, 404 (2020), p. 108973.

[19] P. Lindsay, J. Fike, I. Tezaur, and K. Carlberg, *Preconditioned least-squares petrov-galerkin reduced order models*, arXiv preprint arXiv:2203.12180, (2022).

[20] M. A. Mirhoseini and M. J. Zahr, *Model reduction of convection-dominated partial differential equations via optimization-based implicit feature tracking*, arXiv preprint arXiv:2109.14694, (2021).

[21] N. J. Nair and M. Balajewicz, *Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks*, International Journal for Numerical Methods in Engineering, 117 (2019), pp. 1234–1262.

[22] E. J. Parish and F. Rizzi, *On the impact of dimensionally-consistent and physics-based inner products for pod-galerkin and least-squares model reduction of compressible flows*, arXiv preprint arXiv:2203.16492, (2022).

[23] B. Peherstorfer, *Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling*, SIAM Journal on Scientific Computing, 42 (2020), pp. A2803–A2836.

[24] E. Phipps and R. Pawlowski, *Efficient expression templates for operator overloading-based automatic differentiation*, in Recent Advances in Algorithmic Differentiation, Springer, 2012, pp. 309–319.

[25] F. Rizzi, P. J. Blonigan, E. J. Parish, and K. T. Carlberg, *Pressio: Enabling projection-based model reduction for large-scale nonlinear dynamical systems*, arXiv preprint arXiv:2003.07798, (2020).

[26] D. Saunders, S. Yoon, and M. Wright, *An approach to shock envelope grid tailoring and its effect on reentry vehicle solutions*, in 45th AIAA Aerospace Sciences Meeting and Exhibit, 2007, p. 207.

[27] T. Taddei, *A registration method for model order reduction: data compression and geometry reduction*, SIAM Journal on Scientific Computing, 42 (2020), pp. A997–A1027.

[28] T. Taddei and L. Zhang, *Space-time registration-based model reduction of parameterized one-dimensional hyperbolic pdes*, ESAIM: Mathematical Modelling and Numerical Analysis, 55 (2021), pp. 99–130.

[29] R. Tezaur, F. As'ad, and C. Farhat, *Robust and globally efficient reduction of parametric, highly nonlinear computational models and real time online performance*, Computer Methods in Applied Mechanics and Engineering, 399 (2022), p. 115392.

[30] K. M. Washabaugh, *Faster fidelity for better design: A scalable model order reduction framework for steady aerodynamic design applications*, PhD thesis, Stanford University, 2016.

MONOLITHIC ALGEBRAIC MULTIGRID PRECONDITIONERS FOR STOKES SYSTEMS

ALEXEY VORONIN*, RAYMOND S. TUMINARO†, LUKE N. OLSON‡, AND SCOTT MACLACHLAN §

Abstract. We investigate a novel monolithic algebraic multigrid solver for the Stokes problem discretized with stable mixed-finite elements. In this work, we build upon a previously published methodology of using geometric multigrid based on the $\mathbb{Q}_1$iso$\mathbb{Q}_2/\mathbb{Q}_1$ discretization as a preconditioner for a higher-order discretization such as $\mathbb{Q}_2/\mathbb{Q}_1$[26]. We introduce a robust algorithm to construct a smoothed aggregation algebraic multigrid solver utilizing independent coarsening of the velocity and pressure fields (the unknown approach) that does not require any geometric information about the system. The effectiveness of this preconditioner is verified for the $\mathbb{P}_2/\mathbb{P}_1$ (Taylor-Hood) discretization in two and three dimensions.

1. Introduction.

Numerical solution of saddle point problems lies at the heart of many scientific and engineering applications. One such problem of interest arises from the finite-element discretization of the Stokes equations, which is used to model viscous fluid flow [6, 4]. Discretization of the Stokes system using stable finite-element pairs usually leads to a 2×2 block structure in which the lower-right diagonal block is a zero matrix. The indefiniteness of such a system and the coupling between the two different unknowns prevents a straightforward application of the classic algebraic and geometric multigrid methods. Monolithic geometric multigrid approaches have been shown to achieve robust convergence for such systems once a successful coupled relaxation scheme and coarsening approach are devised.

Many popular and well understood monolithic multigrid solvers for Stokes problems utilize geometric coarsening of pressure and velocity fields on a nested sequence of increasingly coarser meshes, connected by canonical interpolation operators for each field [2, 1, 5, 25, 17, 20, 14, 13, 7, 11, 16]. Geometric multigrid (GMG) approaches preserve the spatial relationship between the pressure and velocity unknowns and, therefore, the stability of coarse-grid re-discretizations. These approaches are not easily adaptable to algebraic multigrid methods due to the irregular nature of algebraic coarsening. In addition, most inf-sup stable finite-element discretizations of the Stokes equations utilize higher-order basis functions in constructing the discrete operators. The stiffness matrices assembled using these higher-order bases are generally not M-matrices, for which AMG methods were originally intended. As a result, the convergence rates of AMG solvers suffer as the order of such discretizations is increased. Many AMG approaches attempt to bypass the coarsening issues related to higher-order bases by creating algebraic coarsening that mimics the fine-level spatial relationship between the unknowns (for example, maintaining the ratio between the number of degrees of freedom for the two unknowns). These methods frequently rely on some geometric knowledge of the system, such as the location of the unknowns or on which element cells they reside [27, 28, 12, 22, 19].

In this paper, we outline an unknown-based smoothed-aggregation AMG algorithm, where scalar AMG algorithms are applied to each physical unknown individually, in conjunction with Vanka-based relaxation that yields satisfactory convergence rates even for the higher-order Taylor-Hood discretizations of Stokes systems. To combat increased cycle complexity and iteration growth associated with AMG based on higher-order discretizations,

*University of Illinois Urbana-Champaign, voronin2@illinois.edu
†Sandia National Laboratories, rstumin@sandia.gov
‡University of Illinois Urbana-Champaign, lukeo@illinois.edu
§Memorial University of Newfoundland, smaclachlan@mun.ca

we build upon the "double discretization" or "defect correction" scheme that leverages an $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ re-discretization of the original problem, which we discuss in Section 2.1, to construct an effective preconditioner [26]. The work in [26] demonstrates that the related $\mathbb{Q}_1\text{iso}\mathbb{Q}_2/\mathbb{Q}_1$ discretization can be used to build an effective GMG preconditioner for the $\mathbb{Q}_2/\mathbb{Q}_1$ system. The main contribution of this paper is extending this approach to the algebraic multigrid context and demonstrating its effectiveness in preconditioning the $\mathbb{P}_2/\mathbb{P}_1$ Taylor-Hood discretized systems on structured and unstructured grids.

In Section 2 we introduce the Stokes equations and the accompanying discretizations. Section 3 describes a monolithic algebraic multigrid framework and how it is used to assemble preconditioners for the $\mathbb{P}_2/\mathbb{P}_1$ discretization of the Stokes equations. In Section 4 we present the main contribution of this paper: AMG convergence results for various Stokes problems and a comparison of the computational cost between the $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$-based and $\mathbb{P}_2/\mathbb{P}_1$-based AMG preconditioners. Section 5 provides concluding remarks and potential future research directions.

2. Discretization of the Stokes Equations.

In this paper, we consider the steady-state Stokes problem on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d, d \in \{2,3\}$ with boundary $\partial\Omega = \Gamma_N \cup \Gamma_D$, given by

$$\begin{aligned} -\nabla^2 \boldsymbol{u} + \nabla p &= \boldsymbol{f} \quad \text{in } \Omega && (2.1a)\\ -\nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega && (2.1b)\\ \boldsymbol{u} &= \boldsymbol{w} \quad \text{on } \Gamma_D && (2.1c)\\ \frac{\partial \boldsymbol{u}}{\partial n} - \boldsymbol{n}p &= \boldsymbol{s} \quad \text{on } \Gamma_D, && (2.1d) \end{aligned}$$

where $\boldsymbol{u}$ is the velocity of the fluid, p is the pressure, $\boldsymbol{f}$ is a forcing term, $\boldsymbol{n}$ is the outward pointing normal to $\partial\Omega$, $\boldsymbol{w}$ and $\boldsymbol{s}$ are given boundary data, and Γ_D and Γ_N are disjoint Dirichlet and Neumann boundaries, respectively. In cases where velocity is specified everywhere along the boundary, requiring $\int_{\partial\Omega} \boldsymbol{w} \cdot \boldsymbol{n} = 0$, the pressure solution is unique only up to a constant. The suitable trial space for pressure is $\mathcal{Q} := L^2_0(\Omega)$. When $|\Gamma_N| > 0$, the typical choice of boundary data satisfying (2.1d) is $\boldsymbol{s} = \boldsymbol{0}$. The resulting solution is always unique under the suitable trial space for pressure, $\mathcal{Q} := L^2(\Omega)$ [6].

We define finite dimensional spaces $(\boldsymbol{\mathcal{V}}_h^{\boldsymbol{w}}, \mathcal{Q}_h) \subset (\boldsymbol{\mathcal{H}}^1_{\boldsymbol{g}}(\Omega), \mathcal{Q})$, where $\boldsymbol{\mathcal{V}}_h^{\boldsymbol{w}}$ satisfies the Dirichlet boundary conditions specified by (2.1c). The resulting discrete weak formulation of (2.1) is: find $\boldsymbol{u} \in \boldsymbol{\mathcal{V}}_h^w$ and $p \in \mathcal{Q}_h$ such that

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{v}) + b(p, \boldsymbol{v}) &= F(\boldsymbol{v}) && (2.2a)\\ b(q, \boldsymbol{u}) \qquad\qquad &= 0, && (2.2b) \end{aligned}$$

for all $q \in \mathcal{Q}_h$ and $\boldsymbol{v} \in \boldsymbol{\mathcal{V}}_h^0$. Here, $a(\cdot,\cdot)$ and $b(\cdot,\cdot)$ are bilinear forms and $F(\cdot)$ is a linear form given by

$$a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v}, \quad b(p, \boldsymbol{v}) = -\int_\Omega p \nabla \cdot \boldsymbol{v}, \quad F(\boldsymbol{v}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} + \int_{\Gamma_N} \boldsymbol{s} \cdot \boldsymbol{v}.$$

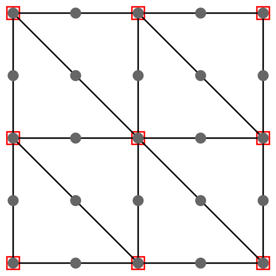
From here on out, we overload the notation and use $\boldsymbol{u}$ and p to denote the discrete velocity and pressure unknowns. Given an inf-sup stable choice of finite-dimensional spaces $(\boldsymbol{\mathcal{V}}_h^0, \mathcal{Q}_h)$, we obtain a saddle-point system of the form

$$K \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ 0 \end{bmatrix} = b, \qquad (2.3)$$

where matrix A corresponds to the discrete vector-Laplacian, and B represents the negative of the discrete divergence operator.

2.1. Finite Element Discretizations and Meshes. We focus on two different but related discretizations for the finite-dimensional spaces $(\boldsymbol{\mathcal{V}}_h, \mathcal{Q}_h)$ on d-dimensional simplexes $(\mathcal{T}_h)$. The first discretization is the Taylor-Hood (TH) $\mathbb{P}_k/\mathbb{P}_{k-1}$ discretization, which uses continuous piecewise polynomials of degree k for the velocity space $\boldsymbol{\mathcal{V}}_h$ and continuous piecewise polynomials of degree $k-1$ for the pressure space $\mathcal{Q}_h$. The TH element-pair is inf-sup stable for $k \geq 2$ on any given mesh $\mathcal{T}_h$ obtained by triangularization (tetrahedralization) of the domain Ω. For the remainder of this paper, we use $k = 2$ for the TH $(\mathbb{P}_2/\mathbb{P}_1)$ element pair, which corresponds to a piecewise quadratic approximation of the velocity components and a piecewise linear approximation of the pressure. This discretization is used to assemble the discrete Stokes system, Equation (2.3), which we are interested in solving. However, the preconditioning approach described in Section 3 can easily be extended to higher-order TH $(k > 2)$ discretizations.

The second discretization used in this paper is the $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$. It replaces the $\mathbb{P}_2$ space for velocities with an $\mathbb{P}_1$ approximation on a uniform refined mesh, $\mathcal{T}_{h/2}$ while keeping the same pressure space and mesh $(\mathcal{T}_h)$. $\mathcal{T}_{h/2}$ is obtained by overlaying the higher-order nodes $(\mathbb{P}_2)$ with a lower-order mesh $(\mathcal{T}_{h/2})$, which is depicted by dotted lines in Figure 2.1(b). $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ discretization is also inf-sup stable [8] but is not as well-known due to its relatively low accuracy and computational efficiency.


(a) $\mathbb{P}_2/\mathbb{P}_1$

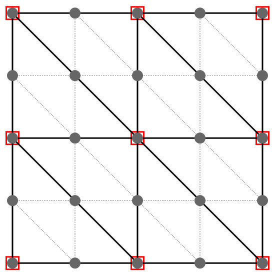

(b) $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$

FIG. 2.1. *Meshes and degrees of freedom for the* $\mathbb{P}_2/\mathbb{P}_1$ *and* $\mathbb{P}_1$*iso*$\mathbb{P}_2/\mathbb{P}_1$ *discretizations. Dark circles correspond to the velocity locations (two velocity components per marker), and red squares correspond to the pressure locations.*

3. Monolithic Multigrid. This section proposes an algorithm for constructing a monolithic AMG solver for the Stokes system $Kx = b$ described in Section 2. This approach follows the familiar multi-level error-reduction strategy present in all multigrid methods. At the foundation of successful multigrid methods lies robust relaxation schemes and complementary interpolation operators. The relaxation scheme must quickly reduce the high-frequency error associated with the initial solution guess, leaving the near-null space modes mostly unaffected. The interpolation operator is then used to represent the residual equation on a coarser set of unknowns (coarse grid), where the correction is easier to compute. The coarse-level solution is interpolated to the original (fine-level) grid and is used to correct the initial guess.

For a linear system $K_\ell x_\ell = b_\ell$ on level ℓ of the multigrid hierarchy, we define the relaxation scheme as a fixed-point iteration

$$x_\ell^{j+1} = (I - \omega_\ell M_\ell^{-1} K_\ell) x_\ell^j + \omega_\ell M_\ell^{-1} b_\ell,$$

where M_ℓ is an inexpensive approximation to K_ℓ, whose inverse is easy to apply. The $(I - \omega_\ell M_\ell^{-1} K_\ell)$ term is the error-propagation operator for relaxation on K_ℓ with a damping parameter ω_ℓ. Combing one pre- and one post-relaxation sweeps with the coarse-grid correction operator, we obtain the following two-grid error-propagation operator

$$G_\ell = (I - \omega_\ell M_\ell^{-1} K_\ell)(I - P_\ell K_{\ell+1}^{-1} R_\ell K_\ell)(I - \omega_\ell M_\ell^{-1} K_\ell), \tag{3.1}$$

where the P_ℓ is the interpolation between levels $\ell + 1$ and ℓ, and $K_{\ell+1}$ is the coarse-grid operator computed via Galerkin coarsening $K_{\ell+1} = P_\ell^T K_\ell P_\ell$. The hierarchy is usually recursively extended until $K_{\ell+1}^{-1}$ is small enough to be computed directly via the Moore-Penrose pseudo-inverse or sparse LU decomposition.

3.1. Low-Order Preconditioning. Voronin et al. [26] demonstrate that the $\mathbb{Q}_1$iso$\mathbb{Q}_2/\mathbb{Q}_1$ finite-element discretization can be used to construct a robust geometric multigrid (GMG) preconditioner for the higher-order $\mathbb{Q}_2/\mathbb{Q}_1$ finite-element discretization of the Stokes equations. The paper utilized a multigrid defect-correction approach where the higher-order ($\mathbb{Q}_2/\mathbb{Q}_1$) finite-element system was preconditioned with GMG based on the lower-order system ($\mathbb{Q}_1$iso$\mathbb{Q}_2/\mathbb{Q}_1$). In this paper, we leverage these results in developing a defect-correction method based on a monolithic AMG solver.

This defect-correction approach follows the same notation described in [26]. Let K_0 and K_1 be the Stokes operators assembled using a higher-order ($\mathbb{P}_2/\mathbb{P}_1$) and lower-order ($\mathbb{P}_1iso\mathbb{P}_2/\mathbb{P}_1$) discretization. P_0 and R_0 are the grid transfer operators between higher-order and lower-order discretization, which are to be defined later. Since the K_1 system is still the same size linear system, we form a monolithic AMG hierarchy based on K_1, denoted G_1, to help accelerate its convergence. Adding ν_1 pre- and ν_2 post- relaxation sweeps on the K_0 system results in the defect-correction method

$$E = \left(I - \omega_0 M_0^{-1} K_0\right)^{\nu_2} \left(I - \tau(I - G_1^\gamma) K_1^{-1} K_0\right) \left(I - \omega_0 M_0^{-1} K_0\right)^{\nu_1}, \tag{3.2}$$

where M_0 is a relaxation operator based in the K_0 system and ω_0 is the respective relaxation parameter.

The grid transfer operator, P_0, between higher-order and lower-order discretization can be represented as a block-diagonal matrix

$$P_0 = \begin{bmatrix} P_{\boldsymbol{u}} & \\ & P_p \end{bmatrix}, \tag{3.3}$$

where $P_{\boldsymbol{u}}$ and P_p correspond to velocity and pressure field interpolation operators, respectively. The velocity field operators in the Stokes equations discretized by $\mathbb{P}_2$ and $\mathbb{P}_1$iso$\mathbb{P}_2$ elements on corresponding meshes are of the same size, and their DoFs are collocated in physical space. We can, thus, take $P_{\boldsymbol{u}} = I$, where I is a square-matrix identity operator. The same is true for the pressure-field grid-transfer operator for the $\mathbb{P}_2/\mathbb{P}_1$ and $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ discretizations, $P_p = I$. This implies that K_1 is a non-Galerkin coarse-grid approximation of K_0. However, the coarser levels of the hierarchy $K_{\ell>1}$ contained within G_1 are Galerkin coarse-grid operators built from K_1.

3.2. Coarse-level Stokes by Smoothed Aggregation. This subsection outlines an algorithm for constructing a smoothed aggregation (SA) based AMG hierarchy for the Stokes system. This AMG hierarchy is equivalent to the multi-level operator G_ℓ described in (3.1). Algorithm 1 is an unknown-based approach where interpolation operators are formed for each unknown field by performing SA for each matrix-block corresponding to the unknown field. Algorithm 1 relies on the user's ability to access individual blocks of the Stokes systems, Equation (2.3), or infer what those discrete operators are based on the knowledge of the total number of unknowns in each field. Notice in Algorithm 1 that a pressure stiffness matrix, A_p, is also supplied in addition to the Stokes operator. A_p is used to construct the pressure interpolation operator. However, if the pressure-stiffness matrix is unavailable, it can be approximated by a post-processed $\hat{A}_p = BB^T$ operator, where the stencil width is trimmed to match the one of A_p [22].

SA is applied to each scalar field independently to form the grid-transfer operators. Algorithm 1 outlines velocity-field coarsening in a component-wise fashion, which requires breaking up the velocity vector-Laplacian, A, into its components, A_x and A_y. However, this is not strictly necessary in practice. In the case of the velocity field, its scalar components are invariant subspaces, which means SA can be applied directly to vector-Laplacian, given that the supplied near-nullspace modes contain two (three) linearly-independent constant modes corresponding to each block of the 2D (3D) vector-Laplacian, which can be represented as

$$\mathcal{N}_v = \begin{bmatrix} \mathcal{N}_x & \\ & \mathcal{N}_y \end{bmatrix}, \tag{3.4}$$

where $\mathcal{N}_x$ and $\mathcal{N}_y$ are two constant vectors of size $n^{(v)}$, corresponding to the total number of velocity nodes. For the pressure field, SA is applied directly to the pressure-stiffness matrix, A_p, or its approximation along with a constant near-nullspace mode, $\mathcal{N}_p$, of size $n^{(p)}$ corresponding to the total number of pressure nodes.

After the interpolation operators for each field (velocity and pressure) are formed, they are combined as a block-diagonal matrix into a single monolithic interpolation operator, P_ℓ. Algorithm 1 describes an AMG algorithm for a 2D Stokes system, which can easily be extended to 3D by simply adding the z-component of velocity interpolation (P^z) to the block-diagonal interpolation operator P_ℓ.

Algorithm 1 SA-AMG Hierarchy Setup for Stokes System

```
1:  Input: $K_1 = \begin{bmatrix} A & B^T \\ B & \end{bmatrix}$, Stokes system where $B \in \mathbb{R}^{n^{(p)} \times (d \cdot n^{(v)})}$
2:         $A_p$, pressure stiffness matrix
3:  Output: $K_1, \ldots, K_{\ell_{max}}$, Grid hierarchy
4:          $P_1, \ldots, P_{\ell_{max}-1}$, Interpolation Operators
5:
6:  $A_x,\ A_y = \text{split(A)}$                 // Split vector-Laplacian component-wise
7:  $\mathcal{N}_x = \mathcal{N}_y = \mathbf{1}_{n^{(v)}}$          // Near-nullspace velocity mode
8:  $\mathcal{N}_p = \mathbf{1}_{n^{(p)}}$                // Near-nullspace pressure mode
9:  $\{P_\ell^x\} = \text{sa\_amg}(A_x, \mathcal{N}_x)$      // interpolation for x-component of velocity
10: $\{P_\ell^y\} = \text{sa\_amg}(A_y, \mathcal{N}_y)$      // interpolation for y-component of velocity
11: $\{P_\ell^p\} = \text{sa\_amg}(A_p, \mathcal{N}_p)$      // interpolation for pressure
12: $\ell_{max} = \min(\text{size}(\{P_\ell^x\}), \text{size}(\{P_\ell^y\}), \text{size}(\{P_\ell^p\})) + 1$
13: for $\ell = 1, \ldots, \ell_{max} - 1$ do
14:     $P_\ell = \begin{bmatrix} P_\ell^x & & \\ & P_\ell^y & \\ & & P_\ell^p \end{bmatrix}$,   // assemble monolithic interpolation operator
15:     $K_{\ell+1} = P_\ell^T K_\ell P_\ell$            // Compute coarse-grid operator
16: end for
```

While the smoothed aggregation of each field (lines 9-11 in Algorithm 1) is not explicitly coordinated (as in most traditional monolithic MG methods), it is crucial to maintain a similar ratio between velocity and pressure DoFs throughout the hierarchy to improve the stability of the coarse-grids operators. This can be achieved by matching the coarsening rates for each field approximately. The coarsening rates are determined by the strength of connection (SoC) measures, which determine the matrix graph edges that should be ignored during the coarsening algorithm. We have found that this approximate matching does not tend to occur with traditional SoC measures. Instead, we employ an evolution SoC that we discovered does much better at maintaining a consistent coarsening rate across velocity and pressure grids than the more standard symmetric SoC. Evolution SoC produces more accurate interpolation by integrating the local knowledge of algebraically smooth error with the local behavior of interpolation [21]. Evolution SoC results in better convergence rates across different sets of problems in a parameter-independent fashion, unlike the symmetric SoC-based hierarchies, whose results are significantly more sensitive to the drop-tolerance parameter.

3.3. Relaxation. The final piece of the monolithic AMG method for Stokes is the coupled relaxation method M_ℓ^{-1}. In this section, we drop the operator subscript ℓ since the relaxation operator construction is the same on all levels of the multigrid hierarchy.

For this paper, we use the algebraic version of the additive Vanka method [24], but other coupled relaxation schemes should also work here. Vanka relaxation is an additive Schwarz-type relaxation method for saddle-point systems. It breaks up the global saddle-point system K into many small (overlapping) saddle-point systems, denoted K_i^{loc}, that need to be solved exactly and whose solutions are summed back up into a global update vector.

We construct a Vanka patch K_i^{loc} around each one of $n^{(p)}$ pressure DoFs. For each pressure DoF (p_i), we select the algebraically connected velocity DoFs set ($\{v_i\}_{i=0}^{n_i^{(v)}}$), which can be identified using the sparsity pattern of the divergence matrix's row corresponding to p_i. The global sparsity pattern ($n_i^{(v)}$ column indices) of the local divergence operator B_i^{loc} is then used to extract the velocity coefficients from the global vector-Laplacian to form a local vector-Laplacian operator, A_i^{loc}. The coefficients of the local divergence operator, B_i^{loc}, correspond to the non-zero values of B's row i. Local divergence and vector-Laplacian operators form a local saddle-point problem

$$K_i^{loc} = \begin{bmatrix} A_i^{loc} & (B_i^{loc})^T \\ B_i^{loc} & 0 \end{bmatrix}. \tag{3.5}$$

Figure 3.1 depicts what typical patches look for the $\mathbb{P}_2/\mathbb{P}_1$ and $\mathbb{P}_1\text{iso}\mathbb{P}_2$ discretizations on the fine-level of the hierarchy.

The injection operator V_i^T is an $m_i \times m$ binary matrix that extracts the subset of the global saddle point DoFs coinciding with the i^{th} patch. Here, m is the number of rows/columns of K while m_i is $n_i^{(v)} + 1$, the number of DoFs contained in K_i^{loc}. It follows that $K_i^{loc} = V_i K V_i^T$ and therefore is the sub-matrix of K associated with the i^{th} patch.

A single sweep of Vanka relaxation can then be expressed as

$$M^{-1} = \sum_{i=1}^{n^{(p)}} V_i^T W_i \left(V_i K V_i^T\right)^{-1} V_i,$$

where the diagonal weighting matrix W_i is a partition of unity, where each diagonal entry is equal to the reciprocal of the number of patches that contain the associated degree of freedom.

In the context of this paper, the solution of each patch K_i is performed using the Moore-Penrose pseudo-inverse of a matrix. For some patch sparsity patterns, sparse LU factorization can result in faster setup and solution time. To minimize the number of

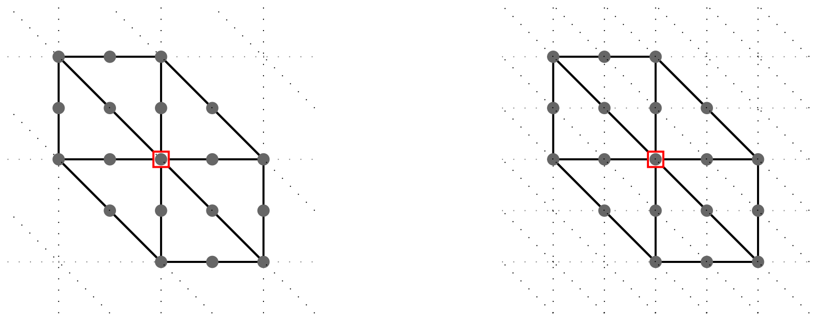

(a) $\mathbb{P}_2/\mathbb{P}_1$ (b) $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$

Fig. 3.1. *On the left, construction of a Vanka patch for the $\mathbb{P}_2/\mathbb{P}_1$ discretization. On the right, construction of a Vanka patch for the $\mathbb{P}_1 iso\mathbb{P}_2/\mathbb{P}_1$ discretization. Both patches include a single pressure DoF (red square) and all the velocity DoFs directly algebraically connected to that pressure DoF (black circles). These patches have an element/mesh interpretation on the finest mesh, while on algebraically coarsened levels, there is no simple element/mesh interpretation.*

potential optimization strategies in this paper, we always use the pseudo-inverse for the patch-system solution.

3.4. Relaxation Parameter Space Reduction. Vanka relaxation methods are often relatively sensitive to the choice of damping-parameter ω, which is needed for each level within the AMG hierarchy. Fourier analysis has often been used to choose this damping parameter when Vanka is employed within a GMG scheme [9]. As Fourier analysis is generally restricted to structured mesh computations, the traditional Fourier space approach is not possible for the algebraically generated irregular coarse meshes. Two possible alternatives based on polynomial preconditioning ideas come to mind to address the damping parameter choice in an algebraic setting.

The first idea is based on the Chebyshev polynomials, while the second is based on the GMRES polynomial [18]. In the case of Chebyshev polynomials, one needs to have an estimate for the largest and smallest magnitude eigenvalues of KM^{-1}, which can be problematic. Often a fixed fraction of the largest magnitude eigenvalue can be used for the smallest magnitude eigenvalue estimate when the Chebyshev polynomial is used as a multigrid smoother. As an alternative, we consider an approach analogous to the Chebyshev iteration to determine an appropriate damping parameter. Instead of performing k-iterations of Vanka relaxation with a fixed damping parameter ω, we construct a residual-minimizing polynomial based on the right-preconditioned Krylov-subspace of size n,

$$\mathcal{K}_n(KM^{-1}, r_0) = \{KM^{-1}r_0, \ldots, (KM^{-1})^n r_0\}, \tag{3.6}$$

where K is the Stokes system, M^{-1} is a single sweep of stationary Vanka relaxation, and r_0 is the initial residual. This automatic approach requires no particular eigenvalue calculations as it is fully equivalent to n iterations of FGMRES preconditioned with a single sweep ($k = 1$) of Vanka. GMRES can also be used here in place of FGMRES. For the remainder of the paper, we fix $k = 1$ relaxation steps per Krylov step and $n = 2$ FGMRES steps per relaxation.

The Chebyshev iteration and FGMRES methods have unique advantages and disadvantages when it comes to improving the relaxation convergence rates. The Chebyshev method requires the user to provide eigen-bounds for the preconditioned K matrix. In contrast, the FGMRES-based approach computes the eigen-bounds implicitly and forms an interpolating polynomial that is optimal from the perspective of residual minimization. However, the FGMRES-based polynomial may not be optimal from the multigrid point of view. Multigrid relaxation methods are generally optimized to reduce the high-frequency error range while leaving the slow-to-converge modes for the coarse-grid correction. Chebyshev iteration allows for relaxation-range tuning, resulting in stricter convergence bounds, while GMRES does not allow for this type of tuning.

The additional computational cost associated with wrapping Vanka relaxation with FGMRES is negligible relative to the cost of Vanka relaxation. However, from the high-performance computing standpoint, Chebyshev iteration can be seen as more efficient because it avoids additional inner products, which FGMRES needs. For this paper, we simply use FGMRES to automate the choice of Vanka's damping parameter, ω.

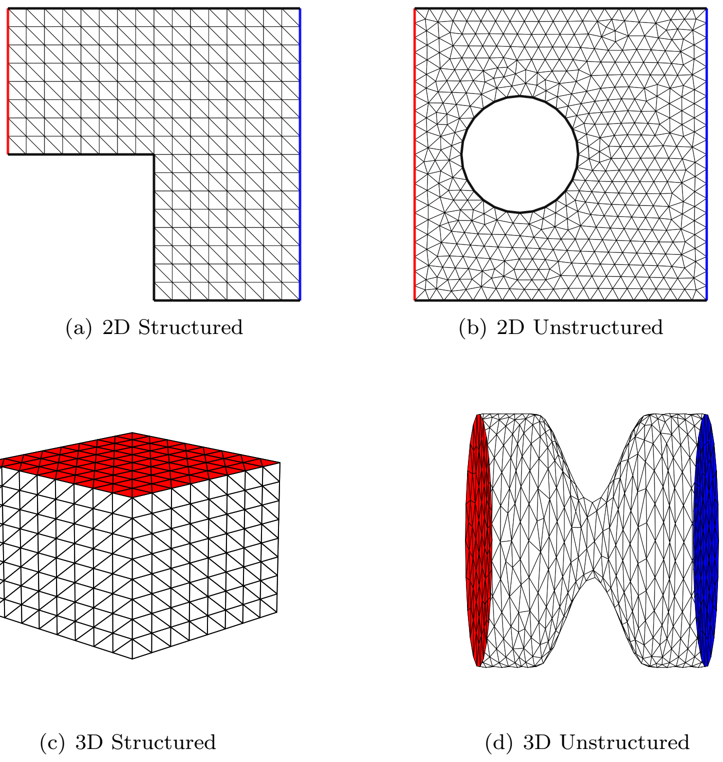
4. Numerical Results.

In this section, we evaluate the effectiveness and the relative computational cost of the monolithic AMG preconditioners constructed using Algorithm 1 and the defect-correction approach (3.2). We compare the iterations to the convergence of FGMRES preconditioned with AMG for the structured and unstructured 2D and 3D $\mathbf{P}_2/\mathbb{P}_1$ systems. Use of both $\mathbf{P}_2/\mathbb{P}_1$ and $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$ operators is explored in the construction of the AMG hierarchy based on (3.1). The $\mathbf{P}_2/\mathbb{P}_1$ and $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$ systems are assembled with the help of Firedrake (version 0.13.0) [23, 15]. While Firedrake does not currently provide the capability to directly construct $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$ matrices, they can be formed by assembling the $\mathbf{P}_2/\mathbb{P}_1$ discretization on a once refined mesh and then coarsening-in-order the velocity field and coarsening-in-space to the pressure field. The AMG preconditioners are implemented in the custom codebase by utilizing multigrid functionality offered by the PyAMG library (version 4.2.1) [3] to combine Algorithm 1 with Vanka relaxation methods. The recorded iteration counts correspond to the total FGMRES iterations required to reduce the l^2 norm of the residual by a factor of 10^{-10}. The reported times to solution are based on the lowest time from 5 different measurements.

We introduce notation M_h and M_{ph} to distinguish between multigrid preconditioners corresponding to G_0 and E operators, respectively. The M_h preconditioners result from applying Algorithm 1 directly to the higher-order discretization (K_0: $\mathbf{P}_2/\mathbb{P}_1$). The defect-correction approaches (E) which apply Algorithm 1 to the lower-order discretization (K_1: $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$) are denoted as M_{ph}, whose subscripts indicate coarsening in order (p) and followed by coarsen in space (h). In case of M_{ph} preconditioners we explore three different fine-level (K_0 and K_1) relaxation approaches that are abbreviated as $M_{ph}^{\omega_i>0}$, $M_{ph}^{\omega_0=0}$, and $M_{ph}^{\omega_1=0}$. $M_{ph}^{\omega_i>0}$ performs pre- and post-relaxation on each level of the hierarchy ($\omega_i = 1$ for $i \in [0, n_{lvls})$), except for the coarsest-level (n_{lvls}), where exact-solve takes place. $M_{ph}^{\omega_0=0}$ skips the relaxation on the K_0 system, but performs pre- and post-relaxation on the remaining levels ($\omega_i = 1$ for $i \in [1, n_{lvls})$). $M_{ph}^{\omega_1=0}$ skips the relaxation on the K_1 system, but performs relaxation on the K_0 system and the remaining levels of the $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$ hierarchy ($\omega_i = 1$ for $i \in [0] \cup [2, n_{lvls})$).

The optimal τ, $\mathbf{P}_1\text{iso}\mathbf{P}_2/\mathbb{P}_1$-cycle damping parameter is selected for each problem type using brute-force parameter scan, where τ_{opt} corresponds to the smallest number of iterations to convergence for the largest (most-refined) problem.

We consider 2D and 3D structured and unstructured meshes for the test problems with two types of boundary conditions. The first type of boundary conditions corresponds to lid driven-cavity flow boundary problem [6], illustrated in Figure 4.1(c). The second type of problems consists of parabolic inflow boundary conditions and natural outflow (Neumann) boundary condition, displayed in Figures 4.1(a), 4.1(b) and 4.1(d). The structured meshes are constructed using Firedrake, and the unstructured meshes are constructed using the Gmsh meshing software [10].


(a) 2D Structured (b) 2D Unstructured

(c) 3D Structured (d) 3D Unstructured

Fig. 4.1. *Meshes used to construct the Stokes problems. The boundaries are marked with three colors (red, blue, and black), corresponding to three types of velocity field boundary conditions. Red corresponds to non-zero Dirichlet boundaries. Blue corresponds to Neumann boundary conditions. Black indicates zero Dirichlet boundary condition.*

4.1. Preconditioning $\mathbb{P}_2/\mathbb{P}_1$ Systems. In this section, we examine the performance of the preconditioners on the $\mathbb{P}_2/\mathbb{P}_1$ discretizations of the Stokes problems described above. Figure 4.2 summarizes the convergence data collected with damping parameters τ displayed in Table 4.1. First, consider the preconditioner based on the $\mathbb{P}_2/\mathbb{P}_1$ hierarchy of grids, M_h. For most problems, M_h exhibits minor growth in iteration count as the problems are refined. Since we expect the convergence to degrade in the case of higher-order Taylor-Hood discretizations (not shown here), we also evaluate defect-correction approaches (M_{ph}) based $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ discretization, which consists of more amicable linear finite-element bases.

$M_{ph}^{\omega_1=0}$, which effectively replaces M_h's coarse-grids with $\mathbb{P}_1$iso$\mathbb{P}_2/\mathbb{P}_1$ coarse-grids, converges in a similar number of iterations as the M_h-based solvers. Since the $M_{ph}^{\omega_1=0}$ coarse-grids are sparser than M_h's, the relative cost of the solution is slightly cheaper, which can be seen in the middle and bottom rows of Figure 4.2. However, the times to solution are still pretty similar for M_h and $M_{ph}^{\omega_1=0}$, because the overall solve time is dominated by the relaxation cost on the finest level. In the case of $M_{ph}^{\omega_0=0}$, relaxation on the K_0 system is skipped and replaced by the relaxation on the K_1 system. The convergence of $M_{ph}^{\omega_0=0}$-based solvers suffers for all problem types, which can be seen in the top row of Figure 4.2. This effect is particularly drastic for 3D problems. Despite having smaller cost per iteration than M_h, the increase in iteration count makes $M_{ph}^{\omega_0=0}$ less computationally effective than $M_{ph}^{\omega_1=0}$ and M_h. This can be seen in the last two columns of Figure 4.2. The last preconditioner,

$M_{ph}^{\omega_i>0}$, performs relaxation on all levels of the hierarchy. While $M_{ph}^{\omega_i>0}$-based solver converges in fewer iterations for most problem types, there is a possible decrease in robustness or asymptotic approach to $M_{ph}^{\omega_1=0}$'s convergence as the problem size is refined. Due to the doubling in the number of relaxation sweeps on fine levels K_0 and K_1, the time to solution for this approach is significantly longer than for the M_h approach.

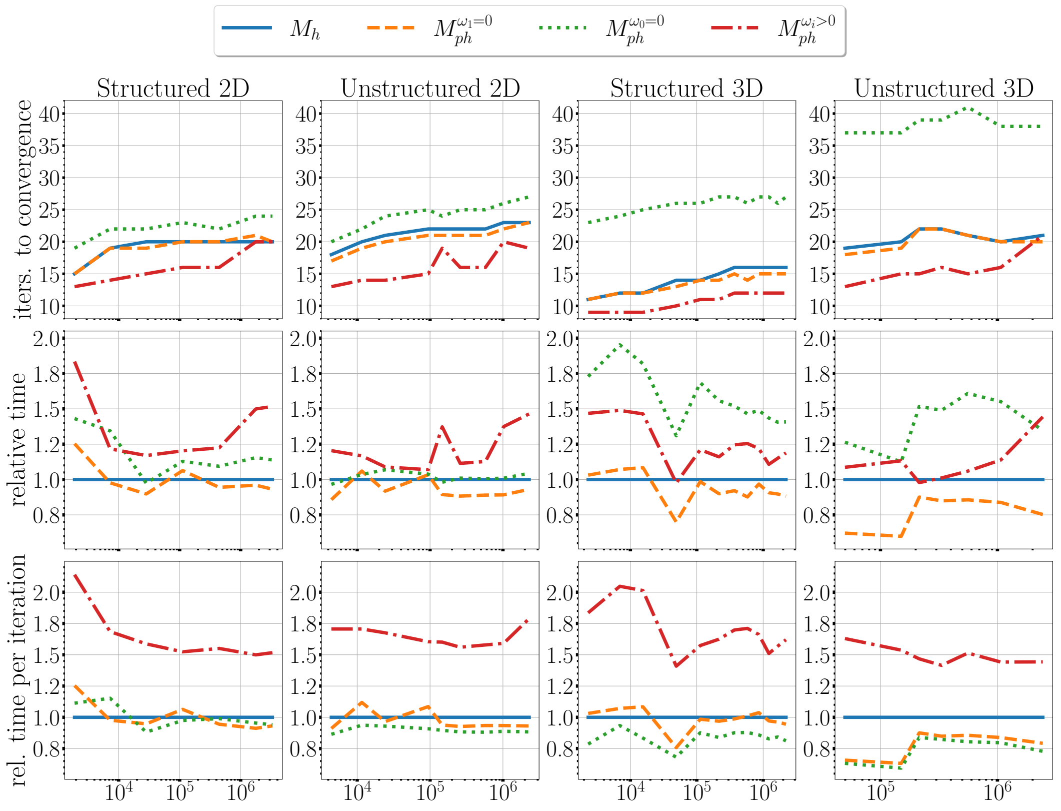


FIG. 4.2. *Iterations to convergence, relative time to solution, and relative time per iteration. The timings are relative with respect to M_h solver. The x axis gives the number of degrees-of-freedom.*

TABLE 4.1
Optimal τ damping parameters used in the preconditioners.

	Mesh Type			
	Structured		Unstructured	
Preconditioner	2D	3D	2D	3D
$M_{ph}^{\omega_1=0}$	0.84	1.00	0.90	0.95
$M_{ph}^{\omega_0=0}$	1.00	1.00	1.00	1.00
$M_{ph}^{\omega_i>0}$	0.79	0.95	0.81	0.89

Convergence and timings results inFigure 4.2, especially for $M_{ph}^{\omega_0=0}$ and $M_{ph}^{\omega_1=0}$, demonstrate conclusively that the relaxation on the original problem (K_0) should not be skipped when designing a defect-correction type AMG preconditioner. Additional relaxation on the

K_1 system can sometimes help improve the convergence further but at a much higher computational cost. This conclusion is supported by the numerical results for the GMG-based defect-correction approach published in [26].

5. Conclusion. This report develops a novel algorithm for constructing monolithic AMG preconditioners for the Stokes problem discretized with $\mathbb{P}_2/\mathbb{P}_1$ elements. We demonstrate that this algorithm can be applied directly to the $\mathbb{P}_2/\mathbb{P}_1$ to produce a robust solver. We also explore a defect-correction approach where we assemble a monolithic AMG solver based on the lower-order $\mathbb{P}_1\text{iso}\mathbb{P}_2/\mathbb{P}_1$ discretization to precondition the higher-order system. Both methods demonstrate a relatively steady iteration count for various Stokes problems on structured and unstructured meshes in 2D and 3D. In future work, we intend to use the defect-correction using $\mathbb{P}_1\text{iso}\mathbb{P}_2$ discretization to construct robust preconditioners for the higher-order Taylor-Hood and Scott-Vogelious discretizations.

REFERENCES

[1] J. H. Adler, T. Benson, E. C. Cyr, P. E. Farrell, S. MacLachlan, and R. Tuminaro, *Monolithic multigrid for magnetohydrodynamics*, SIAM J. Sci. Comput., 43 (2021), pp. S70–S91.

[2] J. H. Adler, T. R. Benson, and S. P. MacLachlan, *Preconditioning a mass-conserving discontinuous Galerkin discretization of the Stokes equations*, Numerical Linear Algebra with Applications, 24 (2017), p. e2047.

[3] N. Bell, L. N. Olson, and J. Schroder, *PyAMG: Algebraic multigrid solvers in python*, Journal of Open Source Software, 7 (2022), p. 4142.

[4] M. Benzi, G. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[5] A. Brandt, *Multigrid techniques: 1984 guide with applications to fluid dynamics*, GMD–Studien Nr. 85, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[6] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Oxford University Press, USA, 2014.

[7] M. Emami, *Efficient multigrid solvers for the Stokes equations using finite elements*, Lehrstuhl für Informatik 10 (Systemsimulation), (2013).

[8] A. Ern and J.-L. Guermond, *Theory and Practice of Finite Elements*, vol. 159 of Applied Mathematical Sciences, Springer-Verlag New York, 2004.

[9] P. E. Farrell, Y. He, and S. P. MacLachlan, *A local Fourier analysis of additive Vanka relaxation for the Stokes equations*, Numerical Linear Algebra with Applications, 28 (2021), p. e2306.

[10] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities*, International Journal for Numerical Methods in Engineering, 79 (2009), pp. 1309–1331.

[11] B. Gmeiner, M. Huber, L. John, U. Rüde, and B. Wohlmuth, *A quantitative performance study for Stokes solvers at the extreme scale*, J. Comput. Sci., 17 (2016), pp. 509–521.

[12] A. Janka, *Smoothed aggregation multigrid for a Stokes problem*, Comput. Vis. Sci., 11 (2008), pp. 169–180.

[13] V. John, P. Knobloch, G. Matthies, and L. Tobiska, *Non-nested multi-level solvers for finite element discretisations of mixed problems*, Computing, 68 (2002), pp. 313–341.

[14] V. John and L. Tobiska, *Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations*, International Journal For Numerical Methods In Fluids, 33 (2000), pp. 453–473.

[15] R. C. Kirby and L. Mitchell, *Solver composition across the PDE/linear algebra barrier*, SIAM Journal on Scientific Computing, 40 (2018), pp. C76–C98.

[16] M. Larin and A. Reusken, *A comparative study of efficient iterative solvers for generalized Stokes equations*, Numer. Linear Algebra Appl., 15 (2008), pp. 13–34.

[17] J. Linden, G. Lonsdale, B. Steckel, and K. Stüben, *Multigrid for the steady-state incompressible Navier-Stokes equations: a survey*, in 11th International Conference on Numerical Methods in Fluid Dynamics (Williamsburg, VA, 1988), vol. 323 of Lecture Notes in Phys., Springer, Berlin, 1989, pp. 57–68.

[18] J. A. Loe and R. B. Morgan, *Toward efficient polynomial preconditioning for GMRES*, Numerical Linear Algebra with Applications, (2021), p. e2427.

[19] B. Metsch, *Algebraic multigrid (AMG) for saddle point systems*, PhD thesis, Universitäts-und Landesbibliothek Bonn, 2013.

[20] A. Niestegge and K. Witsch, *Analysis of a multigrid Stokes solver*, Appl. Math. Comput., 35 (1990), pp. 291–303.

[21] L. N. Olson, J. Schroder, and R. S. Tuminaro, *A new perspective on strength measures in algebraic multigrid*, Numerical Linear Algebra with Applications, 17 (2010), pp. 713–733.

[22] A. Prokopenko and R. S. Tuminaro, *An algebraic multigrid method for Q_2-Q_1 mixed discretizations of the Navier-Stokes equations*, Numerical Linear Algebra with Applications, 24 (2017), p. e2109.

[23] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. Mcrae, G.-T. Bercea, G. R. Markall, and P. H. J. Kelly, *Firedrake: Automating the finite element method by composing abstractions*, ACM Trans. Math. Softw., 43 (2016), pp. 24:1–24:27.

[24] J. Schöberl and W. Zulehner, *On Schwarz-type smoothers for saddle point problems*, Numerische Mathematik, 95 (2003), pp. 377–399.

[25] S. P. Vanka, *Block-implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.

[26] A. Voronin, Y. He, S. MacLachlan, L. N. Olson, and R. Tuminaro, *Low-order preconditioning of the Stokes equations*, Numerical Linear Algebra with Applications, (2021), p. e2426.

[27] M. Wabro, *Coupled algebraic multigrid methods for the Oseen problem*, Comput. Vis. Sci., 7 (2004), pp. 141–151.

[28] ———, *AMGe—coarsening strategies and application to the Oseen equations*, SIAM J. Sci. Comput., 27 (2006), pp. 2077–2097.

II. Software & High Performance Computing

Articles in this section discuss the implementation of high performance computing (HPC) and productivity software. In many cases, performance improvements and portability are demonstrated for many-core architectures, such as conventional multicore CPUs, the Intel Many Integrated Core coprocessor (MIC), and graphical processing units (GPUs).

1. *Almgren-Bell, Moore* and *Gallis* implemented and tested the performance and accuracy of Bird's transient adaptive subcell collision method in SPARTA, an open-source parallel Direct Simulation Monte Carlo code.
2. *Gordon* and *Pedretti* ported the Kitten lightweight kernel to RISC-V to enable co-design for high-performance computing workloads.
3. *Grayson* and *Milewicz* investigate the use of theories of technology adoption such as the Perceived Characteristics of Innovation model to explain why scientific software developers adopt certain tools and practices and to help tool creators better understand their target audience.
4. *Li, Phipps* and *Kolla* develop a new algorithm for computing the Tucker decomposition of streaming tensors via an incremental SVD and blocked tensor-matrix multiplication.
5. *Logan* and *Lofstead* evaluate DOAS, a state-of-the-art distributed storage system optimized for high-performance storage and networking hardware.
6. *Parrish, Ciesko* and *Olivier* leverage *MPI Continuations* to integrate user-level threading with MPI in Sandia's QThreads runtime.
7. *Spigarelli* and *Davis* improve a satellite scheduling software user interface using a human predictive performance model called CogTool.
8. *Viens, Hart* and *Phillips* use mixed-integer programs to solve scheduling problems for Sandia testing facilities.

S.K. Seritan
J.D. Smith

November 11, 2022

COLLISION-PARTNER SELECTION SCHEMES IN DIRECT SIMULATION MONTE CARLO METHODS

JAMES ALMGREN-BELL [*], STAN MOORE [†], AND MICHAEL GALLIS [‡]

Abstract. Modern Direct Simulation Monte Carlo (DSMC) methods as detailed by Bird in 1994 are considered the primary method for the modeling of rarefied gas flows [1]. Since then, substantial research efforts have gone into proving the theoretical correctness and accuracy of DSMC as well as improving the numerical and computational methods involved. SPARTA is an open-source parallel DSMC code developed at Sandia with the goal of efficient large-scale simulation on cutting edge computer architectures. One of the most complex steps of DSMC simulation is the collide kernel, in which gas-phase collisions and chemistry occur within each cell. Collision partner selection methods can play a major role in improving problem runtimes. Here, we present an implementation of Bird's transient adaptive subcell (TASC) collision method in SPARTA along with discussions of accuracy as well as performance with MPI and KOKKOS.

1. Introduction. Although Monte Carlo algorithms have been around for over 50 years, modern Direct Simulation Monte Carlo (DSMC) methods for modeling rarefied gas flows were truly introduced by Graeme Bird in 1994 [1]. With modern computational capabilities enabling high precision DSMC simulations, substantial work has been done to improve the theoretical, numerical, and computational aspects of DSMC methods. Unlike molecular dynamics, DSMC methods track representative particles, or simulators, which are used to statistically represent groups of particles. In certain physical regimes, in particular where the Navier-Stokes equations are not applicable, DSMC can often offer more physical precision than continuum models by accurately capturing kinetic effects.

While DSMC methods can be incredibly powerful tools, they are not without their drawbacks. The particle representations used in DSMC require many more unknowns than standard continuum methods, and as a result high fidelity DSMC simulations are very computationally demanding. It is not uncommon to use billions of particles, each of which must be tracked and updated at each timestep. For the collision step of DSMC to be physically correct, locality of collisions between particles must be enforced. That is, a particle should only be able to collide with other particles that are sufficiently close. This maximal distance is the mean free path, the mathematical average distance a particle travels before experiencing a collision. Standard DSMC collision partner selection is done randomly from within the same grid cell. As a result, the cell size in each dimension must be smaller than the mean free path; in practice, cell size is often set to be a factor of 3 or 4 smaller than the mean free path. Problems with a large spatial domain can require hundreds of millions of grid cells. Within each of these cells, simulator density must be sufficiently high to accurately represent collision dynamics. Some problems may consider densities as low as approximately 20 simulators per cell; others may use over 1,000. As an additional numerical constraint, the timestep must be sufficiently small such that the probabilistic collision kernel does not misrepresent true collision statistics.

One way of reducing the cost of DSMC simulations is to increase the grid cell size; of course, the collision partner distance limit must still be enforced. The transient adaptive subcell (TASC) method introduced by Bird accomplishes both of these tasks through the computation of an additional grid structure within each cell at each timestep [3]. Collision partners are then chosen according to their distance within this subcell structure. By doing so, the TASC method allows for larger cell sizes, leading to an overall need for fewer simulators. In this paper, we describe our implementation of the TASC method into SPARTA

[*]University of Texas at Austin, jalmgrenbell@utexas.edu
[†]Sandia National Laboratories, stamoor@sandia.gov
[‡]Sandia National Laboratories, magalli@sandia.gov

and discuss its performance and accuracy relative to other collision partner schemes.

2. SPARTA. SPARTA, short for Stochastic PArallel Rarefied-gas Time-accurate Analyzer, is a parallel DSMC code developed at Sandia designed for performing simulations of low-density gases [6]. A timestep for a parallel DSMC simulation with SPARTA can be summarized by the following steps

1. Create: create particles at simulation-box or surface boundaries.
2. Move: advect each particle, surface collisions and chemistry may occur.
3. Communicate: migrate particles to new owning processors.
4. Sort: group particles by grid cell.
5. Collide: perform gas-phase collisions and chemistry within each grid cell.
6. Stats: tally values on a particle, grid-cell, or surface-element basis.

In this paper, discussion is limited to Step 5, the collision kernel, which is often the second most computationally expensive step after the Move step. For high precision large scale simulations with a high number of simulated particles, the collision step can dominate runtime. For this reason, it is essential to place focus on improvements to the methods and implementations within the collision kernel. A detailed discussion of this kernel is presented below.

3. Collisions in DSMC. DSMC methods model collisions between simulated particles within each grid cell. The number of possible collisions at each timestep is a function of all possible particle pairs, giving it $O(N^2)$ computational complexity (for N the number of particles in the cell). The industry standard collision scheme is the **no-time-counter method**, which computes the number of collision attempts N_c at each timestep as

$$N_c = \frac{1}{2}\frac{N(N-1)F_N(\sigma_T c_r)_{\max}\Delta t}{V_c} \tag{3.1}$$

where N is the number of simulators in the cell, F_N is the ratio of real molecules to simulators, $(\sigma_T c_r)_{\max}$ is the maximum of the product of the total collision cross section and particle velocity, Δt is the timestep size, and V_c is the volume of the cell. We note that, for computational efficiency, the value of $(\sigma_T c_r)_{\max}$ is rarely computed at each timestep; it is often approximated or infrequently recomputed. For each of the N_c collision pairs, two particles i, j are chosen and a collision probability P_{ij} is computed as

$$P_{ij} = \frac{(\sigma_T c_{r(i,j)})}{(\sigma_T c_r)_{\max}} \tag{3.2}$$

where σ_T is the cross section for the collision, $c_{r(i,j)}$ is the magnitude of the relative velocity of the two particles, and $(\sigma_T c_r)_{\max}$ is as before. The collision is then either accepted (and performed) or rejected depending on the value of a random number drawn compared to P_{ij}. But how are the two particles i, j in each of the N_c pairs chosen?

4. Partner Selection Methods. Typically, each of these N_c collision pairs are chosen by selecting two random particles i, j from within the cell. As previously mentioned, one way to increase the accuracy of a DSMC simulation is by selecting a random particle i but using careful selection of the partner particle j (as opposed to true randomness). One method of selecting the partner particle is using the **nearest neighbor method**. For a collision with primary particle i, a search is done over the cell to select the nearest neighbor to particle i. While this does perform collisions with the closest pairs of particles, this search can be very computationally expensive, as each collision partner search must consider all particles within the cell An adaptation of this method is to truncate the search after a certain number of potential partners and choose an approximate nearest neighbor from among those particles. What is gained in runtime, however, is lost in accuracy. A notable downside to the nearest neighbor method is that bias can be introduced, as the loss of randomness in the partner selection can hurt the accuracy of the simulation. Because DSMC separates the move kernel from the collision kernel, the nearest neighbor method can only select the nearest neighbor partner at the end of the particle's spatial move. Particles that could be collided with during the particle's trajectory during that step are missed. This bias is particularly noticeable for high velocity particles.

The **transient adaptive subcell (TASC) method** for partner selection offers an improvement to the nearest neighbor method [3]. At each timestep, each cell is refined with an additional grid structure, such that we expect to have approximately one simulator per grid subcell. The number of subcells in each direction is given by $n^{\frac{1}{d}}$, where n is the number of simulators in the cell and d is the dimensionality of the problem. For a primary particle i, the collision partner j is selected at random from the set of particles in subcells at the closest possible radial distance to the subcell of particle i. This distance refers to distance between subcells on the grid and is thus an integer; exact distance between particles is never considered. A noteworthy difference between the method implemented and the method originally described by Bird. Bird's method selects a random subcell from this minimal distance and then selects a random particle from within that random subcell. Our method counts all particles at this distance and selects randomly from them without the selection of a random subcell as an intermediate step. In this way, we avoid introducing any spatial bias which would arise if neighboring subcells had highly variable particle densities. The TASC method can be seen as an approximation to the nearest neighbor approach, but in a more efficient manner. There is an overhead to setting up the subcell structure, but repeatedly finding partners is much more efficient as this step can reuse information, unlike the nearest neighbor setup. Further, the randomness as a part of the TASC method in which a particle is chosen randomly from a given grid distance helps to reduce the spatial bias built in to the nearest neighbor method, which can only select the exact nearest neighbor rather than other close particles.

5. Performance Analysis. The computational cost of the TASC method can be broken down into two major steps. The first step is the setup of the subcell structure, which includes establishing the necessary data structures and mapping each particle to its corresponding subcell. The second step is the partner selection, in which the subcell grid structure must be traversed and a collision partner selected. The computational cost of the first step has exact $O(n)$ scaling, as all simulators must be mapped to their corresponding subcell. The second step, however, scales solely with the number of collisions performed, which itself is dependent on both the number of simulators in the problem as well as the simulation physics. While we do not present a detailed breakdown of the two steps here, it is worth noting that physical parameters of a given problem will likely have an effect on relative performance of the TASC method compared to the other collision partner methods. For example, problems that require high simulator density but low collision frequency will perform particularly poorly with the TASC method due to overheads. We now present timing data for the TASC method compared to the nearest neighbor methods (full and truncated) as well as the original partner selection method for both a small test problem and a large application problem.

5.1. Hardware. All simulations presented here were done on Sandia SRN HPC production clusters. The test problem, both in serial and using KOKKOS, was run on Attaway, featuring compute nodes with 36 cores each with 2.3 GHz Intel Xeon Gold 6140 processors.

These simulations were done on a single node. For the vortex shedding problem the compute cluster Manzano was used (for easier access to more nodes, as each simulation was run using 32 nodes). Compute nodes on Manzano feature 48 cores each with 2.9 GHz Intel Cascade Lake 8268 v4 processors.

6. Test Problem. We consider the test problem of a closed 3D cube with side length $L = 0.1$mm decomposed into 1,000 grid cells and filled with Argon gas of number density approximately $7 \times 10^{22} \frac{1}{m^3}$ at temperature $T = 273$K. The particles move, collide with each other, and reflect off the boundaries. 1,000 timesteps are taken of size $\Delta t = 7 \times 10^{-9}$s.

We compare the base DSMC collision scheme with fully random partner selection to the TASC method and the full nearest neighbors method. As previously discussed, we also consider the truncated nearest neighbor method, in which the nearest neighbor search is truncated after a certain number of partner trials. This is labeled below as NN-30 (limit of 30 particles searched per collision). For the test problem, we assume all methods can be run using the same timestep and the same cell size. While assumption is useful here for timing purposes, the TASC and nearest neighbor methods typically use larger cells and smaller timesteps in practice. This is further discussed in Section 7 when we examine a practical problem.

Timings presented measure the full collision kernel time during the simulation as a function of the average cell density. Timings were averaged over two runs for confirmation, although variance between runs was shown to be negligible. The results are shown in Table 6.1. The results confirm that the TASC method substantially outperforms the nearest neighbors method, even by almost a factor of two for high densities. Surprisingly, the TASC method outperforms the truncated NN-30 method even at high particle densities.

Simulator Density	Base	TASC	NN-30	NN
10	0.30	0.5	0.4	0.4
20	1.1	1.6	1.6	1.6
40	4.3	5.5	6.5	6.6
80	17.1	20.2	25.4	28.9
160	67.7	79.8	102.3	125.9
320	277.4	324.1	414.2	569.7
640	1083.6	1315.7	1696.9	2523.8

TABLE 6.1

Runtimes (seconds) of collision kernel for 1,000 timesteps in 2-D collisions test problem. Scaling study done as a function of simulator density (particles per cell).

7. Application Problem. The primary application problem we consider models 2-d gas flow around a circular cylinder experiencing vortex shedding, in which an inflow boundary condition on the left side of the domain generates low-pressure zones on the downwind (right) side of the cylinder. This phenomenon results in a fluctuating force in the transverse direction to the flow, which is discussed later. The vortex shedding behavior can be seen in the transverse flow velocity fields shown in Figure 7.1. This problem is drawn from prior work by Gallis and Torczynski, where extensive discussion of the physics involved can be found [4] [7]. In this paper, this problem is used for exploring both performance and accuracy for the TASC method.

We model an Argon gas flow with Mach number 0.3 and Reynolds number of 100 (and thus Knudsen number of Kn = 0.0048). A full parameter list is provided below:

- Problem domain: 15m (streamwise) by 10m (transverse)

- Cylinder diameter: $D = 1\text{m}$
- Molecular mass: $m = 66.3 \times 10^{-27}\text{kg}$
- Specific heat ratio: $\gamma = \frac{5}{3}$
- Viscosity: $\mu = 2.117 \times 10^{-5}\text{Pa s}$
- Temperature: $T = 273.15\text{K}$
- Freestream velocity $U = 92.37\frac{\text{m}}{\text{s}}$
- Density $\rho = 2.292 \times 10^{-5}\frac{\text{kg}}{\text{m}^3}$
- Number density $n = 3.457 \times 10^{20}\frac{1}{\text{m}^3}$
- Mean free path $\lambda = 0.0048\text{m}$

Collisions between the molecules are performed using the hard-sphere collision model. Molecule-surface reflections are performed using a probabilistic combination of diffuse and specular reflection governed by an accommodation coefficient α. Diffuse reflection off of the cylinder occurs with probability α and specular reflection occurs with probability $1-\alpha$. In this work, we limit discussion to the $\alpha = 0$ case, which corresponds to the fully specular limit.

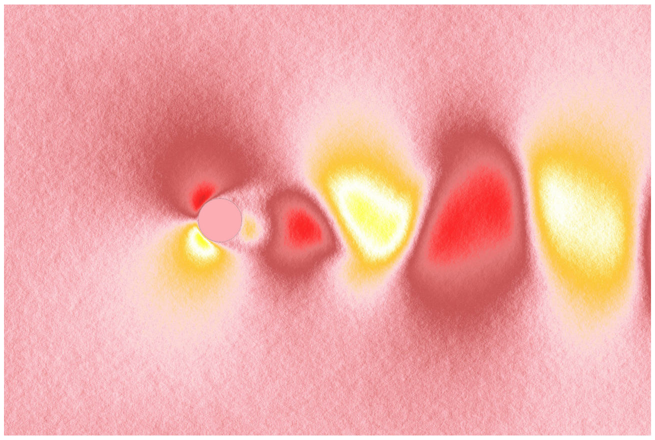


FIG. 7.1. *Transverse flow velocity field at $t = 1.31s$: evidence of vortex shedding*

7.1. Accuracy. To verify the accuracy of the TASC method, the primary data output of interest is the transverse component of the force, a product of vortex shedding. When the problem is initialized with the inflow conditions, the transverse force is almost zero; over time, this force grows and demonstrates oscillatory behavior. While the exact times in the simulation do not necessarily overlap between different runs, we can use a slight temporal shift in order to compare the oscillatory frequency and amplitude of the transverse force between runs. Reference exact solution data is provided by Gallis from a high precision simulation done with the base collision method [4]. We run simulations using the nearest neighbor and the TASC methods for comparison to the exact solution. The results are presented in Figure 7.2. We observe strong agreement of the subcell and nearest neighbor methods with the exact solution, a confirmation that the TASC method is working as intended. The increased noise compared to the exact solution is a result of reduced precision due to lower particle density compared to the exact solution.

7.2. Performance. We now turn our discussion to the performance of the different collision methods with this problem. As previously discussed, the base method has a more restrictive cell size due to the potential distance between collision partners. On the other hand, the nearest neighbor and TASC methods require a more restrictive timestep due to the locality of collision partner selection in large cells. That is, partners are chosen based

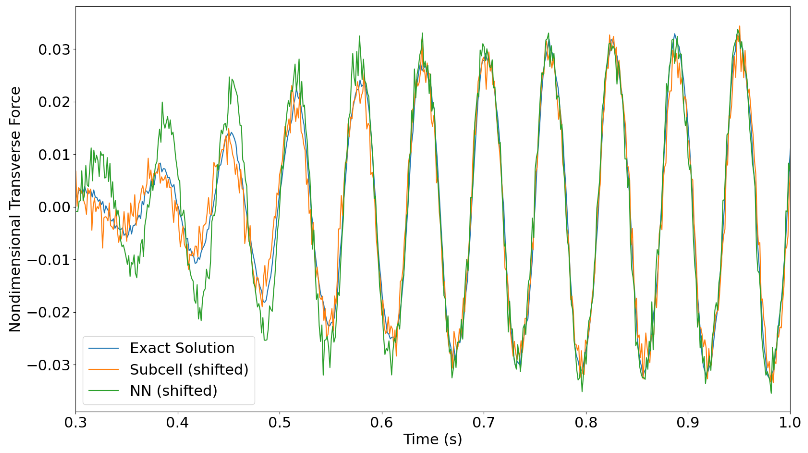


Fig. 7.2. *Transverse force measurements over time*

off of particle location after the advection step. If the timestep is too large, this introduces a substantial bias to partner selection. For this reason, base method is run with 4 cells per mean free path (in each direction) and a timestep of approximately 1.3×10^{-5}s. The subcell and nearest neighbor methods are run with 1 cell per mean free path (in each direction) and a timestep of approximately 1.3×10^{-6}s; they run with 16× fewer cells but 10× more timesteps. The timing results shown in Table 7.1 were done with 32 nodes and to a time horizon of approximately 2.6×10^{-2}s (2,000 and 20,000 timesteps, respectively).

Simulator Density	Base	Subcell	NN
100	2,285	1,138	985
1,000	40,992	19,788	16,867

Table 7.1
Collision kernel runtimes in vortex shedding problem

Unlike the test problem, this vortex shedding problem shows better performance for the nearest neighbors method compared to the TASC method. We note that this behavior is unexpected given the timing results from the test problem, in particular for the high particle density case. However, there are a number of differences that may explain the discrepancy. In particular, we have here a much larger problem size run across multiple nodes. Here, the domain is split into hundreds of millions of grid cells with tens of billions of particles. Unlike the collisions test problem which featured a uniform box, our domain here has flow around the circle, leading to uneven particle densities and flow conditions. Ongoing research is being done to resolve this runtime discrepancy to fully understand the effect of problem details on collision kernel run times, as expectations are that the TASC method should be outperforming the nearest neighbors method.

8. Other SPARTA Development.

8.1. Kokkos Implementation. Kokkos is a C++ performance portability programming model that allows the user to write one code base that is mapped onto different back-end languages for varied hardware. This includes languages such as CUDA or HIP for GPUs as well as OpenMP that can be used on CPUs [8]. Almost all of SPARTA's features are implemented in KOKKOS, an the TASC method implemented in the standard version of SPARTA was also implemented into the KOKKOS version.

Performance validation for the Kokkos implementation of the TASC method can be done in the same way as the serial version. We consider again the collision test problem as detailed in Section 6, with all parameters the same. Here, the problem size is fixed at an average cell density of 200 particle. Performance is measured as a function of the number of OpenMP threads used by Kokkos to confirm that the TASC method scales beyond the serial case. The timings are shown in Table 8.1 and demonstrate similar behavior as the serial case. The TASC method outperforms both the full and 30-limit near neighbor methods as well as shows acceptable performance in comparison to the base method.

Threads	Base	Subcell	NN-30	NN
1	43.6	78.5	78.4	123.6
2	16.9	28.9	33.8	57.2
4	8.4	14.3	17.1	28.7
8	4.4	7.5	9.1	15.3
16	2.6	4.3	5.2	8.7
32	1.4	2.2	2.7	4.3

Table 8.1
Collision kernel runtimes in test problem using KOKKOS

8.2. Majorant Collision Frequency Method. As previously discussed, the standard method for determining number of collisions within a timestep is the no-time-counter (NTC) method. One alternative is the majorant collision frequency (MCF) scheme [5]. With this scheme, first computed is the majorant frequency $v_{\max}$. Collision particle pairs are then drawn using any partner scheme (base, subcell, nearest neighbor). For each collision pair, a local timestep δt is randomly sampled from an exponential distribution for this pair. That is, for R a uniform random number between 0 and 1,

$$\delta t = -\frac{\log R}{v_{\max}} \tag{8.1}$$

Collisions are performed and values of δt are repeatedly drawn until $\sum \delta t > \Delta t$, the overall timestep. In this way, the MCF method has similarities to the time-counter method, which assigns timesteps to collision pairs. The MCF method also shows similarities to the NTC method, but with some improvement. The NTC method reproduces the average collision time, but only for sufficiently large sample sizes. MCF reproduces the exact Poisson distribution of collision time and also reproduces the correct average collision time, however it can do so more effectively with smaller sample sizes.

We implemented the MCF method into SPARTA with capabilities of using the base, nearest neighbor, and subcell collision methods. Preliminary testing has been done to confirm expected runtimes and accuracy for the collisional test problem previously detailed. Future work will look to pair the majorant collision frequency method with the TASC method for large scale problems such as the presented vortex shedding flow problem. The goal will be to explore if the combination of the MCF and the TASC methods provides additional solution accuracy for a fixed problem size or allows for a larger timestep to be taken.

9. Conclusion. In this paper, we have presented contributions and testing using SPARTA centered around the transient adaptive subcell method. While this method shows great potential in test problems, the computational performance proved to be subpar in the application problem. Theoretical estimates suggest that the TASC method should experience better scaling than the nearest neighbor method for simulations with a higher number of simulators [2]. Further, the TASC method holds more true to the desired randomness of partner selection in DSMC than does the nearest neighbor selection method. As a result, ongoing work continues to be done with the expectation of improving the subcell implementation to resolve the performance issues experienced in the full-scale simulation.

REFERENCES

[1] G. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, no. v. 1 in Molecular Gas Dynamics and the Direct Simulation of Gas Flows, Clarendon Press, 1994.

[2] M. Gallis and J. Torczynski, *Effect of collision-partner selection schemes on the accuracy and efficiency of the direct simulation monte carlo method*, International Journal for Numerical Methods in Fluids, 67 (2011), pp. 1057 – 1072.

[3] M. Gallis, J. Torczynski, D. Rader, and G. Bird, *Convergence behavior of a new dsmc algorithm*, Journal of Computational Physics, 228 (2009), pp. 4532–4548.

[4] M. A. Gallis and J. R. Torczynski, *Effect of slip on vortex shedding from a circular cylinder in a gas flow*, Phys. Rev. Fluids, 6 (2021), p. 063402.

[5] A. Pikus, I. B. Sebastião, S. Jaiswal, M. Gallis, and A. A. Alexeenko, *Dsmc-sparta implementation of majorant collision frequency scheme*, AIP Conference Proceedings, 2132 (2019), p. 070026.

[6] S. J. Plimpton, S. G. Moore, A. Borner, A. K. Stagg, T. P. Koehler, J. R. Torczynski, and M. A. Gallis, *Direct simulation monte carlo on petaflop supercomputers and beyond*, Physics of Fluids, 31 (2019), p. 086101.

[7] D. J. Rader, M. A. Gallis, J. R. Torczynski, and W. Wagner, *Direct simulation monte carlo convergence behavior of the hard-sphere-gas thermal conductivity for fourier heat flow*, Physics of Fluids, 18 (2006), p. 077102.

[8] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, *Kokkos 3: Programming model extensions for the exascale era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2022), pp. 805–817.

PORTING THE KITTEN LIGHTWEIGHT KERNEL OPERATING SYSTEM TO RISC-V

NICHOLAS GORDON* AND KEVIN PEDRETTI†

Abstract. Hardware design in high-performance computing (HPC) is often highly experimental. Exploring new designs is difficult and time-consuming, requiring lengthy vendor cooperation. RISC-V is an open-source processor ISA that improves the accessibility of chip design, including the ability to do hardware/software co-design using open-source hardware and tools. Co-design allows design decisions to easily flow across the hardware/software boundary and influence future design ideas. However, new hardware designs require corresponding software to drive and test them. Conventional operating systems like Linux are massively complex and modification is time-prohibitive. In this paper, we describe our port of the Kitten lightweight kernel operating system to RISC-V in order to provide an alternative to Linux for conducting co-design research. Kitten's small code base and simple resource management policies are well matched for quickly exploring new hardware ideas that may require radical operating system modifications and restructuring. Our evaluation shows that Kitten on RISC-V is functional and provides similar performance to Linux for single-core benchmarks. This provides a solid foundation for using Kitten in future co-design research involving RISC-V.

1. Introduction. Today, hardware/software co-design is typically performed by profiling and analyzing applications running on current computing systems, conducting software-based simulation studies using high-level architectural models and simplified application proxies or motifs, and by collaborating with vendors to influence their product roadmaps. Within the HPC community, this model has worked well and it will continue to do so. However, this approach can be slow, taking years to impact the hardware that can be purchased and deployed in supercomputers. Additionally, complete operating systems are usually not considered in co-design research, in part because software-based simulation tools are too slow or do not contain significant fidelity to support them.

The emerging RISC-V ecosystem promises to change this situation by greatly simplifying the task of hardware development by applying modern software development practices and agile methods [23]. Academia and industry have rallied around RISC-V [6] and generated a rich set of open-source tools and open-source hardware components [4,9,10,17,27]. It is now possible for individual researchers and small teams to rapidly construct prototype system-on-chip (SoC) designs using open-source RISC-V processor core and accelerator designs. These designs can then be instantiated as FPGA prototypes, which may run at 100's of MHz, as cycle-accurate FPGA-accelerated simulations [19], which may run at a reasonable fraction of real-time (e.g., simulate a 3.2 GHz SoC design at 65 MHz, or 1/50th real-time), or as physical hardware taped out using open-source tools [2,3,16,29].

These capabilities are revolutionary for co-design. The speed of FPGA-based instantiations is fast enough to boot full operating systems and evaluate real workloads running for trillions of instructions, far exceeding the capabilities of software-based simulation tools. It is now possible to instantiate a custom RISC-V SoC design on a cloud-based FPGA in a matter of minutes, boot Linux on it, and then interactively login and run common HPC benchmarks such as STREAM, HPCG, HPL, and GUPS. Based on the observed results, both hardware and software can then be quickly modified and re-tested, providing rapid iteration.

A gap in the RISC-V co-design ecosystem, however, is that while there are many RISC-V core designs available to experiment with, Linux is the primary OS used for evaluation purposes. This situation is understandable, as Linux is a general-purpose OS that is widely

*University of Pittsburgh, nick.gordon@cs.pitt.edu

†Sandia National Laboratories, ktpedre@sandia.gov

used. However, it is also unwieldy, consisting of millions of lines of code, and requires considerable expertise to modify. It also takes a long time to build.

In this paper, we address this gap by porting the Kitten [21] lightweight-kernel operating system to RISC-V to provide another option for performing co-design research with RISC-V. Kitten specifically targets the needs of HPC workloads and is the most recent in a series of lightweight kernels that have been successfully deployed on production supercomputers [20, 26, 31]. The small code base of Kitten (approx. 20K lines of code in the Kitten core) together with its simple resource management policies provides benefits compared to Linux for conducting hardware/software co-design that requires implementing new OS functionality.

We envision leveraging Kitten on RISC-V to explore novel memory subsystem and tightly-coupled network hardware capabilities that will require modifications to both hardware and operating system software. In some cases, once proven out, these capabilities can be implemented in Linux with further effort. In other cases, the assumptions that Linux makes about hardware may make it difficult or impossible to implement support, in which case Kitten could be used to support the new hardware capabilities by running in a virtual machine hosted by Linux.

In the remainder of this paper, we describe the port of Kitten to RISC-V in more detail and demonstrate that it can provide performance comparable to Linux on emulated and real RISC-V hardware. This work provides a solid foundation for using Kitten in co-design research involving RISC-V. Kitten is available as open-source software and runs on x86_64, Arm, and now RISC-V architectures [1].

2. The Design Barrier and Co-Design.

2.1. The Design Barrier. What we call the *design barrier* is a semantic gap between hardware and software that allows development of one to proceed without consideration for the other. However, this gap also means that *when* a change must be made in one for the sake of the other it is difficult.

Conventional computer design is generally done hardware-first, regardless of the specifications of the system. It is expected that software will catch up to accommodate whatever hardware is developed. From experimental, cutting-edge supercomputers to ultra-low power embedded systems, hardware dictates the design of systems. This paradigm is unsurprising, but the intense focus on hardware often results in neglecting the corresponding software. For example, Intel's Itanium very-large instruction word (VLIW) architecture provided an interesting and powerful processor, but without a corresponding compiler to generate the plan of execution the hardware was difficult to use and struggled with poor adoption.

Additionally, the continued development of the Linux kernel and its enduring and increasing ubiquity has pushed the system to become more general and less hardware-specific. Indeed, Linux *must not* become hardware-specific. Naturally, this limits the degree to which software can be tailored to hardware.

This scheme of development, where hardware and software are developed independently, has permitted kernels such as Linux to become enormously featureful and sophisticated and for hardware to become highly optimized. However, work must be continuously done to knit the edges of each domain back together.

Similarly, the process abstraction has been standard in operating systems for many decades, but only recently has research investigated hardware acceleration. The XPC project by Du shows that substantial gains can be made by extending the underlying hardware to accelerate IPC, which has been a stable abstraction for quite some time [15].

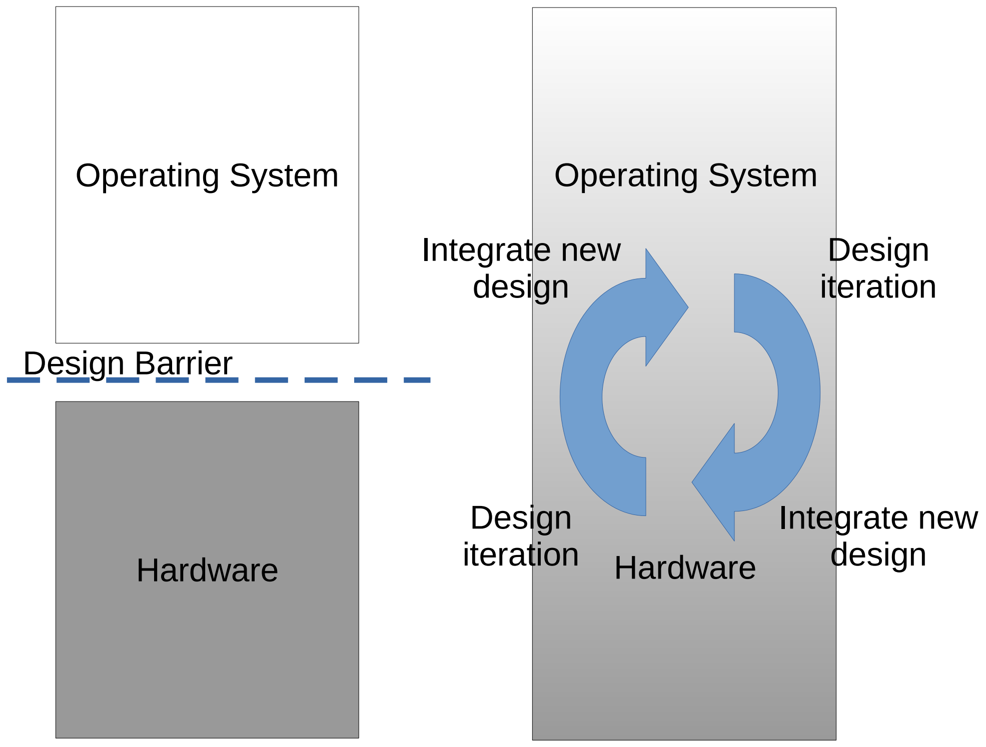


Fig. 2.1: Co-design allows design decisions to flow across the "design barrier."

2.2. Co-design. Co-design is a design strategy that intentionally lowers the design barrier, reducing the semantic gap, and allowing for software and hardware to develop in consideration of each other. Previously, the long design times for hardware and software required design isolation and independence. Developing hardware required sending designs to chip fabs and waiting for production, and then evaluating test units. The introduction of high-productivity open-source hardware design tools and field-programmable gate arrays (FPGAs) simulator platforms have dramatically reduced the time and cost needed to prototype new hardware designs. This approach, coupled with on-demand cloud resources to accelerate build and testing times, has shortened the duration of a design iteration. Design cycle times and labor requirements can now be on the order of weeks and tens of engineers, rather than years and hundreds of engineers.

This new design timeline means that experimental designs both in hardware and software are less risky and the size of the explorable design space for a given timeline has increased substantially. Intentionally changing both hardware and software at the same time and by the same team can yield substantial benefits and should be seriously considered, especially in a co-design research context.

3. LWKs and RISC-V. Two factors come together to encourage research using co-design: lightweight kernels and open hardware. Lightweight kernels (LWKs) are specialized, simplified kernels used in high-performance computing (HPC), intended to replace general-purpose kernels like Linux [28]. These kernels are explicitly designed with performance in mind, eschewing wide-scale hardware support or an abundance of features. These circumstances are suited to HPC due to the unique combination of hardware and workloads enabling these trade-offs.

The Kitten LWK is designed to support HPC workloads running on massively parallel supercomputers. It has a Linux-compatible ABI and provides a low-noise, non-preemptive environment for executing tightly-coupled HPC workloads running across tens of thousands of compute nodes. Kitten makes use of simplified resource management policies, such as providing physically contiguous memory allocation and performing round-robin task scheduling.

Kitten's simplified nature makes development considerably easier, ranging from design advantages such as a less-abstracted hardware and resource management layer to practical considerations such as build time. The iterative nature of co-design accentuates the importance of these traits; quick kernel development enables quick evaluation and a greater number of total iterations available to explore a design. Additionally, a simpler kernel design and small code base allows us to easily change policies or entire kernel mechanisms as experimental hardware requires.

An important distinction is that Kitten is not just an experimental kernel. Research kernels are abundant and developed very quickly, but can become abandonware and their results difficult to generalize or reproduce. Although research kernels are extremely useful, extending a production-targeted LWK OS such as Kitten is an important component of evaluating designs in realistic circumstances.

RISC-V is an open-standard hardware ISA initially developed by the University of California, Berkeley, with an active community of researchers and contributors. Although only first released in 2010, the ISA has gained considerable traction as a new avenue for hardware design due to its open nature. The RISC-V ISA supports multiple word-lengths and a number of instruction set extensions including atomics, quad-precision floating point, and compressed instructions. The ISA is modular and extensible, allowing a compliant chip to support only the instruction set extensions needed for a particular use case.

Many toolchains have been developed to both target the architecture as well as develop it. Hardware simulators such as Verilator [30] and the FPGA-accelerated FireSim [19] are popular for developing and evaluating designs. Many projects exist to provide development frameworks and implementation languages. Cornell has developed the PyMTL [24] project, and Berkeley has developed the Chipyard [4] project which specifically targets RISC-V. Additionally, many hardware description languages (HDLs) can target these RISC-V platforms, including Chisel [8], Clash [7], and PyMTL itself. There are a large number of academic RISC-V designs ranging from conventional superscalar out-of-order CPUs like the BOOM [33] project to many-core designs like Manticore [32].

The foregoing discussion highlights that the combination of an easily-extended LWK with the infrastructure of the growing RISC-V ecosystem can provide a good platform to break the design barrier and facilitate hardware and software co-design.

4. Design, Implementation, and Evaluation. In this paper, we describe initial work to explore the co-design space using LWKs and RISC-V by porting the Kitten LWK to a set RISC-V platforms. As part of this effort we have initially targeted the common software emulator QEMU for ease of development, and have also implemented support for the HiFive Unmatched development board by SiFive [11]. We have furthermore conducted an early evaluation to demonstrate functional correctness on both the QEMU and HiFive platforms, as well as performance comparisons between Linux and Kitten.

4.1. Porting Kitten. Our port of Kitten currently supports a single CPU core and is able to run basic HPC benchmarks. All of the major subsystems have been implemented, including context switching, timers, multi-process scheduling, system calls, signals, and interrupts. Symmetric multi-processor (SMP) support is not yet functional, but it is under active development and should be straightforward to implement using RISC-V's Supervisor Binary Interface (SBI) abstractions.

Porting an operating system kernel primarily involves adapting the assembly code that interfaces with hardware directly, however higher-level components of the kernel are often completely untouched, such as the scheduler and memory allocation policies. To reduce maintenance burden and maintain broad Linux compatibility, we borrow code from Linux wherever possible and adapt it for Kitten.

To establish and contain a development environment we use Podman [13], Red Hat's container software that is compatible with Docker. Now widely used, containers and VMs alike constrain developer system variability, allowing many developers to easily share a single development environment. Inside of our Podman container we use Buildroot [12], a project for bootstrapping an entire system from compiler and toolchain to a working Linux kernel image with a root filesystem. Buildroot can target several architectures including RISC-V and supports RISC-V's many toolchain binary configurations. For testing and development we use version 6.2.0 of QEMU targeting 64-bit RISC-V configured with 1 GB of memory and a single CPU, also contained in the Podman container. This containerized environment allows development, building, and testing to occur quickly and in a reproducible way that is distributable to others.

Porting Kitten highlights engineering trade-offs associated with porting software to new hardware. While a full discussion of this engineering is a better fit for a technical report, we share a few examples. Compared to x86_64, RISC-V has an abundance of registers for storing kernel state. However, RISC-V has fewer such registers than ARM64. To access per-CPU data on ARM64, the correct pointer is stored in the `tpidr_el1` register, and the current running task's information is kept on the stack. However, the ARM64 architecture can use this approach because the kernel retains the kernel stack pointer in the form of `sp_el1`. The RISC-V ABI lacks an analogous register and must retrieve the kernel stack pointer on entry. For the RISC-V port, we overcome this obstacle by following Linux's example and storing in the `tp` register a direct pointer to the *task_struct*. Then the stack pointer can be retrieved. This lack of bankable system registers forces us to rely on having a pointer to the entire task state instead.

4.2. Performance Evaluation. We evaluated performance on several platforms, including QEMU, the Unmatched development board, and partially on the FireSim simulator, not included here. We use HPCG [14], the high-performance conjugate gradient descent benchmark, as our evaluation workload for this work-in-progress.

The Unmatched board has a supervisor core and four general-purpose application cores. Linux and Kitten run on the general-purpose cores and do not make use of the supervisor core. The application cores are RISCV64IMAFDC, often shortened to RISCV64GC, where G is for "general", including integer, multiply, atomic, and single- and double-precision floating point. The C extension is for compressed instructions, which can be leveraged to produce smaller binaries. The cores also support the RISC-V architecture's three-privilege model, firmware runs in machine (M) mode, the operating system in supervisor (S) mode, and user programs in user (U) mode. The board has two levels of cache, a core-local L1 and a shared 2 MB L2 cache, and 16 GB of RAM. RISC-V supports many different memory address widths, including 39-, 48-, and 57-bit standards and a specification for full 64-bit address widths is under development. The specification requires that unused bits in each address are non-canonical and must be set the same as the last canonical bit. The Unmatched board is equipped with the SV39 MMU which implements 39-bit addresses. The computer we used for development and emulator benchmarks is a Dell Precision Power 3420 with an 3.2 GHz i5-6500 CPU with 16 GB of system memory. HPCG was built with `"-O2"` optimization flags and statically-linked. HPCG is a common performance benchmark used to evaluate HPC systems, providing a good balance of compute- and memory-intensiveness. We set the problem size input parameter to `"80 80 80"` for all tests. HPCG is useful as a first-order approximation to see performance characteristics of different platforms. To be a valid run of the benchmark, the problem size must be configured to exceed the machine's cache size. Accordingly, the benchmark especially highlights memory bandwidth constraints.

We ran HPCG under both QEMU and the HiFive Unmatched board, with the results shown in Figure 4.1. The results show, interestingly, that the QEMU tests show nearly a 2x speedup over the Unmatched development board. We speculate that this difference is attributable to the Unmatched board's lack of an L2 cache prefetcher, strongly limiting overall memory performance. Our experience has been that building software under QEMU is generally faster than building software natively on the Unmatched board. For comparison purposes, we also show the performance of HPCG running under FireSim for a single Rocket core configuration in Figure 4.1. We plan to have Kitten running under FireSim in the near future.

On both platforms, Kitten provides a speedup over Linux. This speedup is due to Kitten's policy of allocating user memory contiguously using large pages, defaulting to a 2 MB page size. These changes improve TLB performance and can improve cache prefetch performance. Particularly on the Unmatched board, raw floating point performance does not show a difference between Linux and Kitten; the performance arises from the differences in the kernel memory systems.

To determine memory performance both on QEMU and the Unmatched board, we used two memory benchmarks found in the HPC challenge benchmark suite [25]: STREAM and RandomAccess. We tested memory bandwidth using the STREAM benchmark. STREAM evaluates memory performance across four streaming kernels to characterize the memory subsystem performance of a system. The results in Figure 4.2 show a significant performance advantage for QEMU in this benchmark. As with HPCG, we suspect that the lack of L2 prefetcher severely hinders performance.

We ran RandomAccess with a 128 MB table size. Results in Figure 4.3 continue to show a slight advantage of Kitten over Linux, but interestingly the Unmatched board outperforms QEMU in this benchmark. We suspect the high TLB costs associated with QEMU's emulated memory subsystem may be limiting its performance, allowing the otherwise-slower Unmatched board's memory subsystem to outperform here, but additional investigation is required to confirm.

4.3. Potential Co-Design Targets. In this section we outline several potential avenues of exploration that could be pursued using Kitten on RISC-V.

Global Address Spaces. A large portion of the 64-bit address width on a RISCV64 system is unused and thus extra information could be put there instead. This technique is already employed in software in the form of tagged pointers, often used in program verification; some architectures like x86_64 ignore some bits of a given memory address. An extension to the MMU to use tag bits in a memory access to route requests to remote hosts addressed by those tag bits is a natural fit for Partitioned Global Address Space (PGAS) workloads. Remote memory accesses using RDMA are common and extending this paradigm upward into general userspace code allows for remote host memory to be added to the structural "memory hierarchy" that already contains cache and local system

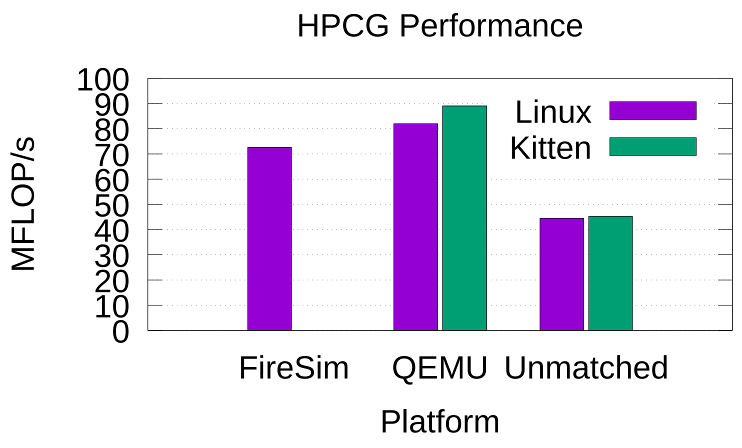


Fig. 4.1: HPCG performance across several systems. Error bars are omitted as result variance was 2% or less in all tests.

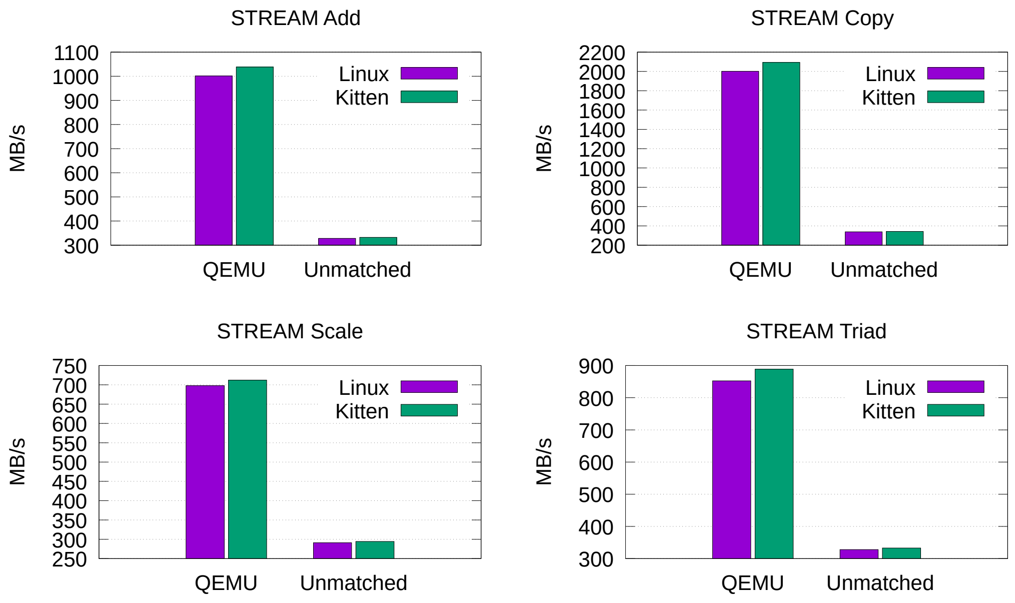


Fig. 4.2: STREAM Benchmark Results

memory. Further, previous research such as Infiniswap has already demonstrated the utility of transparent remote memory accesses [18].

Unmatched Supervisor Core. As projects like Manticore suggest, development in the RISC-V space will not be just on traditional computer organization models centered around a small number of CPUs with a strong interconnect. Not just extremely parallel many-core architectures, but also increasingly system-on-chip (Soc) architectures resemble distributed

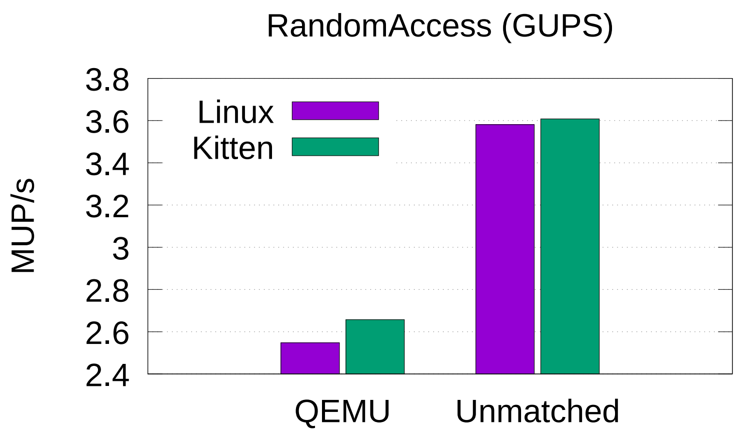


Fig. 4.3: RandomAccess Benchmark Results

systems with significant functionality, such as power and peripheral management, being exposed to the "real" OS as a defined interface via memory-mapped IO. However convenient, this points to a deeper issue that the design of conventional kernels such as Linux are not well-suited for these SoCs.

The Unmatched board is equipped with *five* cores, four of which are application cores. However, the supervisor core is not used by Linux. This supervisor core could be used analogously to a kernel thread and act as a service core; it lacks floating point operations and an MMU, both things operating systems typically only need to provide to userspace. A shim kernel could be split off of Kitten and run underneath each of the cores to provide system services via message passing between the application cores and service core.

Unmatched Scratchpad and L2 Memory. The Unmatched board has a 2 MB L2 cache organized into 16 ways, 15 of which are non-reserved. However, this memory is reconfigurable, referred to as "loosely-integrated memory" by SiFive. When ways are disabled, they are accessible as directly mapped memory instead, providing low-capacity, high-speed access for guaranteed "cache hits." This memory is also accessible as scratchpad. The memory subsystem of Linux has no convenient way to expose this memory, but the simpler memory model of Kitten allows us to expose this memory to applications, e.g. by placing application heap there or exposing mmap flags to prefer it.

Though the board only has approximately 2 MB of this memory, this reconfigurability is not tolerated well by Linux, but is straightforward to accommodate with Kitten. This advantage highlights the agility Kitten has in development compared to Linux. Additionally, this reconfigurable memory could be used as a message-passing buffer, greatly accelerating message-passing between cores.

5. Related Work. Considerable architecture research exists exploring RISC-V. However, much of this research focuses on hardware design and defaults to using Linux as the driving operating system.

Prior work has ported Kitten to other processor architectures and supercomputer platforms. Lange ported Kitten to ARM64 [22] to explore using Kitten as a management kernel to drive Hafnium, a secure hypervisor. This work supports the notion that a simpler kernel like Kitten can provide performance improvements over Linux in specialized situations.

Work exists porting kernels to RISC-V for the purpose of exploring or driving primarily software-targeted design. As mentioned before, XPC [15] investigates accelerating IPC by integrating it into hardware, using RISC-V as a target platform. The CHERI project introduces capabilities into the hardware itself. Initially released as an ARM64 board, the Morello platform, recently RISC-V has been targeted as well [5]. Hardware capabilities move a substantial portion of the operating system's role, managing userspace access control, into hardware by way of capabilities, which are opaque hardware tokens that the operating system grants to userspace.

6. Future Work. As mentioned in the evaluation section, this paper presents preliminary work in porting Kitten to RISC-V and is not yet feature complete. While the current state of the port is sufficient to test functionality and obtain preliminary benchmark results, use on emerging RISC-V hardware will certainly require SMP support. We will add SMP support to Kitten while exploring some of the ideas mentioned in Section 4.3.

In its current state, Kitten will likely boot on any "typical" RISC-V chip that supports the RISC-V three-privilege model. However, we discovered during porting to the Unmatched board an issue similar to that encountered in the Arm ecosystem. Namely, that hardware topology is exposed through a device tree which must be parsed and used for configuration. Unfortunately, the structure of these device trees is not fully consistent, being notably different between QEMU and the Unmatched board. We anticipate that although most chip implementations will feature similar if not identical hardware, it will be necessary to provide platform drivers. Hardware design work will necessarily begin in a simulator like Verilator or FireSim, and we plan to provide drivers for these platforms in the future.

7. Conclusion. We have described our port of the Kitten lightweight kernel operating system to the RISC-V architecture. Our evaluation demonstrated the port running on two different platforms, QEMU and the SiFive Unmatched development board, with performance comparable to and generally slightly better than Linux. This project lays the groundwork for using Kitten in future co-design research involving RISC-V, with potential speed and agility advantages compared to Linux. We outline several potential directions for co-design research in the OS kernel and hardware domains that we plan to pursue in the future leveraging Kitten on RISC-V.

REFERENCES

[1] *Kitten Lightweight Kernel Operating System.* `https://github.com/HobbesOSR/Kitten`.

[2] *SiliconCompiler.* `https://www.siliconcompiler.com`.

[3] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem, G. Pradipta, S. Reda, M. Saligane, S. S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, L. Wang, Z. Wang, M. Woo, and B. Xu, *Toward an open-source digital flow: First learnings from the OpenROAD project*, in Proceedings of the 56th Annual Design Automation Conference 2019, DAC '19, New York, NY, USA, 2019, Association for Computing Machinery.

[4] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, *Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs*, IEEE Micro, 40 (2020), pp. 10–21.

[5] A. Armstrong, T. Bauereiss, B. Campbell, A. D. Reid, K. E. Gray, R. M. Norton, P. Mundkur, M. Wassell, J. French, C. Pulte, S. Flur, I. D. B. Stark, N. R. Krishnaswami, and P. Sewell, *Isa semantics for armv8-a, risc-v, and cheri-mips*, Proceedings of the ACM on Programming Languages, 3 (2019), pp. 1 – 31.

[6] K. Asanović and D. A. Patterson, *Instruction Sets Should Be Free: The Case For RISC-V*, Tech. Rep. UCB/EECS-2014-146, EECS Department, University of California, Berkeley, Aug 2014.

[7] C. Baaij, M. Kooijman, J. Kuper, W. Boeijink, and M. Gerards, *Clash: Structural descriptions of synchronous hardware using haskell*, in Proceedings of the 13th EUROMICRO Conference

on Digital System Design: Architectures, Methods and Tools, United States, Sept. 2010, IEEE Computer Society, pp. 714–721. eemcs-eprint-18376 ; 13th EUROMICRO Conference on Digital System Design, DSD 2010 : Architectures, Methods and Tools, DSD ; Conference date: 01-09-2010 Through 03-09-2010.

[8] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, *Chisel: Constructing Hardware in a Scala Embedded Language*, in Proceedings of the 49th Annual Design Automation Conference, DAC '12, New York, NY, USA, 2012, Association for Computing Machinery, p. 1216–1225.

[9] J. Balkind, T. Chang, P. J. Jackson, G. Tziantzioulis, A. Li, F. Gao, A. Lavrov, G. Chirkov, J. Tu, M. Shahrad, and D. Wentzlaff, *OpenPiton at 5: A Nexus for Open and Agile Hardware Design*, IEEE Micro, 40 (2020), pp. 22–31.

[10] J. Balkind, K. Lim, M. Schaffner, F. Gao, G. Chirkov, A. Li, A. Lavrov, T. M. Nguyen, Y. Fu, F. Zaruba, K. Gulati, L. Benini, and D. Wentzlaff, *BYOC: A "Bring Your Own Core" Framework for Heterogeneous-ISA Research*, in Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '20, New York, NY, USA, 2020, Association for Computing Machinery, p. 699–714.

[11] F. Bellard, *Qemu, a fast and portable dynamic translator*, in USENIX Annual Technical Conference, FREENIX Track, 2005.

[12] Buildroot Association, *Buildroot - Making Embedded Linux Easy*. `https://buildroot.org/`.

[13] Containers Organization, *Podman*. `https://podman.io/`.

[14] J. Dongarra, M. A. Heroux, and P. Luszczek, *HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems*, Tech. Rep. ut-eecs-15-736, Nov. 2015.

[15] D. Du, Z. Hua, Y. Xia, B. Zang, and H. Chen, *Xpc: Architectural support for secure and efficient cross process call*, 2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA), (2019), pp. 671–684.

[16] Efabless, *Efabless Open Multi Project Waffer Shuttle Program*.

[17] F. Fatollahi-Fard, D. Donofrio, J. Shalf, J. Leidel, X. Wang, and Y. Chen, *OpenSoC System Architect: An Open Toolkit for Building Soft-Cores on FPGAs (FPL)*, in 27th International Conference on Field Programmable Logic and Applications, 2017, pp. 1–1.

[18] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, *Efficient memory disaggregation with infiniswap*, in NSDI, 2017.

[19] S. Karandikar, H. Mao, D. Kim, D. Biancolin, A. Amid, D. Lee, N. Pemberton, E. Amaro, C. Schmidt, A. Chopra, Q. Huang, K. Kovacs, B. Nikolic, R. Katz, J. Bachrach, and K. Asanović, *FireSim: FPGA-accelerated Cycle-exact Scale-out System Simulation in the Public Cloud*, in Proceedings of the 45th Annual International Symposium on Computer Architecture, ISCA '18, Piscataway, NJ, USA, 2018, IEEE Press, pp. 29–42.

[20] S. M. Kelly and R. Brightwell, *Software Architecture of the Light Weight Kernel, Catamount*, in Proceedings of the 2005 Cray User Group Annual Technical Conference, 2005.

[21] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, et al., *Palacios and Kitten: New High Performance Operating Systems for Scalable Virtualized and Native Supercomputing*, in IEEE International Symposium on Parallel & Distributed Processing (IPDPS), 2010.

[22] J. R. Lange, N. Gordon, and B. L. Gaines, *Low overhead security isolation using lightweight kernels and tees*, 2021 SC Workshops Supplementary Proceedings (SCWS), (2021), pp. 42–49.

[23] Y. Lee, A. Waterman, H. Cook, B. Zimmer, B. Keller, A. Puggelli, J. Kwak, R. Jevtic, S. Bailey, M. Blagojevic, P.-F. Chiu, R. Avizienis, B. Richards, J. Bachrach, D. Patterson, E. Alon, B. Nikolic, and K. Asanovic, *An Agile Approach to Building RISC-V Microprocessors*, IEEE Micro, 36 (2016), pp. 8–20.

[24] D. Lockhart, G. Zibrat, and C. Batten, *Pymtl: A unified framework for vertically integrated computer architecture research*, 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, (2014), pp. 280–292.

[25] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. D. McCalpin, D. H. Bailey, and D. Takahashi, *Introduction to the hpc challenge benchmark suite*, 2005.

[26] A. B. Maccabe, K. S. McCurley, R. Riesen, and S. R. Wheat, *SUNMOS for the Intel Paragon: A brief user's guide*, in Proceedings of the Intel Supercomputer Users' Group. 1994 Annual North America Users' Conference, June 1994.

[27] P. Mantovani, D. Giri, G. Di Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, and L. P. Carloni, *Agile SoC Development with Open ESP*, in Proceedings of the 39th International Conference on Computer-Aided Design, ICCAD '20, New York, NY, USA, 2020, Association for Computing Machinery.

[28] R. Riesen, A. B. Maccabe, B. Gerofi, D. N. Lombard, J. Lange, K. T. Pedretti, K. B. Ferreira, M. Lang, P. Keppel, R. W. Wisniewski, R. Brightwell, T. Inglett, Y. Park, and

Y. Ishikawa, *What is a lightweight kernel?*, Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers, (2015).

[29] SkyWater and Google, *SKY130 Process Design Kit*, 2020.

[30] W. Snyder, *Verilator*. `https://veripool.org/wiki/verilator`, 2022.

[31] S. R. Wheat, A. B. Maccabe, R. Riesen, D. W. van Dresser, and T. M. Stallcup, *PUMA: An operating system for massively parallel systems*, Scientific Programming, 3 (1994), pp. 275–288.

[32] F. Zaruba, F. Schuiki, and L. Benini, *Manticore: A 4096-core risc-v chiplet architecture for ultra-efficient floating-point computing*, IEEE Micro, 41 (2021), pp. 36–42.

[33] J. Zhao, *Sonicboom: The 3rd generation berkeley out-of-order machine*, 2020.

USING PERCEIVED CHARACTERISTICS OF INNOVATION TO DESIGN SOFTWARE TOOLS FOR SCIENTISTS

SAMUEL GRAYSON* AND REED MILEWICZ†

Abstract. Many scientists need to develop or modify software for their research. However, these scientists often lack the software engineering skills necessary to write maintainable software. Software tools, such as workflow systems, can simplify this process, making their software more maintainable and improving their productivity simultaneously, but those tools see sparse adoption in the scientific community. Sociological theories driving technology adoption, specifically the Perceived Characteristics of Innovation (PCI), can help tool-makers design software that becomes widely used by the scientific community. To support this claim, we analyze interviews from potential users about two software tools from the lens of PCI. Finally, we use PCI to identify recommended practices for tool-makers to consider.

1. Introduction. Computing has become increasingly important to modern science. Two-thirds of researchers say their research would not be possible without it, and more than half develop their own as part of their research [13]. Now, more than ever, software quality is a major factor of research quality.

One promising approach to maintaining software quality with low effort is to "automate" low-level aspects of software quality into tools that automatically enforce or inform some aspect of software quality. Not only this, using the tool can indirectly teach the user better practices in the process of improving one's code. This work studies how to encourage researchers to adopt such tools.

Lenberg et al. argue from their systematic literature review that studying behavioral elements of software engineering is the low-hanging fruit [14]. Hannay et al. argue likewise from their systematic literature review that the field of software engineering could benefit from incorporating empirically-based theories [11]. Therefore, this work looks towards sociological theories of technology acceptance to explain how potential users choose to adopt software tools, hoping to extract insights towards influencing that process. The Perceived Characteristics of Innovation (PCI), in particular, is specifically relevant to the problem of software quality tools. The features, characteristics, marketing, and documentation of software tools influence the constructs of PCI, so PCI can be used to analyze design decisions.

In this work, we examine prior work in technology adoption models (Section 2), then we find prior interviews relating to tools for software engineering quality in research (Section 3), re-contextualize the results through the lense of PCI (Section 4), and compile recommendations for tool-developers targeting a research audience (Section 5).

2. Technology adoption models. "Adoptability" is an opaque term; therefore, one should look to prior literature on *why* computer systems are adopted to elucidate this. Sociology and information science literature define several *technology adoption models* which seek to answer that question. These models build a qualitative theory that can quantitatively predict which tools will be adopted based on a questionnaire submitted by users.

Davis proposed Technology Acceptance Model (TAM) based on the Theory of Reasoned Action (TRA). TRA posits subjective norms and attitudes influence behavioral intent, which influences behavior [8]. TAM adds perceived ease of use and perceived usefulness as factors influencing attitudes [6]. Numerous extensions to TAM have been proposed which either add extra influences to attitudes, add moderating variables, or add other influences to behavioral intent [16].

*University of Illinois at Urbana-Champaign, grayson5@illinois.edu
†Sandia National Laboratories, rmilewi@sandia.gov

Model	Year	Basis
TAM [6]	1986	TRA [8]
TAM ext. [16]	1989 – 2013	TAM
PCI [18]	1991	DI [23]
UTAUT [25]	2003	TAM, PCI, and others

Table 2.1
Origin of quantitative technology-adoption models

Model	Direct Variables[a]	Moderating Variables	Vars	Items
TAM	ease of use, usefulness	3[b]	12[c]	
TAM ext.	experience, self-efficiency	gender, cultural background	many	many
PCI	relative advantage, ease of use, compatibility, trialability, result demonstrability, visibility, image, voluntariness	0	9	27[d]
UTAUT	effort expectancy, performance expectancy, social influence, experience, facilitating conditions	gender, age, voluntariness, experience	42[e]	

[a]All of these constructs should be understood as their "perceived" version, e.g. "perceived ease of use."
[b]See Figure 4.1 in Davis [6]. Note that more variables are listed, but only this many are needed to predict user behavior from the constructs.
[c]TAM studies use many different sets. The most common in Table 7 by Davis [7].
[d]There were originally 43 items proposed by Moore and Benbasat [18], but Gounaris and Koritos [9] successfully use 27 items.
[e]See Table 17 by Venkatesh [25].

Table 2.2
Details of well-known quantitative models of technology-adoption

Diffusion of Innovations (DI) is a theory that explains how new technologies and ideas spread throughout the community developed by Rogers [23]. It identifies relative advantage, complexity, compatibility, observability, and trialability. Rogers intends these as subjective "perceived" qualities, not absolute qualities.[1] Moore and Benbasat adapt these for the specific problem of technology adoption by comparing these to TAM, testing them empirically, and adding their own [18]; the result is called the Perceived Characteristics of Innovation (PCI). Relative advantage is similar to TAM's perceived usefulness according to Moore, which is retained in PCI. Complexity is similar but inverted to TAM's ease of use, and they recommend TAM's terminology in PCI, making PCI a superset of TAM. Compatibility and trialability are retained in PCI. However, Moore and Benbasat found that observability statistically factors into two characteristics they call visibility and result demonstrability. Further, Moore and Benbasat isolate 'image,' which Rogers treated as a part of relative ad-

[1]Note that compatibility refers to the innovation being perceived as compatible with the existing values, needs, and past experiences, not a strictly technological sense of compatibility. See Section 4 for exact definitions.

vantage and add a novel construct corresponding to voluntariness. See Section 4 for exact definitions of PCI characteristics.

Venkatesh et al. proposed a popular extension of TAM called the Unified Theory of Acceptance and Usage of Technology (UTAUT) [25]. In addition to weighting each principal factor, UTAUT weights the interaction of principal factors with one or two moderating factors. For example, there is a coefficient for the product of effort expectancy, gender, and age.

- Compared to TAM, PCI has a slightly better fit [20, 9] and its constructs are a superset of TAM, which means it give more direction for tool-makers. On the other hand, TAM is slightly simpler and takes fewer items to assess. Note that relative advantage in PCI is similar to perceived usefulness in TAM, so PCI has a superset of the variables in TAM.
- Compared to extensions of TAM, PCI is more well-studied and most extensions will have more variables to fit. The inputs to extensions of TAM are often direct effects of properties of the user, such as experience and self-efficiency, which are less useful than PCI inputs which are properties of the tool.
- Compared to UTAUT, PCI is far less complex; UTAUT has five constructs and four moderating variables. Counting two-way and three-way interactions that Venkatesh et al. find significant yields 42 different variables to fit. PCI has just eight constructs and eight variables to fit. Even with this complexity, UTAUT does not always explain as well as TAM [21].

Therefore, PCI is most applicable for this work.

3. Methodology. In this work, we examine whether PCI as an innovation adoption model can be useful in the context of studying tool and practice adoption in scientific software development contexts. To that end, we provide a qualitative analysis of adoption drivers in computational science and engineering projects. Our analysis of PCI is based on interviews conducted to gather data relating to two workflow-related projects:

1. The **Engineering Common Model Framework** (ECMF) project at Sandia National Laboratories is an effort to develop a common platform for storing and sustaining science and engineering workflows developed at the labs. While workflows are tremendously useful in expressing computational experiments, they need to be stored in an accessible way and regularly tested to guard against regressions, and the data that they generate must also be archived along with provenance tracking the history of that data as the underlying computational models evolve. The vision for ECMF is one in which cyberinfrastructures support the adoption and use of workflow technologies, and that those workflows are sustained as an enduring institutional resource, promoting sharing, collaboration, and reusability of computational experiments across the enterprise.
   Interview methodology: One of the authors conducted semi-structured interviews with individual or paired participants according to predefined questions but with freedom to explore points that came up. The subjects were selected from potential users of the ECMF family of workflow-related tools. The interview questions were chosen to explore how the analyst community uses computational models to solve problems and how a system like ECMF could help make analysis tasks easier.
2. The **ExaScale Additive Manufacturing** (ExaAM) project, an application within the Exascale Computing Project, seeks to develop an exascale-capable multi-physics simulation environment for additive manufacturing [24]. ExaAM is a multi-lab effort to make high-fidelity microscale simulations of additive manufacturing available

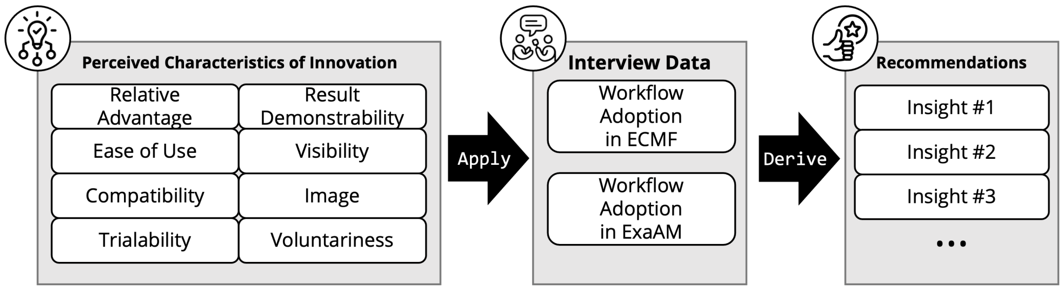


Fig. 3.1. *An overview of the methodology used in the development of this report. The authors selected a theoretical lens (PCI) to study the adoption of innovations in scientific software development, applied this lens to data drawn from science and engineering projects with respect to workflow adoption (ExaAM and ECMF), and used these results to derive actionable guidelines for tool developers seeking to support scientific software development.*

to industry[2]. The ExaAM project has been working towards adopting workflow technologies (specifically using the ExaWorks workflow management system [1]) to reduce their time to science and provide reproducible and reusable computational experiments for the benefit of the research community.

Interview methodology: The authors and two other researchers interviewed two ExaAM team members. The interview subjets were principal investigators of the ExaAm project, who are overseeing the adoption of workflows. The interview questions were selected to study the experiences and best practices of workflow adoption in practice.

In the case of ExaAM, the authors of this report recently interviewed the ExaAM team for an upcoming workshop paper where PCI was used as a theoretical lens to analyze the interview results [15]. We extend that work by reanalyzing previously unpublished ECMF interviews performed as part of requirements gathering with analysts to surface innovation adoption factors with regards to workflows and enabling technologies for workflows. We compare both results to explore the degree to which PCI captures relevant phenomena of interest in the adoption of workflow technologies. Finally, we use these results to provide recommendations to tool developers seeking to apply theoretical models like PCI to better understand how those factors can be measured and used to inform decision-making.

4. Results.

For each construct of PCI, we compiled interview responses that relate to that construct. Overall, the constructs, except for visibility and voluntariness, are well-supported from these interview responses.

4.1. Relative Advantage.

PCI defines relative advantage as "the degree to which using an innovation is perceived as being better than using its precursor" [18]. In research software, relative advantage is the underlying technical benefit of using a software tool. Often, relative advantage is the *primary* concern for tool-makers, since it is the most obvious. The ExaAM interviewees illustrate their rationale for adopting workflows by describing a relative advantage to prior methods, arguing that workflows can streamline the operation of simulation scripts in a way that makes both use and reuse easier:

Jacob[3] (ExaAM): It [before adopting workflows] would require the manual operation of scripts that do each piece of the analysis, with manual

[2]See `https://www.exascaleproject.org/research-project/exaam/`

[3]The names have been changed to protect the privacy of the sources.

manipulation between stages. . . It's less amenable to being used to do new things.

The notion of advantage being *relative* is key here. For Jacob, the overhead of manually executing scripts in succession is acceptable for performing a single task with those scripts. Where workflows shine, according to Jacob, is that they are easy to reconfigure to perform new and different tasks. Among our ECMF interviewees, we found similar sentiments with regards to workflows:

Oliver (ECMF): That's why I'm very excited about SAW NGW[4] because it has proven its mettle. It is possible to make thing modular and reusable. Like my big scripts, do you think I can understand them a year from now? Hell no. Clarity of communication and archival of thought is a big part of workflows.

Oliver shares Jacob's desire for reusability of simulation scripts and adds that the "self-documenting" nature of workflows reduces the cognitive overhead involved in reuse. That is, running everything manually implies that the user must retain knowledge of how to run the scripts or to document that information externally. *Takeaway: There are different ways in which workflow technologies confer relative advantage, and careful case-study analysis can draw out what is most salient to prospective users.*

4.2. Compatibility. PCI defines compatibility as "the degree to which an innovation is perceived as being consistent with the existing values, needs, and past experiences of potential adopters" [18]. In research software, one can consider compatibility in a technical sense, like how the Debian-Almquist shell is compatible with POSIX, or in a social sense, like how git is aligned with the values of decentralized community-building held by Linux kernel developers. The ethos of science holds reproducibility of experiments as a core tenant [17]; tools that align with this mission may see greater adoption. In the case of ExaAM, Jacob argues that workflows help ensure that the simulation capabilities developed for the project remain used and useful in the long term:

Interviewer: How compatible are these practices to project goals for ExaAM for successful implementation?
Jacob (ExaAM) : It's essential. The deliverable needs to be usable by other projects and other problems. . . We could satisfy ECP's goals without it, but it would not be a long-term success.

That is, for Jacob, workflows are desirable not only because they meet the immediate objectives of the project, but also because they serve the team's ethos to make their work widely accessible. Conversely, innovations that lack compatibility with users' existing ways of working and thinking may be seen as unattractive to users, even if they could be advantageous otherwise. As Simon, an analyst interviewed for ECMF, explains:

Simon (ECMF): Most analysts have a way that works for them. By switching over to this new method, they will take a performance hit. It will take them longer to answer questions. I could imagine that turning people away.

Takeaway: One must balance the relative advantage of a new tool with compatibility of the old way of doing things to see adoption.

4.3. Ease of Use. PCI defines ease of use as "the degree to which [using an innovation is perceived as being][5] free of physical or mental effort" [18]. Some software tools, like git,

[4]SAW NGW (Sandia Analysis Workbench Next Generation Workflows) is a workflow used in-house at Sandia National Labs.
[5]The wording within the brackets was changed for consistency's sake

are easy for software engineers to understand, but not necessarily for researchers who do not have a background in software engineering; in this sense, ease of use must be considered relative to an intended audience. For the ExaAM project, interviewees SAW workflows as being particularly valuable for encouraging use by industry partners with limited experience with command-line scripts, even though academic researchers may not care as much:

> **Cameron (ExaAM)**: They [industry partners] don't want to have to dig into the guts of all the components. Maybe they're not a sophisticated programmer, but these workflows allow them to "program" a problem. They benefit moreso than people in labs and universities who may be more adept in computational science.

Interviewees for ECMF, likewise, emphasized the need for workflow system developers to prioritize consistency and ease of use:

> **Ellen (ECMF)**: I think [ECMF is] a really great idea and I hope it starts to be picked up. The most important feature is that it should be stable from the start. Like NGW is experiencing this issue where you created a workflow in the beginning, and now it doesn't run in the newest version. To earn people's trust, these tools need to be trustworthy from the very beginning. You can't have everything changing every day.

Takeaway: Ease of use is not a static property of software tools but one that evolves dynamically over time; usability requires a commitment to keeping the software in line with user expectations.

4.4. Result Demonstrability. PCI defines result demonstrability as the degree to which using an innovation is amenable to demonstration[6] [18]. In research software, result demonstrability implies the user can explain in simple terms what the software helps them with. With regards to ExaAM and scientific workflows, workflows abstract away all the minutiae of running their codes, and this is readily apparent and explainable:

> **Interviewer**: One more question here about not having to dig into the codes, that workflows act as an interface?
>
> **Cameron (ExaAM)**: Exactly. Not having to know about all the internals and how to run the codes individually, that helps a lot.

Workflow abstractions provide tangible benefits, which makes it much easier for Cameron to sell using workflows to his peers. In an interview for ECMF, Oliver concurred on this point:

> **Interviewer**: If I understand you correctly, you're saying we need to show that the workflows are still working and that we can be confident that they won't fail when you run them at scale.
>
> **Oliver (ECMF)**: Yes, yes. This is where the NGW pipeline comes to your help. We break up the workflow into many, many small pieces. Usually, it's hierarchic. Three to four levels of hierarchy. The links in this hierarchy, these are all supremely unit-testable.

Takeaway: Being able to break down complex workflows via these abstractions into simpler, testable steps helps build confidence in the software.

4.5. Visibility. PCI defines visibility as the degree to which an innovation is perceived as being used by others[7] [18]. Researchers want to be unique in their approach, but not in their methodology. If a certain tool or practice is commonly used by others, that votes in favor of the tool working, providing advantage, and having a community where one can

[6]Moore and Benbasat do not give a verbatim definition.
[7]Moore and Benbasat do not give a verbatim definition.

seek help. Workflows have already have high-profile users such as Pegasus at LIGO [2] and vibrant communities like workflows.community[8]. However, we did not find direct references for visibility in the interviews.

4.6. Image. PCI defines image as "the degree to which an innovation is perceived to enhance one's image or status in a social system" [18]. Publishing well-written, usable code can increase the reputation of a researcher. If software tools can claim to improve the public perception of a researchers project, it can accelerate their adoption. Whether or not it makes the project reproducible, there is a perception that having a Dockerfile means the authors are sophisticated enough to think about reproducibility. The ExaAM interviewees did not explicitly mention image as a motivation for adopting workflows. However, workflows inherently aid in the perception that their project is well-written and reproducible and thus gives credence to the group that created it.

On the other hand, as one analyst interviewed for ECMF explains, having such a computational model may imply obligations to maintain that model, and teams may be hesitant to make that commitment:

> **Edvaldo (ECMF)**: The only person right now who is [nominally responsible for a computational simulation] is the lead analyst on the system itself. But the reality is that the model is owned by the system organization itself. There's a disconnect there, because they try to stay clear of getting too involved with the analysis itself, they're more concerned with the result. I don't think that they purposefully try to divorce themselves from this process, it's that there's always this thought that "if I commit to it, I have to provide some monetary support..." There is a cost because they must make someone accountable to respond to questions about that product/system.

Takeaway: Maintaining social "image" may come at a cost, and how teams perceive the image conferred by adopting an innovation can be nuanced and complex.

4.7. Trialability. PCI defines trialability as "the degree to which [one perceives that][9] an innovation may be experimented with before adoption" [18]. In research software, it should be possible to adopt a tool in a piecemeal. This is especially important for potential users creating an adoption schedule. For example, Mypy[10] is a static type-checker for Python that supports *progressive typing*: if the user writes type-annotations in their code, it can statically detect type errors. Mypy cannot detect errors on un-annotated code, but it at least does not crash on un-annotated code. A user can start using Mypy on their pre-existing unannotated code and progressively add annotations in future iterations.

In the case of ExaAM, the team learned how to use workflows over time, as Jacob explains:

> **Jacob (ExaAM)**: You learn by doing. It's awkward initially when you first adopt something, but overtime you become comfortable with it. We started with manual scripts, and built up towards workflows over time...In terms of ExaAM, the team has never had to create a "meta-program" like this, and we're learning as we go. How do we build these workflows? How do we make them all compatible and able to be managed by a workflow manager. It's a learning process, an educational process.

That is, if adopting workflows were an all-or-nothing proposition, the learning curve would be too steep for the team to consider using them. Rather, workflows are attractive

[8]See `https://workflows.community`

[9]The bracketed text has been added for consistency. These are intended to be perceived characteristics.

[10]See `http://www.mypy-lang.org/`

because the team can get useful results by assembling workflows for individual tasks and subcomponents, building upon each success, and gaining confidence in the technology. Bryan, an analyst interviewed for the ECMF project, shared these same sentiments:

> **Bryan (ECMF)**: I'm in a transition period right now where I try to convert everything to NGW. So I don't think the NGW models I've been building are ready for prime time, but I'm getting them there. I'm trying them out on dummy problems and gaining confidence.

Takeaway: The ability to acquire competency over time with workflows, applying them to progressively greater challenges, is attractive and a significant factor in his decision to adopt them.

4.8. Voluntariness. PCI defines voluntariness as "the degree to which use of the innovation is perceived as being voluntary or free of will" [18]. This is usually outside the control of tool-makers. However, there are some examples of voluntariness affecting adoption, such as when well-used journals like Nature require software to be available to the reader.[11] One can imagine journals requiring a sharing contract [5] or an OSI license. Voluntariness, however, did not come up in the ExaAM interviews or the ECMF interviews.

5. Recommendations. In all, through our exploratory analysis of ExaAM and ECMF as workflow adoption use cases, we found support for seven of the eight innovation factors described by the PCI model. We note that complete coverage is not expected; depending on the innovation and user community involved, some innovation characteristics may be especially important and others not at all. But this does show that formal theories of innovation adoption can be used to elucidate different drivers influencing the decision to use (or not use) a software development tool or practice.

Table 5.1
PCI Characteristics Attested In Interviews with ExaAM and ECMF Stakeholders.

Innovation Characteristic	Present in ExaAM interviews	Present in ECMF interviews
Relative Advantage	✓	✓
Ease of Use	✓	✓
Compatibility	✓	✓
Trialability	✓	✓
Result Demonstrability	✓	✓
Visibility		
Image		✓
Voluntariness		

While computational science and engineering benefits from modern software engineering technologies used in mainstream industry (*e.g.,* version control, continuous integration [12]), there is a need for innovative solutions that can meet the specific needs of scientific software development teams. Scientific workflows are a prime example of the research community seeking to develop "native" software tools to advance the state of practice. However, the development of these tools should be viewed as a research challenge. As noted by Bernholdt et al., "Although many scientists have extensive intuition about the principles and dynamics

[11] This was the policy at the time of writing `https://www.science.org/content/page/science-journals-editorial-policies#TOP-guidelines` on 23 August 2022.

of how their community develops, uses, and sustains its software products, research is needed to develop a deeper and broader understanding of software's role in scientific processes" [3]. As with the science these tools enable, systematic investigation is necessary to build out theoretical and empirical bases around the engineering of these tools.

In this report, we have shown how social science theories like PCI can help disentangle the complex needs of the workflow user community. However, PCI is a theory for analysis and understanding not one of design and action, in Gergor's taxonomy of information system theories [10]. PCI cannot prescribe solutions, but it can provide a basis for evidence-based decision-making in tool design and development. In this section, we outline strategies for how the results we present can be used productively:

- **Use of Theory in Conceptualization**: Developers of software tools for the scientific software community are frequently part of that same community. While having rich knowledge of the domain is useful in understanding potential user needs, tool developers can easily fall into the "tacit assumption tar pit" — assuming their attitudes towards their own tool reflect the attitudes of the broader community [4]. The use of well-defined theoretical models like PCI in the design and conceptualization of tools can help to draw out and critique those assumptions. With that in mind, we recommend that tool developers articulate how and whether each of the innovation characteristics described in the PCI model are significant to themselves and their potential users.
- **Use of Theory in User Engagement**: Theories such as PCI can guide tool-makers on how to engage with their user community. This is illustrated in our report: PCI gives us a lens through which we can more closely examine what workflow users need and value in workflow technologies. Interview questions should seek to elucidate where the tool falls on the eight PCI characteristics, and how each can be furthered.
- **Use of Theory in Tool Analysis**: Finally, PCI can help tool-makers quantitatively and qualitatively analyzing the capabilities of their tools, as a way of guiding their future evolution. For scientific workflow tools, we found compatibility with existing norms, values, and experiences was a significant factor in the decision to adopt. As such, it would be prudent to consider how scientific workflow tools can better serve those use cases such as reuse and shareability.

There are many varied reasons why users may want to adopt software tools, and different sets of priorities (*e.g.,* relative advantage, ease of use) may lead to different requirements for the design of the tool. This can go together with other ideation techniques meant to put the developer "in the shoes" of their potential users. For instance, Mundt et al. suggests conducting design thinking workshops and using persona-based techniques to better understand how scientific software developers relate to and understand their software [19, 22].

5.1. Threats to Validity. This study does not show the superiority of PCI *c*ompared to other methods, just that PCI does relate to scientific software development and is deserving of further research. We do not test whether PCI is exhaustive (there could be important aspects of adoptability that fall outside of PCI), just that it covers important aspect of the selected responses. This motivates its use in recommendations and in future work.

6. Conclusion. PCI can be a useful lens to help tool-makers create tools that are widely used by the community. Those tools could improve the quality of software engineering done by scientists, which would lead to better research.

6.1. Future Work. As this report is exploratory, follow-up work could evaluate PCI against other technology adoption models rigorously. One would want to know if PCI captures all relevant considerations for adoption, or just some. Similarly, one might study if any PCI attributes are less relevant in the specific case of scientific software.

One might re-assess the factor loadings of survey questions used to measure PCI constructs in the domain of scientific software. The original survey instrument uses terms such as "outside this organization" to define visibility that do not necessarily translate into research software. What should this term mean for a small academic lab and how is that different for a federally funded research and development lab? The factor loadings could be different for those working on research software.

Finally, one might pursue action research of applying the PCI-based analysis to up-and-coming software tools such as ECMF, ExaFlow, SAW, DataSEA, PSI/J, Dask, Pegasus, NextFlow, CWL, or others. One can be studying the general application of PCI while simultaneously developing a first-class workflow manager or other scientific software tool.

REFERENCES

[1] A. Al-Saadi, D. H. Ahn, Y. Babuji, K. Chard, J. Corbett, M. Hategan, S. Herbein, S. Jha, D. Laney, A. Merzky, T. Munson, M. Salim, M. Titov, M. Turilli, T. D. Uram, and J. M. Wozniak, *ExaWorks: Workflows for exascale*, in 2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS), 2021.

[2] R. M. Badia, E. Ayguade, and J. Labarta, *Workflows for science: a challenge when facing the convergence of HPC and Big Data*, Supercomputing Frontiers and Innovations, 4 (2017). interest: 95.

[3] D. E. Bernholdt, J. Cary, M. Heroux, and L. C. McInnes, *The science of scientific-software development and use*, tech. rep., USDOE Office of Science (SC)(United States), 1 2022.

[4] D. M. Berry, *The importance of ignorance in requirements engineering*, Journal of Systems and Software, (1995).

[5] C. Collberg and T. A. Proebsting, *Repeatability in computer systems research*, Communications of the ACM, 59 (2016), pp. 62–69.

[6] F. D. Davis, *A technology acceptance model for empirically testing new end-user information systems: theory and results*, thesis, Massachusetts Institute of Technology, 1985.

[7] ———, *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*, MIS Quarterly, 13 (1989), p. 319.

[8] M. Fishbein and I. Ajzen, *Belief, attitude, intention, and behavior: an introduction to theory and research*, Addison-Wesley series in social psychology, Addison-Wesley Pub. Co, Reading, Mass, 1975.

[9] S. Gounaris and C. Koritos, *Investigating the drivers of internet banking adoption decision: A comparison of three alternative frameworks*, International Journal of Bank Marketing, 26 (2008), pp. 282–304.

[10] S. Gregor, *The nature of theory in information systems*, MIS quarterly, (2006), pp. 611–642.

[11] J. E. Hannay, D. I. Sjoberg, and T. Dyba, *A Systematic Review of Theory Use in Software Engineering Experiments*, IEEE Transactions on Software Engineering, 33 (2007), pp. 87–107.

[12] D. Heaton, *Software engineering for enabling scientific software development*, PhD thesis, University of Alabama Libraries, 2015.

[13] S. Hettrick, *Softwaresaved/Software_in_research_survey_2014: Software In Research Survey*, Feb. 2018.

[14] P. Lenberg, R. Feldt, and L. G. Wallgren, *Behavioral software engineering: A definition and systematic literature review*, Journal of Systems and Software, 107 (2015), pp. 15–37.

[15] A. Malviya-Thakur, R. Milewicz, S. Grayson, P. Fackler, J. Belak, and J. A. Turner, *Real-world experiences adopting workflows at exascale on the exaam project*, 2nd Workshop on Reproducible Workflows, Data Management, and Security (ReWoRDS 2022), (2022).

[16] N. Marangunić and A. Granić, *Technology acceptance model: a literature review from 1986 to 2013*, Universal Access in the Information Society, 14 (2015), pp. 81–95.

[17] R. K. Merton, *The sociology of science: theoretical and empirical incvestigations*, Univ. of Chicago Pr, Chicago, 4. dr. ed., 1974.

[18] G. C. Moore and I. Benbasat, *Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation*, Information Systems Research, 2 (1991), pp. 192–222.

[19] M. Mundt, R. Milewicz, and E. Raybourn, *In their shoes: Persona-based approaches to software quality practice incentivization*, Computing in Science & Engineering, (2022), pp. 1–1.

[20] C. R. Plouffe, J. S. Hulland, and M. Vandenbosch, *Research Report: Richness Versus Parsimony in Modeling Technology Adoption Decisions—Understanding Merchant Adoption of a Smart Card-Based Payment System*, Information Systems Research, 12 (2001), pp. 208–222.

[21] M. M. Rahman, M. F. Lesch, W. J. Horrey, and L. Strawderman, *Assessing the utility of TAM, TPB, and UTAUT for advanced driver assistance systems*, Accident Analysis & Prevention, 108 (2017), pp. 361–373.

[22] E. Raybourn, R. Milewicz, and M. Mundt Brandenburg, *Incentivizing adoption of software quality practices.*, tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2022.

[23] E. M. Rogers, *Diffusion of innovations*, Free Press ; Collier Macmillan, New York : London, 3rd ed ed., 1983.

[24] J. A. Turner, J. Belak, N. Barton, M. Bement, N. Carlson, R. Carson, S. DeWitt, J.-L. Fattebert, N. Hodge, Z. Jibben, W. King, L. Levine, C. Newman, A. Plotkowski, B. Radhakrishnan, S. T. Reeve, M. Rolchigo, A. Sabau, S. Slattery, and B. Stump, *Exaam: Metal additive manufacturing simulation at the fidelity of the microstructure*, The International Journal of High Performance Computing Applications, 36 (2022), pp. 13–39.

[25] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, *User Acceptance of Information Technology: Toward a Unified View*, MIS Quarterly, 27 (2003), p. 425.

EFFICIENT FACTOR UPDATE FOR STREAMING TUCKER DECOMPOSITION WITH INCREMENTAL SVD

ZITONG LI[*], ERIC PHIPPS[†], AND HEMANTH KOLLA[‡]

Abstract. The Tucker decomposition of a static tensor has been studied extensively. However, there are fewer studies on computing the Tucker decomposition of streaming tensors. In this paper we propose a new algorithm for this problem. More specifically, we assume that the input tensor has fixed sizes for all but its last mode. The last mode is assumed to be the time dimension and grows indefinitely. At any time step, we expect to obtain the Tucker decomposition of the data accumulated thus far. The main challenge of this problem is to limit the memory usage and computation complexity as the tensor grows. In our proposed algorithm, we take advantage of the Tucker decomposition of the previous time steps to update the factor matrices when a new slice of the tensor becomes available. In addition, we use blocked matrix tensor multiplication to reduce the cost of updating the core. As a result, our algorithm is much more efficient for computing the Tucker decomposition of a streaming tensor than the algorithms proposed for static tensor.

1. Introduction.

There is a wealth of literature on computing the Tucker decomposition of a static tensor [5][1]. However, in many cases, the dataset is not static and is constantly changing or growing in size. This streaming scenario poses challenges to algorithms designed for static tensors because the cost of computation and storage increases rapidly as the tensor becomes larger. An overview of the recent developments in tensor decomposition can be seen in this survey paper [7]. In this work, we consider the problem of efficiently computing the Tucker decomposition of a d-way tensor that has a fixed size for its first $d-1$ modes and is growing in its last mode. This corresponds to real-world problems where the tensors are growing as time goes on. A good example would be a simulation of a physics phenomenon over many time steps.

To make our problem more specific. We assume that our d-way data set $\mathcal{X}$ is of size $\{I_1, I_2, \dots, I_d\}$ where I_d increases by 1 at each time step. In other words, a new slice of data will appear at each time step. Now at any time step n, we denote the previously available tensor as $\mathcal{X}$ and the new slice as $\mathcal{Y}$, and the new tensor is formed by concatenating $\mathcal{X}$ and $\mathcal{Y}$ along mode d (see the left half of Fig. 1.1 for an example). We also assume that the Tucker decomposition of $\mathcal{X}$ has already been obtained.

2. Background.

2.1. Notations and definitions.

In this work, we treat tensor as simply as a multi-dimensional array. We use a capital letter in script font (e.g. $\mathcal{X}$) to denote a tensor. To specify a subset of the tensor we use the Matlab notation. For example, the first element of a 3-way (or 3D) tensor $\mathcal{X}$ is denoted as $\mathcal{X}(1,1,1)$; the first mode-1 fiber of $\mathcal{X}$ is denoted as $\mathcal{X}(:,1,1)$; the first frontal slice of $\mathcal{X}$ is denoted as $\mathcal{X}(:,:,1)$. The letters $I_1, \dots, I_d$ are reserved to denote the size of a tensor. Similarly, the letters $R_1, \dots, R_d$ is reserved to denote the *n-rank* of a tensor. It is sometimes useful to convert a tensor into a matrix. A typical way to do this is to unfold the tensor along a certain mode. We denote the mode-n unfolding of a tensor as $\mathbf{X}_{(n)}$. The tensor times matrix (TTM) operations is another important operation. We define and denote the multiplication of tensor $\mathcal{X}$ and matrix $\mathbf{U}$ along the n-th mode of $\mathcal{X}$ as the following:

$$\mathcal{X} \times_n \mathbf{U} = \mathbf{U}\mathbf{X}_{(n)} \tag{2.1}$$

For more detailed definitions, notations, and basic tensor operations we refer readers to [6].

[*]University of California Irvine, zitongl5@uci.edu
[†]Sandia National Laboratories, etphipp@sandia.gov
[‡]Sandia National Laboratories, hnkolla@sandia.gov

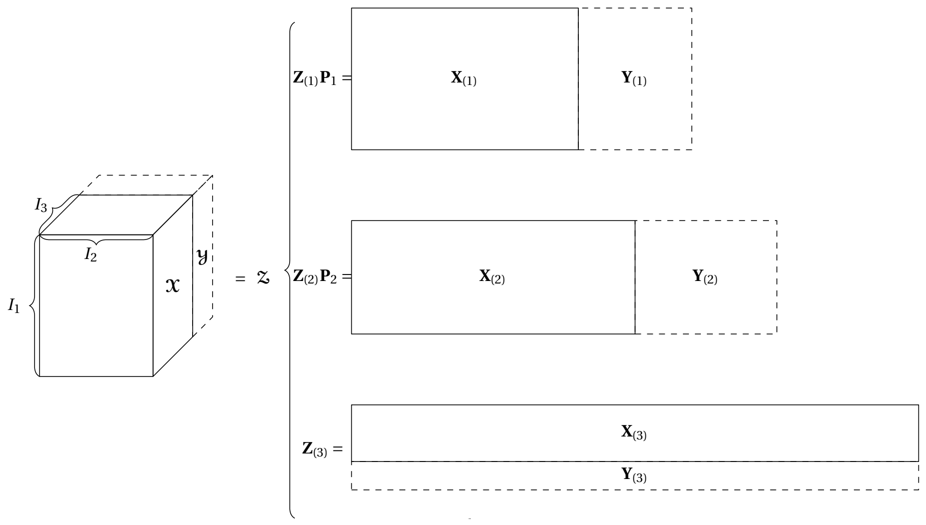


FIG. 1.1. *Tensor unfolding of the new tensor on each mode.* $\mathbf{P}_1$ *and* $\mathbf{P}_2$ *are permutation matrices*

2.2. Tucker tensor decomposition. The Tucker decomposition of a d-way tensor is a low-rank representation of a large tensor as a smaller tensor multiplied with d factor matrices. An example is shown in Fig. 2.1. More often instead of an exact low-rank representation,

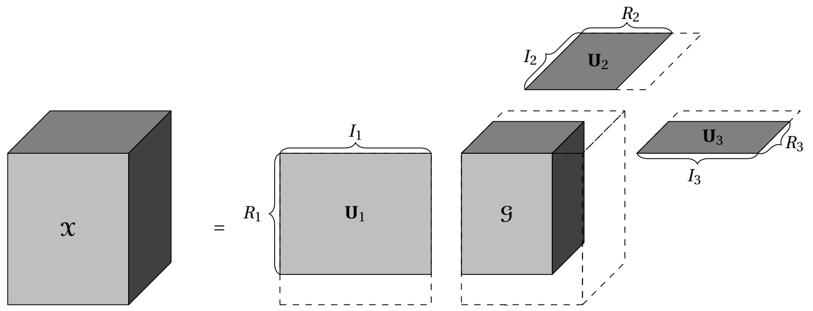


FIG. 2.1. *Tucker decomposition of a 3-way tensor* $\mathcal{X}$ *with size* $I_1 \times I_2 \times I_3$ *and n-rank* $R_1 \times R_2 \times R_3$

we want an approximation. One way to compute such a low-rank approximation is known as the higher order singular value decomposition (HOSVD) [8]. The pseudocode of which is shown in Algorithm 1. This algorithm takes as an input the error tolerance ϵ and produces a $\mathcal{G}$ such that

$$\frac{\|\mathcal{X} - (\mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times \cdots \times_d \mathbf{U}_d)\|_F}{\|\mathcal{X}\|_F} < \epsilon\sqrt{d}. \tag{2.2}$$

In this paper we will refer to the size of this core tensor $\mathcal{G}$ as the n-rank of $\mathcal{X}$.

2.3. Incremental SVD . Incremental SVD is a technique that allows us to reduce cost in our streaming Tucker algorithm. Given the SVD of a rank-r matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$:

$$\mathbf{A} = \mathbf{MSW}^\mathsf{T},$$

and that $\mathbf{Z} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\end{bmatrix}$ where $\mathbf{b} \in \mathbb{R}^m$, this technique computes the SVD of $\mathbf{Z}$ by leveraging the SVD of $\mathbf{A}$. (we deviate from the normal $\mathbf{U\Sigma V}^\mathsf{T}$ notation to avoid confusion in the

Algorithm 1 HOSVD

```
1: function HOSVD($\mathcal{X}, \epsilon$)
2:     for $n = 1$ to $d$ do
3:         $[\mathbf{U}, \mathbf{\Sigma}, \sim] = \text{SVD}(\mathbf{X}_{(n)})$        ▷ right singular vectors not computed
4:         $R_n = \min\left\{R| \sum_{i=R_n+1}^{I_n} \sigma_i^2 \leq \epsilon^2 \|\mathcal{X}\|^2/d\right\}$        ▷ determine rank
5:         $\mathbf{U}_n = \mathbf{U}(:, 1 : R_n)$        ▷ $n$th factor matrix
6:     end for
7:     $\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^T \ldots \times_d \mathbf{U}_d^T$
8:     return $(\mathcal{G}, \mathbf{U}_1, \ldots, \mathbf{U}_d)$
9: end function
```

following discussions).

We adopt the method proposed in [3] and [4] by Brand. In this method, when given $\mathbf{b}$, we first find the component of $\mathbf{b}$ that is orthogonal to the range of $\mathbf{U}$ and the norm of this component:

$$\mathbf{e} = \mathbf{b} - \mathbf{M}\mathbf{M}^\mathsf{T}\mathbf{b},\ p = |\mathbf{e}|,\ \text{and } \tilde{\mathbf{e}} = \frac{\mathbf{e}}{|\mathbf{e}|} \tag{2.3}$$

Then using the following identity:

$$\begin{bmatrix}\mathbf{M} & \tilde{\mathbf{e}}\end{bmatrix} \begin{bmatrix}\mathbf{S} & \mathbf{M}^\mathsf{T}\mathbf{b} \\ \mathbf{0} & p\end{bmatrix} \begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix}^\mathsf{T} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\end{bmatrix}, \tag{2.4}$$

Since $\begin{bmatrix}\mathbf{M} & \tilde{\mathbf{e}}\end{bmatrix}$ and $\begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix}$ both have orthonormal columns, we can form the SVD of $\mathbf{Z}$ by computing the SVD of the middle matrix in Eq. (2.4):

$$\mathbf{N}\mathbf{\Sigma}\mathbf{V}^\mathsf{T} = \begin{bmatrix}\mathbf{S} & \mathbf{M}^\mathsf{T}\mathbf{b} \\ \mathbf{0} & p\end{bmatrix}$$

Now Eq. (2.4) can be rewritten as the following:

$$\mathbf{Z} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\end{bmatrix} = \begin{bmatrix}\mathbf{M} & \tilde{\mathbf{e}}\end{bmatrix} \mathbf{N}\mathbf{\Sigma}\mathbf{V}^\mathsf{T} \begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix}^\mathsf{T}, \tag{2.5}$$

By updating $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{W}$ as $\mathbf{M} = \begin{bmatrix}\mathbf{M} & \tilde{\mathbf{e}}\end{bmatrix} \mathbf{N}$, $\mathbf{S} = \mathbf{\Sigma}$, and $\mathbf{W} = \begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix} \mathbf{V}$, we have the SVD of $\mathbf{Z} = \mathbf{M}\mathbf{S}\mathbf{W}^T$. The same process applied when a new vector arrives.

This method avoids computing the SVD of the entire $\mathbf{Z}$ which is $m \times (n+1)$ and instead only compute the SVD of $\begin{bmatrix}\mathbf{S} & \mathbf{M}^\mathsf{T}\mathbf{b} \\ \mathbf{0} & p\end{bmatrix}$ which is $(r+1) \times (r+1)$.

2.4. Loss of orthogonality. Updating $\mathbf{M}$ as $\begin{bmatrix}\mathbf{M} & \tilde{\mathbf{e}}\end{bmatrix} \mathbf{N}$ and $\mathbf{W}$ as $\begin{bmatrix}\mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1\end{bmatrix} \mathbf{V}$ can cause $\mathbf{M}$ and $\mathbf{W}$ to lose orthogonality over many updates due to numerical error. A solution proposed in [4] is to avoid these multiplications and keep $\mathbf{Z}$ represented as the product of 5 matrices, namely:

$$\mathbf{Z} = \mathbf{M}\mathbf{N}\mathbf{S}\mathbf{V}^\mathsf{T}\mathbf{W}^\mathsf{T}$$

This way we only have to update the smaller $\mathbf{N}$ and $\mathbf{V}$ with matrix multiplications, thus making the loss of orthogonality much slower.

In addition to losing orthogonality from the matrix multiplications, appending $\tilde{\mathbf{e}}$ to $\mathbf{M}$ can also causes $\mathbf{M}$ to lose orthogonality as noted in [10] by Zhang. Following Zhang's solution, we project $\tilde{\mathbf{e}}$ again by $(\mathbf{I} - \mathbf{M}\mathbf{M}^{\mathsf{T}})$ once $\left\|\tilde{\mathbf{e}}^{\mathsf{T}}\mathbf{M}(:, end)\right\|$ becomes larger than a threshold value around machine epsilon.

Another issue that arises from numerical error or when computing the truncated SVD of $\mathbf{Z}$ with the above method is that when $\mathbf{b}$ is mostly in the range of $\mathbf{U}$, due to noise in the data and numerical error, p might not be exactly 0. This will cause the rank to increase when it shouldn't. To address this issue [4] proposes to set a tolerance around the machine epsilon or the noise level. If p is less than said tolerance, p is set to 0. This way the rank will not increase. In addition, $\mathbf{M}$ and $\mathbf{W}$ will remain unchanged. However, in our experiments, we find that for matrices that are not exactly low rank or when the singular values decay more slowly, how to determine the appropriate tolerance for p becomes less clear. The incremental SVD algorithm we use is shown in Algorithm 2.

3. Algorithm.

We continue our discussion in this section with the notations used in Fig. 1.1 in mind. Without taking advantage of the Tucker decomposition of $\mathcal{X}$, we can carry out the HOSVD algorithm on $\mathcal{Z}$. As we have discussed before, computing the SVD of each unfolding of the tensor makes up a significant portion of the total cost of HOSVD. As more and more updates arrive indefinitely, the unfolding of $\mathcal{Z}$ grows indefinitely. Thus computing the SVD of those unfoldings becomes infeasible eventually. Our solution to this problem, for the first $d-1$ modes, is to use the Gram-SVD as shown in Algorithm 3. The right half of Fig. 1.1 shows the relationship between the unfolding of the new tensor $\mathcal{Z}$ and those of $\mathcal{X}$ and $\mathcal{Y}$ on each mode in the 3D case. As we can see, for the first 2 modes, the unfolding of $\mathcal{Z}$ is formed by concatenating the unfoldings of $\mathcal{X}$ and $\mathcal{Y}$ horizontally. For the first $d-1$ modes, the Gram matrix of $\mathbf{Z}_{(n)}$ can be computed as follows:

$$\mathbf{G} = \mathbf{Z}_{(n)}\mathbf{Z}_{(n)}^{\mathsf{T}} = \mathbf{X}_{(n)}\mathbf{X}_{(n)}^{\mathsf{T}} + \mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^{\mathsf{T}}$$

Notice that $\mathbf{X}_{(n)}\mathbf{X}_{(n)}^{\mathsf{T}}$ would have been computed in the previous time step. In addition, since we assume each update $\mathcal{Y}$ has the same size, $\mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^{\mathsf{T}}$ have the same cost for each update.

On the other hand, for the last mode, $\mathbf{Z}_{(d)}$ is formed by concatenating $\mathbf{X}_{(d)}$ and $\mathbf{Y}_{(d)}$ vertically. This makes the Gram-SVD method less suitable since the Gram matrix of $\mathbf{Z}_{(n)}$ would grow in size as updates stream in. Therefore we adopt the incremental SVD method introduced in Section 2.3. Note that the incremental SVD method that we summarized assumes the update is a column vector while $\mathbf{Y}_{(d)}$ is a row vector. To address this we simply compute the incremental SVD of $\mathbf{Z}_{(d)}^{\mathsf{T}}$ and use its right singular vectors instead of left singular vectors as the factor matrix. For the tolerance of p, we used $\epsilon\,\|\mathcal{Y}\|$. Note that in Line 10 we take the leading r singular vectors to form the last factor matrix $\tilde{\mathbf{U}}_d$ while not discarding the trailing singular vectors. We find that in practice this improves the accuracy while increasing the computational cost. Another potential way to do truncation is to increase the tolerance for p so that iSVD() can filter out the less significant singular vectors so that Line 10 becomes unnecessary. However, in practice, we find this to cause the accuracy of the iSVD() to deteriorate over updates quickly.

To update the core tensor without having to access the whole $\mathcal{Z}$, we employ the blocked tensor times matrix method proposed by Xiao et al. in [9]. The main result that we will use is the following: Let $\mathcal{X} = \mathcal{G} \times_1 \tilde{\mathbf{U}}_1 \times_2 \tilde{\mathbf{U}}_2 \times \cdots \times_d \tilde{\mathbf{U}}_d$ where the size of $\mathcal{X}$ is $I_1 \times I_2 \times \cdots \times I_d$. Let $\mathcal{Y}$ be a $I_1 \times I_2 \times \cdots \times n$ tensor (n is any positive integer) and $\mathcal{Z}$ be the result of concatenating

Algorithm 2 Incremental SVD

```
function iSVD($\mathbf{M}$, $\mathbf{N}$, $\mathbf{S}$, $\mathbf{V}$, $\mathbf{W}$, $\mathbf{b}$, tol)
    $\mathbf{t} = \mathbf{M}^{\mathsf{T}}\mathbf{b}$
    $\mathbf{d} = \mathbf{N}^{\mathsf{T}}\mathbf{t}$
    $\mathbf{e} = \mathbf{b} - \mathbf{M}\mathbf{t}$
    $p = |\mathbf{e}|$
    $\tilde{\mathbf{e}} = \frac{\mathbf{e}}{p}$
    if $p >$ tol then
        if $\sqrt{|\tilde{\mathbf{e}}^{\mathsf{T}}\mathbf{M}(:, \mathrm{end})|} <$ tol then
            $\tilde{\mathbf{e}} = \tilde{\mathbf{e}} - \mathbf{M}(\mathbf{M}^{\mathsf{T}}\tilde{\mathbf{e}})$        ▷ Reorthogonalize $\tilde{\mathbf{e}}$ if needed
            $\tilde{\mathbf{e}} = \frac{\tilde{\mathbf{e}}}{|\tilde{\mathbf{e}}|}$
        end if
        $\mathbf{S} = \begin{bmatrix} \mathbf{S} & \mathbf{d} \\ \mathbf{0} & p \end{bmatrix}$
        $[\tilde{\mathbf{U}}, \tilde{\mathbf{S}}, \tilde{\mathbf{V}}] = \mathrm{svd}(\mathbf{S})$
        $\mathbf{M} = \begin{bmatrix} \mathbf{M} & \tilde{\mathbf{e}} \end{bmatrix}$
        $\mathbf{W} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$
        $\mathbf{N} = \begin{bmatrix} \mathbf{N} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{U}}$
        $\mathbf{S} = \tilde{\mathbf{S}}$
        $\mathbf{V} = \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{V}}$
    else
        $\mathbf{S} = \begin{bmatrix} \mathbf{S} & \mathbf{d} \end{bmatrix}$
        $[\tilde{\mathbf{U}}, \tilde{\mathbf{S}}, \tilde{\mathbf{V}}] = \mathrm{svd}(\mathbf{S})$
        $\mathbf{N} = \mathbf{N}\tilde{\mathbf{U}}$
        $\mathbf{S} = \tilde{\mathbf{S}}$ without its last column
        Remove the last column of $\tilde{\mathbf{V}}$
        $\tilde{\mathbf{V}} = \begin{bmatrix} \tilde{\mathbf{V}}_1 \\ \tilde{\mathbf{V}}_2 \end{bmatrix}$        ▷ Split $\tilde{\mathbf{V}}$ into its last row and the rest.
        $\mathbf{V} = \mathbf{V}\tilde{\mathbf{V}}_1$
        $\mathbf{W} = \begin{bmatrix} \mathbf{W} \\ \tilde{\mathbf{V}}_2\mathbf{V}^{+} \end{bmatrix}$
    end if
    return $\mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}$
end function
```

$\mathcal{X}$ and $\mathcal{Y}$ along mode d. Then we have:

$$\mathcal{Z} \times_1 \mathbf{U}_1 \times \cdots \times_d \mathbf{U}_d = (\mathcal{G} \times \mathbf{U}_1\tilde{\mathbf{U}}_1 \times \cdots \times_d \mathbf{U}_{d_1}\tilde{\mathbf{U}}) + (\mathcal{Y} \times_1 \mathbf{U}_1 \times \cdots \times_d \mathbf{U}_{d_2}) \tag{3.1}$$

where $\mathbf{U}_i \in \mathbb{R}^{R_i \times I_i}$ for $i \leq d-1$ and $\mathbf{U}_d \in \mathbb{R}^{R_d \times (I_d+n)}$. $\mathbf{U}_{d_2}$ and $\mathbf{U}_{d_1}$ are the last n rows of $\mathbf{U}_d$ and the rest of its rows, respectively.

Our proposed update algorithm is shown in Algorithm 4. To clarify the notations, let the previously available tensor be $\mathcal{X}$ and $\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times \cdots \times_d \mathbf{U}_d$ and $\mathbf{G}_i = \mathbf{X}_{(i)}\mathbf{X}_{(i)}^{\mathsf{T}}$ for $i \in [1, d-1]$. The ϵ input parameter is the desired reconstruction error of the Tucker decomposition. Algorithm 5 shows the how the problem is initialized and how Algorithm 4 will be used.

Algorithm 3 Gram-SVD

```
1: function GRAM-SVD($\mathbf{X}$)
2:     $\mathbf{G} = \mathbf{X}\mathbf{X}^\intercal$
3:     $[\mathbf{E}, \mathbf{\Lambda}] = \text{eig}(\mathbf{G})$
4:     return $\mathbf{G}$, $\mathbf{E}$, $\mathbf{\Lambda}$
5: end function
```

Algorithm 4 Update

```
1: function UPDATE($\mathcal{G}, \{\mathbf{U}_1, \ldots, \mathbf{U}_d\}, \{\mathbf{G}_1, \ldots, \mathbf{G}_{d-1}\}, \mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}, \epsilon, \mathcal{Y}$)
2:     for $n = 1 : (d-1)$ do
3:         $\mathbf{G}_n = \mathbf{G}_n + \mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^T$
4:         $[\mathbf{E}, \mathbf{\Lambda}] = \text{EIG}(\mathbf{G}_n)$
5:         $r = \text{argmin}_r \sum_{i=r+1}^{I_n} \lambda_i < \epsilon^2(\|\mathcal{X}\|^2 + \|\mathcal{Y}\|^2)/d$
6:         $\tilde{\mathbf{U}}_n = \mathbf{E}_{(:,1:r)}$
7:     end for
8:     $\mathbf{y} = \mathcal{Y}(:)$                    ▷ Convert the update tensor into a vector
9:     $[\mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}] = \text{ISVD}(\mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}, \mathbf{y}, \epsilon\,\|\mathbf{y}\|)$
10:    $r = \text{argmin}_r \sum_{i=r+1}^{I_n} \mathbf{S}_{(i,i)}^2 < \epsilon^2(\|\mathcal{X}\|^2 + \|\mathcal{Y}\|^2)/d$
11:    $\tilde{\mathbf{U}}_d = \mathbf{W}\mathbf{V}_{(:,1:r)}$
12:    $\tilde{\mathbf{U}}_d = \begin{bmatrix}\tilde{\mathbf{U}}_{d_1} \\ \tilde{\mathbf{U}}_{d_2}\end{bmatrix}$     ▷ Let the last row of $\tilde{\mathbf{U}}_d$ be $\hat{\mathbf{U}}_2$ and the rest be $\hat{\mathbf{U}}_1$.
13:    $\mathcal{G} = \mathcal{G} \times_1 \tilde{\mathbf{U}}_1^\intercal \mathbf{U}_1 \times_2 \tilde{\mathbf{U}}_2^\intercal \mathbf{U}_2 \times \cdots \times_d \tilde{\mathbf{U}}_{d_1}^\intercal \mathbf{U}_d + \mathcal{Y} \times_1 \tilde{\mathbf{U}}_1^\intercal \times_2 \tilde{\mathbf{U}}_2^\intercal \times \cdots \times_d \tilde{\mathbf{U}}_{d_2}^\intercal$
14:    return $\mathcal{G}, \{\mathbf{U}_1, \ldots, \mathbf{U}_d\}, \{\mathbf{G}_1, \ldots, \mathbf{G}_{d-1}\}, \mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}$
15: end function
```

3.1. Complexity. In this section, we analyze the complexity of our update algorithm as shown in Algorithm 4. For simplicity, we assume that the input tensor has the same size, I, for its first $d-1$ modes and size I_d for its last mode. We also denote the rank of $\mathbf{X}_{(n)}$ as r_n for $n \in [1, d]$. Note that r can increase as updates arrive. For $n \in [1, d-1]$, r_n is bounded above by I while we consider r_d to have no upper limit.

For the first $d-1$ Gram-SVD, computing $\mathbf{Y}_{(n)}\mathbf{Y}_{(n)}^\intercal$ dominates the cost with its cost being $O(2I^d)$.

For the incremental SVD algorithm, we list the costs of important steps of the algorithm in Table 3.1. As we can see, projecting the update vector $\mathbf{b}$ will be the more expensive cost when $r \ll I$, but as the rank of $\mathbf{X}_{(d)}$ increases, the cost of computing the SVD and pseudo-inverse will become significant. The other major cost expense is updating the core tensor.

Operation	Cost/
$\mathbf{M}^\intercal\mathbf{b}$	$O(2rI^{d-1})$
$\mathbf{b} - \mathbf{M}\mathbf{t}$	$O(2rI^{d-1})$
Update $\mathbf{N}$ and $\mathbf{V}$	$O(2r^3)$
svd($\mathbf{S}$)	$O(r^3)$
Computing $\mathbf{V}^+$	$O(r^3)$

TABLE 3.1
Cost of iSVD

Computing the first half of Line 13 costs $O(2\prod_{n=1}^{d} r_n^2 I + 2r^{d+1})$. Computing the second half costs $O(2r_1 I^{d-1})$.

Algorithm 5 Streaming Tucker decomposition

```
1: function s-Tucker($\mathcal{X}, \{\mathcal{Y}_1, \ldots, \mathcal{Y}_m\}, \epsilon$)        ▷ $\mathcal{Y}_i$ = update tensor at time step $i$
2:     for $n = 1 : (d-1)$ do
3:         $[\mathbf{G}_n, \mathbf{E}, \mathbf{\Lambda}] = \text{Gram-SVD}(\mathbf{X}_{(n)})$
4:         $r = \text{argmin}_r \sum_{i=r+1}^{I_n} \lambda_i < \epsilon^2 \|\mathcal{X}\|^2 / d$
5:         $\mathbf{U}_n = \mathbf{E}_{(:,1:r)}$
6:     end for
7:     $[\mathbf{M}, \mathbf{S}, \mathbf{W}] = \text{svd}(\mathbf{X}_{(d)})$
8:     $r = \text{argmin}_r \sum_{i=r+1}^{I_n} \mathbf{S}_{(i,i)}^2 < \epsilon^2 \|\mathcal{X}\|^2 / d$
9:     $\mathbf{S} = \mathbf{S}_{(1:r,1:r)}$
10:    $\mathbf{U}_d = \mathbf{M} = \mathbf{M}_{(:,1:r)}$
11:    $\mathbf{W} = \mathbf{W}_{(:,1:r)}$
12:    $\mathbf{N} = \mathbf{V} = \mathbf{I}$        ▷ $\mathbf{I}$ is $r \times r$ identity matrix
13:    $\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^\mathsf{T} \times_2 \mathbf{U}_2^\mathsf{T} \times \cdots \times_d \mathbf{U}_d^\mathsf{T}$
14:    for $i = 1 : m$ do
15:        $[\mathcal{G}, \{\mathbf{U}_1, \ldots, \mathbf{U}_d\}, \{\mathbf{G}_1, \ldots, \mathbf{G}_{d-1}\}, \mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}] =$
16:            Update($\mathcal{G}, \{\mathbf{U}_1, \ldots, \mathbf{U}_d\}, \{\mathbf{G}_1, \ldots, \mathbf{G}_{d-1}\}, \mathbf{M}, \mathbf{N}, \mathbf{S}, \mathbf{V}, \mathbf{W}, \epsilon, \mathcal{Y}_i$)
17:    end for
18: end function
```

Note that the cost of all three phases of our update algorithm do not involve I_d and instead depends on r. This entails that as updates arrive, the cost of computing each update will increase much slower or stay the same.

4. Numerical experiments. To examine the accuracy of our algorithm and its performance, we run our s-Tucker() algorithm on the Homogeneous Charge Compression Ignition (HCCI) dataset [2]. This data is generated from a numerical simulation of a combustion process. The entire dataset is $627 \times 627 \times 33 \times 626$ where the first 2 modes represent a 2D spatial grid and the third mode represents different variables and the last mode represents 626 time steps. In this experiment, we only use the 29th variable and the 200th to 400th time step. The resulting tensor is $627 \times 627 \times 200$. We choose the 200-250th time step as our initial input tensor. Starting from the 251st time step, each update contains only 1 time step. We used 150 updates in total. The tolerance is set to $\epsilon = 10^{-4}$.

We tracked the reconstruction error of the Tucker representation computed by our algorithm in Fig. 4.1. The reconstruction error is computed as the following:

$$err = \frac{\|\mathcal{Z} - \mathcal{G} \times \mathbf{U}_1 \times \cdots \times \mathbf{U}_d\|}{\|\mathcal{Z}\|}. \tag{4.1}$$

We can see that the error initially increases but decreases eventually to satisfy our specified tolerance.

Another interesting value to track is the rank of $\mathbf{Z}_{(3)}$ after each update. As we have mentioned in the previous section, the rank of $\mathbf{Z}_{(3)}$ computed by the incremental SVD method might not be the same as the number of columns of $\mathbf{U}_3$ which can also differ from the rank of $\mathbf{Z}_{(3)}$ returned by the truncated Gram-SVD of the full $\mathbf{Z}_{(3)}$. In Fig. 4.2, we record the three values after each update. At the end of 150 updates, the size of $\mathbf{S}$ reaches 88, which is significantly better than no truncation. However, when compared to the actual rank computed by the Gram-SVD, this rank is still a significant overestimation. We can also see that by doing a truncation on $\mathbf{S}$ (Line 10 of Algorithm 4), we can get a relatively accurate estimate of the rank of $\mathbf{Z}_{(3)}$.

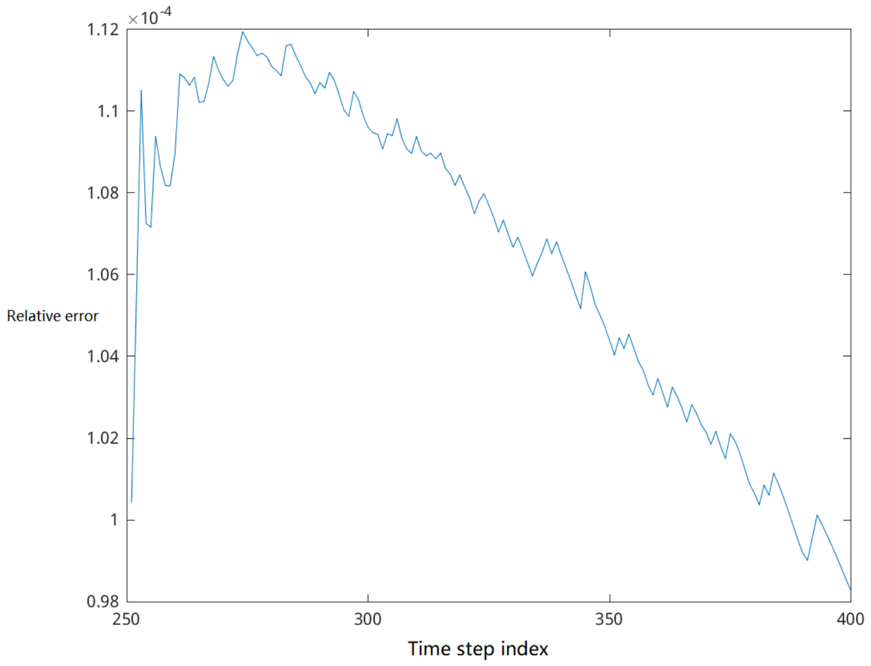


Fig. 4.1. *Reconstruction error after each update*

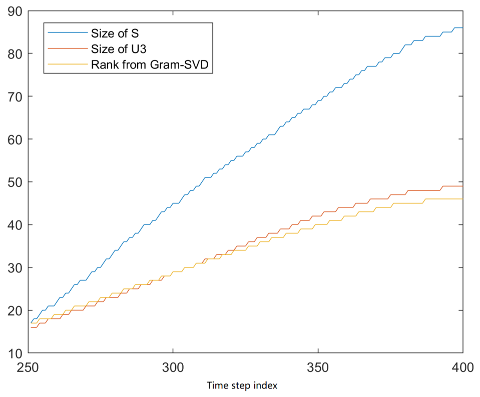


Fig. 4.2. *Rank of* $\mathbf{Z}_{(3)}$ *after each update*

The time spent on each update is shown in Fig. 4.3. We break down each update into roughly 3 phases: updating the factor matrices of the first $d-1$ mode with Gram-SVD (blue), updating the factor matrix of the last mode with incremental SVD (orange and yellow), updating the core, and others. As we can see, the time spent on updating the factor matrices for the first $d-1$ modes stays constant even as more and more updates arrive. The time spent on iSVD fluctuates with a slowly increasing trend. For the incremental SVD algorithm, in this case, most of the time (yellow bar) is spent on projecting the update vector as represented in Line 2 and Line 4 of Algorithm 2. The spikes in iSVD time in certain iterations are caused by the reorthogonalization that happens in Line 9. Finally, we can see that the time spent on updating the core increases but at a slow pace as well. The general increasing trend of the cost of the whole algorithm is caused by the increase of the rank of $\mathbf{Z}_{(3)}$. We believe that when given a streaming tensor whose rank of the last mode stops increasing at some point, the cost of each update will stop increasing as well.

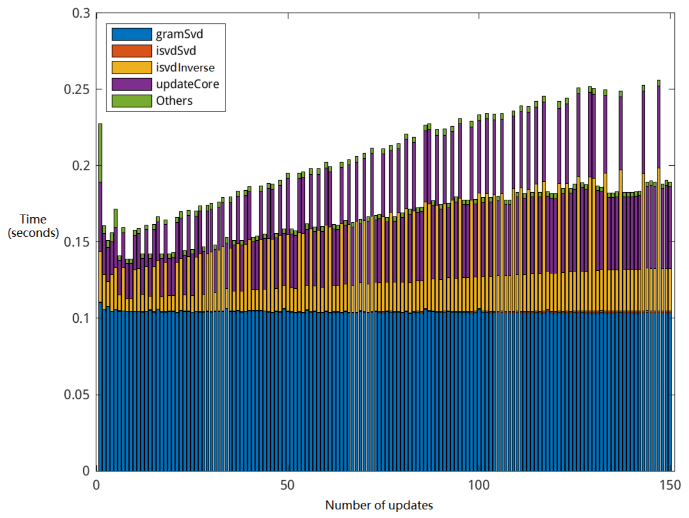


Fig. 4.3. *Total time spent for each update.*

5. Discussion. In this section we list some of the questions left unanswered in this paper and discuss potential solutions as future research directions.

As we have mentioned, how the tolerance of p changes the accuracy of the incremental SVD when the singular values decrease gradually is unclear. In our implementation, we used $\epsilon|\mathbf{b}|$, the reconstruction error tolerance times the norm of the update vector. As we have seen in the previous section, the reconstruction error turns out to satisfy the tolerance we specified. However, this is at the cost of having a larger $\mathbf{S}$. In our experiments, once we increase this tolerance of p (we tried $\epsilon|\mathbf{Z}|$), the size of $\mathbf{S}$ decreases as expected. On the other hand, the accuracy of incremental SVD starts to deteriorate as the updates stream in.

Another limitation of this algorithm comes from the fact that we use Gram-SVD to compute and update the first $d-1$ factor matrices. As we know, Gram-SVD is not numerically stable and has trouble computing the smallest singular values when the singular values have high variance. This sets a limit on the reconstruction error tolerance we can achieve with this algorithm.

REFERENCES

[1] S. Ahmadi-Asl, S. Abukhovich, M. G. Asante-Mensah, A. Cichocki, A. H. Phan, T. Tanaka, and I. Oseledets, *Randomized algorithms for computation of tucker decomposition and higher order svd (hosvd)*, IEEE Access, 9 (2021), pp. 28684–28706.

[2] A. Bhagatwala, J. H. Chen, and T. Lu, *Direct numerical simulations of HCCI/SACI with ethanol*, Combustion and Flame, 161 (2014), pp. 1826–1841.

[3] M. Brand, *Incremental Singular Value Decomposition of Uncertain Data with Missing Values*, in Computer Vision — ECCV 2002, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, eds., Lecture Notes in Computer Science, Berlin, Heidelberg, 2002, Springer, pp. 707–720.

[4] ———, *Fast low-rank modifications of the thin singular value decomposition*, Linear Algebra and its Applications, 415 (2006), pp. 20–30.

[5] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-(r1 ,r2 ,. . .,rn) approximation of higher-order tensors*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342.

[6] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[7] L. T. Thanh, K. Abed-Meraim, N. Linh-Trung, and A. Hafiane, *A Contemporary and Comprehensive Survey on Streaming Tensor Decomposition*, June 2022.

[8] N. Vannieuwenhoven, R. Vandebril, and K. Meerbergen, *A New Truncation Strategy for the Higher-Order Singular Value Decomposition*, SIAM J. on Sci. Comp., 34 (2012), pp. A1027–A1052.

[9] H. Xiao, F. Wang, F. Ma, and J. Gao, *eOTD: An Efficient Online Tucker Decomposition for Higher Order Tensors*, in 2018 IEEE International Conference on Data Mining (ICDM), Nov. 2018, pp. 1326–1331. ISSN: 2374-8486.

[10] Y. Zhang, *An answer to an open question in the incremental SVD*, Apr. 2022. Number: arXiv:2204.05398 arXiv:2204.05398 [cs, math].

RE-ARCHITECTING STORAGE SYSTEMS FOR MODERN HARDWARE: IS IT WORTH IT?

LUKE LOGAN AND MENTOR JAY LOFSTEAD*

Abstract. Traditionally, distributed storage systems have relied upon the interfaces provided by the OS kernel to interact with storage hardware. However, much research has shown that OSes impose serious overheads on every I/O operation, especially on high-performance storage and networking hardware (e.g., persistent memory and 200GBe). Thus, distributed storage stacks are being re-designed to take full advantage of this modern hardware by utilizing new hardware interfaces which bypass the kernel entirely. However, the impact of these optimizations has not been well-studied at the distributed level on real hardware. In this work, we provide a comprehensive evaluation of DAOS: a state-of-the-art distributed storage system which re-architects the storage stack from scratch to optimize for high-performance hardware. We compare the various storage and networking backends provided by DAOS and demonstrate that by utilizing optimal interfaces to hardware, performance improvements of up to 15x can be observed in real scientific and machine-learning applications.

1. Introduction.

HPC systems have traditionally suffered from an I/O bottleneck due to the gap between storage and CPU performance [1]. To lower this gap, HPC centers are adopting high-performance storage (e.g., PMEM) and networking (e.g., 200GBe) hardware. This hardware is typically organized in a hierarchy [1–5], where data is initially buffered in a high-performance tier and then flushed to a high-capacity tier. This improvement in hardware provides substantial benefits to overall application performance. However, while performance benefits can be observed, many storage systems were designed under the assumption that I/O is significantly slower than compute. This assumption is no longer true, with many works noting significant performance loss caused by software overheads [6–8] and context switching [9, 10] under various workloads. To fully take advantage of the low latency and high bandwidth provided by this hardware, the design and implementation of storage systems are being thoroughly reconsidered.

One source of performance degradation is caused by the storage and networking stacks provided by OS kernels. The Linux I/O stack, for example, is well-tested and provides strong data consistency guarantees, while maintaining acceptable performance for general users. However, many high-performance storage systems rely on the Linux I/O stack for interacting with storage. Many works have demonstrated that the Linux I/O stack causes significant performance loss due to its lengthy I/O path [11–13], interrupts [9, 10], and context switches [9, 10]. To improve this, new kernel-bypass storage stacks [11, 14, 15] have been developed to take full advantage of the characteristics of modern hardware. One such work is the SPDK [15], an NVMe driver that utilizes userspace function calls and polling instead of system calls and interrupts to interact with hardware. Many single-node storage systems (e.g., filesystems [12, 13] and key-value stores [8, 11]) have been developed on top of these kernel-bypass stacks and have demonstrated significant improvements to I/O performance in terms of latency. However, the evaluations of these works are only at the single-node.

Due to the significant performance impacts of kernel-bypass and hardware optimization on single-node storage stacks, distributed storage stacks are also emerging that aim to fully utilize modern hardware. Many well-established storage systems are being patched to better support modern hardware. One such system is CephFS [16], a distributed filesystem that recently replaced kernel-level filesystems for storing metadata in NVMe and PMEM with kernel-bypass technologies [17]. However, since existing systems were already largely designed towards slower storage mediums, other works decided to re-architect the storage stack completely [18–20]. For example, the Distributed Asynchronous Object Store (DAOS) [18] by Intel is a state-of-the-art storage system which re-architects the I/O path entirely, bypassing I/O paths which are known to be slow.

*Sandia National Laboratories, gflofst@sandia.gov

While many distributed storage stacks have been proposed to take advantage of modern hardware, the impact of these different optimizations has not been well-evaluated. For the most part, evaluations of storage systems that incorporate kernel-bypass technologies are limited to single-node cases, and many of these evaluations are over emulated hardware [12, 13, 21, 22] or outdated software versions. For example, to the best of our knowledge, there has been no published benchmarks of DAOS 2.0, nor has there been any comparison of DAOS against other storage systems over the same hardware configuration and software stack. In addition, all known published benchmarks of DAOS assume its optimal configuration, and do not demonstrate the performance impacts of the various kernel-based and kernel-bypass networking and storage backends provided by DAOS. Overall, from the existing work, it is not clear the extent to which re-desigining storage stacks for modern hardware has affected the performance of distributed scientific applications.

In this work, we aim to quantify the extent to which re-architecting storage stacks for modern hardware truly impacts the overall performance of modern distributed scientific applications. To do this, we provide a comprehensive evaluation of DAOS 2.0: a state-of-the-art distributed storage stack that has been built from scratch to maximize the benefits of modern storage and networking hardware. We quantify the performance impacts of DAOS's diverse software stack optimizations compared to traditional kernel-based approaches over modern storage and networking hardware, including NVMe and PMEM. We demonstrate that DAOS outperforms traditional storage systems, such as OrangeFS, by as much as 15x under various real workloads on high-performance storage hardware.

2. Background & Related Work. There has been growing interest in optimizing storage stacks to maximize the bandwidth and latency potential of storage. This effort has spanned from OS-level changes to device drivers to entire distributed storage system designs. Various works have proposed new storage stack designs, and some work has been done to evaluate the implications of these new designs at the distributed level.

2.1. Single-Node Storage Stacks. Traditionally, storage devices are protected by the OS kernel through system calls. Therefore, applications interact indirectly with storage devices using I/O system calls which typically involves interrupts and context switching between user processes and the kernel. However, this approach has been shown to cause significant performance degradation [8, 9, 11, 15, 23] on high-performance storage hardware. Modern storage systems have aimed to bypass these overheads by offering alternative approaches for interacting with hardware through the use of userspace drivers and memory mapping. Many single-node storage systems have been developed on top of these new mechanisms that offer performance much closer to the theoretical characteristics of the hardware.
Hardware APIs: The Storage Performance Development Kit (SPDK) [15] is an Intel open-source project that provides a set of tools and libraries for writing userspace storage applications for NVMe devices. SPDK leverages the NVMe specification that allows the mapping of the PCI Base Address Register (BAR) into user applications. Similarly, LightNVM [24] is a userspace driver for OpenChannel SSDs that enables the programming of the Flash Translation Layer (FTL) and submission of I/O requests to OpenChannel SSDs entirely in userspace. The Dataplane Development Kit (DPDK) [25] provides a userspace polling API to a variety of networking cards and has been shown to minimize latency for networking operations. File Direct Access (DAX) [26] is a methodology for treating PMEM as regular memory through the use of memory mapping, avoiding context switching and metadata management overheads. Lastly, the Demikernel [27] provides an abstraction over most of the aformentioned kernel-bypass accelerators in order to improve storage system development speeds.
Filesystems & Key-Value Stores: To provide general users a simple, high-performance interface to hardware, various storage systems (e.g., POSIX filesystems) have been built on top of these userspace drivers to maximize performance. Nova [12] is a log-structured filesystem

that is designed to minimize latency and maximize scalability of PMEM devices. SplitFS [13] is another PMEM filesystem that utilizes the production-ready EXT4 filesystem for metadata management, but uses memory-mapped I/O in userspace, avoiding the overhead of system calls typically incurred by the EXT4 filesystem. Another work, Simurgh [28], proposes a PMEM filesystem that implements all I/O functionality within a single library while protecting against buggy, but not necessarily malicious, code. This approach avoids software overhead of the kernel entirely for PMEM. For NVMe, uFS [29] proposes a filesystem that dedicates CPU resources to handle filesystem threads to minimize latency on NVMes while also maintaining security guarantees. While many of these storage stacks have been proposed, they were designed only for single-node applications and were therefore evaluated primarily at the single node. The impact of the optimizations proposed in these works are not well-understood at a distributed level.

2.2. Distributed Storage Stacks. While many changes have been made to the design of single-node storage stacks, the impacts of these changes must also be understood at the distributed level. Due to the success of single-node I/O stack optimizations, distributed storage stacks are also evolving to optimize for the high-performance of modern storage hardware. Some storage systems are being patched to utilize these new interfaces while making minimal changes to their overall design, whereas other systems have been developed entirely from scratch.

Traditional Storage Stacks: OrangeFS [30] and BeeGFS [31] are both traditional high-end computing storage systems. OrangeFS has shipped with the Linux kernel since 4.6 and provides a userspace FUSE plugin for performing I/O. BeeGFS comes with a custom kernel module that acts as a kernel-level filesystem. When performing I/O, OrangeFS and BeeGFS divide data into stripes and distributes them among storage servers. The locations of stripes are managed by the metadata servers. In both systems, metadata can be partioned across a set of metadata servers (servers can be spread across nodes) to achieve scalability. It is up to the user to decide which nodes contain metadata servers and storage servers. Both systems currently utilize the I/O interfaces provided by the OS, and have not been specifically optimized to support high-performance modern hardware, outside of RDMA capabilities.

Patched Storage Stacks: CephFS [16] is a POSIX-compliant filesystem built on top of Ceph's distributed object store, RADOS. CephFS endeavors to provide a state-of-the-art, multi-use, highly available, and performant file store for a variety of applications, including traditional use-cases like shared home directories, HPC scratch space, and distributed workflow shared storage. CephFS has recently replaced kernel-level filesystems for storing metadata with BlueStore, which utilizes kernel-bypass technologies [17], including the SPDK and DAX. The work demonstrated significant performance improvement of their kernel-bypass storage backend. The evaluation of this work was primarily over a 16-node cluster using RAM and HDD, and was not evaluated over PMEM. Other works have benchmarked CephFS over NVMe [32]. CephFS is also developing SeaStore [33], which currently focuses solely on NVMe optimizations, primarily using the zoned-namespace APIs provided by modern NVMes. PMEM optimizations are not yet considered in this work.

Modular Storage Stacks: Recent efforts have been aiming to provide modularized storage stacks in order to combat software overheads caused by the OS kernel and provide workload- and hardware-specific customization. Mochi [34] provides a diverse set of building blocks, which can be composed by users to rapidly build highly customized distributed storage systems. LabStor [11] proposes an extensible platform to facilitate the rapid development of new, hardware-optimized, workload-specific storage stacks. Users can develop a wide variety of storage modules, ranging from per-node I/O schedulers to distributed storage systems, which can then be composed to form highly optimized I/O stacks. The evaluations of these works were primarily over emulated hardware and NVMe.

DAOS: The Distributed Asynchronous Object Store (DAOS) [18] is a state-of-the-art storage system developed by Intel to take full advantage of modern hardware, such as NVMe and PMEM. DAOS requires the existence of either NVMe or PMEM for acting as a persistent cache to metadata. For NVMe, the SPDK can be used for interacting directly with NVMe while bypassing the kernel. For PMEM, DAX can be used for mapping PMEM directly into DAOS's address space, minimizing kernel overheads.

DAOS has two storage concepts: pools and containers. A DAOS pool abstracts over the storage hardware for a subset of storage nodes. A pool can contain a variety of different storage hardware. The storage in a DAOS pool is tiered based off of the type of storage (e.g., NVMe vs PMEM). A DAOS container, which is a transactional object store, can be allocated from a DAOS pool. A DAOS container can represent many different storage systems. For example, a container can be exposed as a filesystem or a key-value store. There can be up to a few hundred containers. By default, DAOS recommends 6% of a container's allocated space be PMEM/NVMe while 94% be allocated to mass storage (e.g., HDD). This ensures that all metadata will be stored in high-performance tiers, while all data operations are stored in slower, high-capacity tiers. Metadata is required to be stored in either PMEM or DRAM, and it is assumed that a user's pool will contain a sufficient amount of space to store metadata.

DAOS is intended to scale to hundreds of thousands of nodes. In this case, it is expected that there will be no more than a few hundred pools, and no more than a few hundred containers per-pool. Each container can store billions of objects (e.g., files). While these are not hard limits, going above these recommendations may cause performance issues.

DAOS provides a variety of interfaces for interacting with containers, including POSIX, MPI-IO, HDF5, Spark and Hadoop. For POSIX, both a FUSE filesystem and an interceptor (which can be loaded using LD_PRELOAD) are provided. The filesystem interceptor is intended to reduce the overheads imposed by FUSE. Unlike a typical parallel filesystem such as OrangeFS, DAOS stores variable-sized blobs of information and does not use fixed-sized striping. This avoids high-latency accesses for I/O operations smaller than the stripe size.

Using PMEM and NVMe, DAOS 1.0 was able to achieve top marks on the IO500 list at sc19 [35]. Currently, there are no published benchmarks of DAOS 2.0.

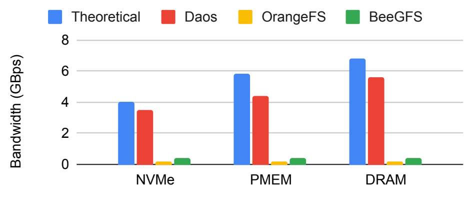
The Future of Modern Hardware: Recently, Intel announced that their Optane PMEM [36] product line has been discontinued. However, this doesn't ruin the future of DAOS, nor the future of PMEM. While Intel is not marketing this specific product anymore, new technologies such as Compute Express Link (CXL) storage and memory [37] are emerging, which have similar properties to PMEM. In addition, DAOS has significant optimization for NVMe, which is already well-established.

2.3. Motivation. While many proposals have been made to change the way storage systems are designed to incorporate the high-performance of modern storage hardware, it is not well-understood how these design choices have impacted the performance of distributed applications over real modern hardware. Evaluations of single-node storage stacks have primarily been in a single-node setting, where network costs are avoided entirely. Evaluations that show distributed impacts are typically conducted over emulated hardware or only NVMe. To truly understand the performance impacts of optimizing storage stacks for modern hardware, an evaluation across a state-of-the-art system and a traditional system over real hardware is necessary.

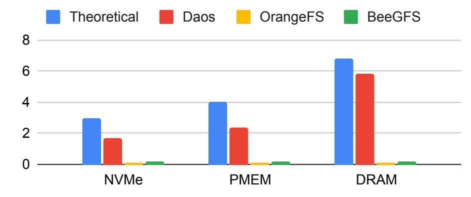
3. Evaluations. Hardware: We ran all our experiments on a 3-node cluster. Each node contains 512GB RAM, 8x 256TB of Intel Optane DC Persistent Memory, and 16x 4TB NVMes. Each node uses Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz. Each CPU has 24 cores and 48 threads, and there are two CPUs per-node. In total, there are 144 cores and 288 threads. The network interconnect is 100GBe.

Software: For our experiments, we use Centos8 with kernel 4.18. We install DAOS 2.1.104-tb, OrangeFS 2.9.8, and BeeGFS 3.7.1. For benchmarks, we use IO500 (isc'22 branch) and DLIO (commit: 2a5ed47). We use mpich 3.3.2 for all experiments. Each experiment is executed 3 times and the average is reported.

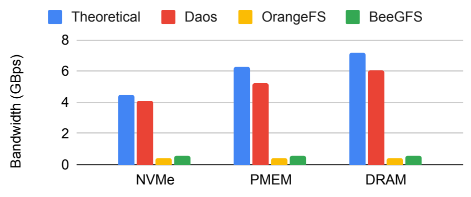
Experiment Setup: In each experiment, we run the workload generator with 128 processes. In each test, we co-locate the application with the storage/metadata servers. Caches are cleared before every experiment. We use the default configuration for OrangeFS and BeeGFS in all experiments. For OrangeFS, a 64KB stripe size is used and libaio is used as the I/O backend. BeeGFS is configured with a 512KB stripe size. BeeGFS automatically scales I/O threads depending on the current workload. We use up to 1x NVMe per-node for these tests. For both OrangeFS and BeeGFS, we deploy one storage and metadata server on each node. For DAOS, the sample configuration dedicates a single core for handling I/O operations to the hardware. This parameter could be configured to better utilize hardware parallelism. More detail about experimental setup is in each evaluation.



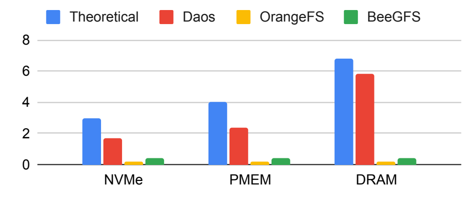
(a) IOR-Easy Write



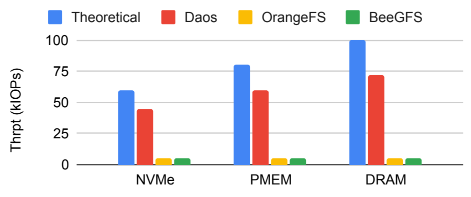
(b) IOR-Hard Write



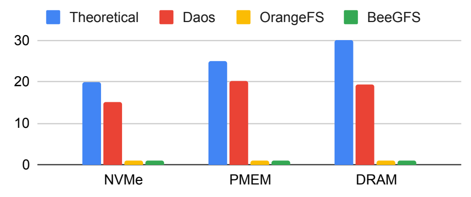
(c) IOR-Easy Read



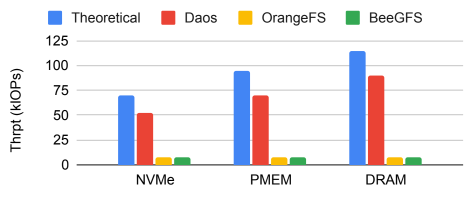
(d) IOR-Hard Read



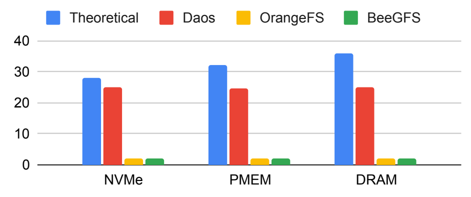
(e) Mdtest Easy Write



(f) Mdtest Hard Write



(g) Mdtest Easy Read



(h) Mdtest Hard Read

FIG. 3.1. *IO500 over various hardware and storage systems. DAOS outperforms OrangeFS and BeeGFS by at least 10x in every workload.*

3.1. IO500. In this evaluation, we measure the performance benefits of utilizing storage systems that have been re-architected from scratch for modern hardware. To do this, we use the

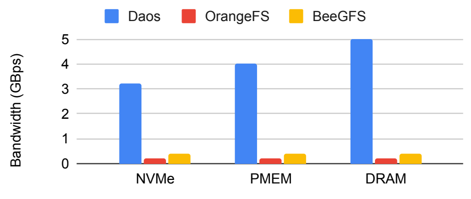
IO500 [38], which is a community benchmark designed to stress storage systems. We compare DAOS, OrangeFS [30], and BeeGFS [31] running over RAM, PMEM and NVMe, each using RDMA-capable networks. For OrangeFS and BeeGFS, we use EXT4 as the filesystem for interacting with storage (for PMEM, DAX is enabled). A stripe size of 64KB is used and data is distributed among the metadata servers in a round-robin fashion. Metadata and data servers are co-located. For DAOS, we use the SPDK for storing data on NVMe and DAX for storing data on PMEM. For the NVMe case, DAOS is configured with 50GB of PMEM and 5TB of NVMe; the majority of I/O will be to the NVMe instead of the PMEM. We run the IO500 for 5 minutes for each test. We measure I/O bandwidth and metadata throughput for the various workloads executed by the IO500. We also measure the theoretical bandwidth and throughput of the underlying hardware using the dd tool over the device file for the PMEM and NVMe devices per-node. The measurement of theoretical bandwidth does not account for network impacts. For bandwidth, we used a 1MB block size for dd. For throughput, we used 4KB as the block size. The sample configuration of DAOS uses a single I/O thread per-server, which is why single-threaded dd was used instead of a multi-threaded benchmark.

From Figure 3.1, it can be observed that DAOS provides performance benefits across the different benchmarks. In terms of bandwidth reported in the IO500-Easy experiment, DAOS outperforms OrangeFS by 10x over NVMe and by 15x over PMEM. IO500-Easy performs a workload which is optimal towards parallel filesystems such as OrangeFS, making large, sequential, and aligned I/O. However, although it is the best-case scenario for OrangeFS, DAOS's leaner I/O stack still provides significant performance improvements.

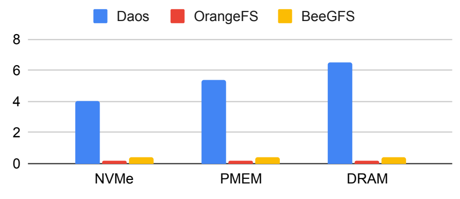
IO500-Hard performs a less optimal workload, which generates small and unaligned I/O. In this case, DAOS outperforms both BeeGFS and OrangeFS by 8x on NVMe and 10x over PMEM. There are two reasons for performance differences. First, IO500-Hard stresses metadata and small-I/O performance significantly more, which accrues significant overheads due to the kernel I/O stack. Second, BeeGFS and OrangeFS perform I/O in units of stripes (64KB). For I/O which is smaller than this and when boundaries are misaligned, an increased amount of I/O occurs.

For the mdtest-easy and mdtest-hard workloads, DAOS performs at least 18x faster than OrangeFS on both NVMe and PMEM. This is because DAOS uses a minimalistic I/O path for storing and querying metadata. Both BeeGFS and OrangeFS rely on the kernel's I/O stack. OrangeFS is a FUSE filesystem running atop EXT4, and BeeGFS is a kernel-level filesystem also running atop EXT4. While metadata queries don't (typically) go directly to disk, they must travel through multiple levels of software and network in order to complete. This leads to a long, expensive I/O path for every metadata access.

Overall, it can be observed that by re-architecting the storage stack from scratch for modern hardware, significant performance benefits can be observed for both bandwidth and latency-sensitive applications.



(a) VPIC



(b) BD-CATS

FIG. 3.2. *HPC workloads over different hardware and storage systems. DAOS improves performance significantly over other other storage system types.*

3.2. HPC. In this evaluation, we quantify the benefit of an optimized storage stack on typical checkpoint-restart workloads experienced in HPC. To do this, we run two common HPC workloads: VPIC [39] and BD-CATS [40]. VPIC is a particle simulation code where each process produces particle data and writes them at each time step. VPIC writes 8 million particles where each particle is a vector of 8 floating point values. We run VPIC for 16 time steps. BD-CATS reads the data generated by VPIC to perform a parallel clustering algorithm. Both VPIC and BD-CATS primarily perform large, aligned I/Os to HDF5 files.

From Figure 3.2, it can be observed that, for both VPIC and BD-CATS, DAOS is at least 6x faster than both OrangeFS and BeeGFS over NVMe and PMEM. This performance benefit is mainly due to the fact BeeGFS and OrangeFS have a significant metadata cost to keep track of all the 64KB stripes, which are distributed and queried among servers during reads and writes. Their lengthy, kernel-based I/O path causes significant software overhead on the metadata servers, which is where performance degradation arises. Overall, traditional HPC workloads can experience great performance benefits by using hardware-optimized storage stacks.

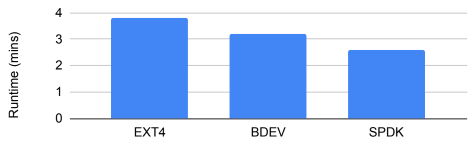


Fig. 3.3. *CosmicTagger for various storage and networking configurations of DAOS. The storage backend varies between SPDK, Linux block layer, and the EXT4 filesystem. SPDK performs up to 30% faster than others as the storage backend.*

3.3. Machine Learning. In this evaluation, we specifically quantify the extent to which different storage APIs impact I/O performance. To do this, we run CosmicTagger [41] through the DLIO [42] benchmark, which is a convolutional neural net for separating cosmic pixels, background pixels, and neutrino pixels. The training dataset contains 430,000 samples, where each sample contains three images of size 1280x2048. The overall dataset size is roughly 400GB. The samples are stored in an HDF5 dataset sparsely. At each iteration, 32 images are read from the dataset and preprocessed. Most I/Os are between 20 and 50KB in size. DAOS is configured to have 50GB of PMEM and 5TB of NVMe. We configure the NVMe to use either SPDK, Linux block layer, or the EXT4 filesystem as the storage backend.

From Figure 3.3, we see significant variations in overall performance due to the choice of local storage backend. When comparing different storage backends, it can be seen that SPDK performs 14% faster than the Linux kernel block layer and 30% faster than the EXT4 filesystem. Both kernel-level filesystems and the Linux block layer have been noted to impose several overheads [11], such as memory allocations, interrupts, and context switches. By using a leaner storage stack, significant performance benefits are observed in the distributed layer.

3.4. Future Work. One limitation of this work was the limited test environment. HPC systems typically have separate nodes for compute and storage. In a future work, we could explore the networking effects of having dedicated storage nodes and the effects of different network topologies. Another future work could be a deep profiling of the different storage systems to determine the exact areas of their code where performance was lost.

4. Conclusions. In this work, we quantified the value of re-architecting distributed storage systems to support the high-performance and low-latency of modern hardware. We performed a comprehensive benchmark of a state-of-the-art storage system (DAOS) and a traditional storage system (OrangeFS) using a variety of workloads over both persistent memory and NVMe. We demonstrated that traditional storage systems which do not optimize for modern hardware can experience performance degradation of as much as 15x under a variety of synthetic and HPC workloads. In addition, we also quantify the performance benefits caused by utilizing kernel-bypass storage and networking interfaces. We demonstrated that by using storage and networking interfaces which specifically optimize for modern storage hardware, performance improvements of up to 30% can be gained for real distributed workloads. Overall, re-architecting storage stacks for modern hardware was well worth the effort.

References.

[1] A. Kougkas, H. Devarajan, and X.-H. Sun, "Hermes: a heterogeneous-aware multi-tiered distributed i/o buffering system," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018, pp. 219–230.

[2] J. Cernuda, H. Devarajan, L. Logan, K. Bateman, N. Rajesh, J. Ye, A. Kougkas, and X.-H. Sun, "Hflow: A dynamic and elastic multi-layered i/o forwarder," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021, pp. 114–124.

[3] A. Kougkas, H. Devarajan, K. Bateman, J. Cernuda, N. Rajesh, and X.-H. Sun, "Chronolog: a distributed shared tiered log store with time-based data ordering," in *Proceedings of the 36th International Conference on Massive Storage Systems and Technology (MSST 2020)*, 2020.

[4] A. Kougkas, H. Devarajan, J. Lofstead, and X.-H. Sun, "Labios: A distributed label-based i/o system," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 13–24.

[5] H. Devarajan, A. Kougkas, L. Logan, and X.-H. Sun, "Hcompress: Hierarchical data compression for multi-tiered storage environments," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 557–566.

[6] S. Zheng, M. Hoseinzadeh, and S. Swanson, "Ziggurat: A tiered file system for {Non-Volatile} main memories and disks," in *17th USENIX Conference on File and Storage Technologies (FAST 19)*, 2019, pp. 207–219.

[7] S. Kannan, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, Y. Wang, J. Xu, and G. Palani, "Designing a true {Direct-Access} file system with {DevFS}," in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, 2018, pp. 241–256.

[8] L. Logan, J. Lofstead, S. Levy, P. Widener, X.-H. Sun, and A. Kougkas, "pmemcpy: a simple, lightweight, and portable i/o library for storing data in persistent memory," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021, pp. 664–670.

[9] G. Lee, S. Shin, W. Song, T. J. Ham, J. W. Lee, and J. Jeong, "Asynchronous i/o stack: A low-latency kernel i/o stack for ultra-low latency ssds," in *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, 2019, pp. 603–616.

[10] H.-J. Kim, Y.-S. Lee, and J.-S. Kim, "{NVMeDirect}: A user-space {I/O} framework for application-specific optimization on {NVMe}{SSDs}," in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[11] L. Logan, J. C. Garcia, J. Lofstead, X.-H. Sun, and A. Kougkas, "Labstor: A modular and extensible platform for developing high-performance, customized i/o stacks in userspace," in *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'22), November 14–17, 2022*. IEEE, 2022.

[12] J. Xu and S. Swanson, "{NOVA}: A log-structured file system for hybrid {Volatile/Non-volatile} main memories," in *14th USENIX Conference on File and Storage Technologies (FAST 16)*, 2016, pp. 323–338.

[13] R. Kadekodi, S. K. Lee, S. Kashyap, T. Kim, A. Kolli, and V. Chidambaram, "Splitfs: Reducing software overhead in file systems for persistent memory," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 494–508.

[14] I. Zhang, J. Liu, A. Austin, M. L. Roberts, and A. Badam, "I'm not dead yet! the role of

the operating system in a kernel-bypass era," in *Proceedings of the Workshop on Hot Topics in Operating Systems*, ser. HotOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 73–80. [Online]. Available: https://doi.org/10.1145/3317550.3321422

[15] Z. Yang, J. R. Harris, B. Walker, D. Verkamp, C. Liu, C. Chang, G. Cao, J. Stern, V. Verma, and L. E. Paul, "Spdk: A development kit to build high performance storage applications," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2017, pp. 154–161.

[16] G. Borges, S. Crosby, and L. Boland, "Cephfs: a new generation storage platform for australian high energy physics," in *Journal of Physics: Conference Series*, vol. 898, no. 6. IOP Publishing, 2017, p. 062015.

[17] A. Aghayev, S. Weil, M. Kuchnik, M. Nelson, G. R. Ganger, and G. Amvrosiadis, "File systems unfit as distributed storage backends: Lessons from 10 years of ceph evolution," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 353–369. [Online]. Available: https://doi.org/10.1145/3341301.3359656

[18] J. Lofstead, I. Jimenez, C. Maltzahn, Q. Koziol, J. Bent, and E. Barton, "Daos and friends: a proposal for an exascale storage system," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2016, pp. 585–596.

[19] S. Byna, Q. Koziol, V. Vishwanath, J. Soumagne, H. Tang, J. Mu, B. Dong, R. A. Warren, F. Tessier, T. Wang *et al.*, "Proactive data containers (pdc): An object-centric data store for large-scale computing systems," in *AGU Fall Meeting Abstracts*, vol. 2018, 2018, pp. IN34B–09.

[20] T. E. Anderson, M. Canini, J. Kim, D. Kostić, Y. Kwon, S. Peter, W. Reda, H. N. Schuh, and E. Witchel, "Assise: Performance and availability via client-local {NVM} in a distributed file system," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 1011–1027.

[21] Y. Kwon, H. Fingler, T. Hunt, S. Peter, E. Witchel, and T. Anderson, "Strata: A cross media file system," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 460–477.

[22] J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. Memaripour, Y. J. Soh, Z. Wang, Y. Xu, S. R. Dulloor *et al.*, "Basic performance measurements of the intel optane dc persistent memory module," *arXiv preprint arXiv:1903.05714*, 2019.

[23] Y. Lu, J. Shu, Y. Chen, and T. Li, "Octopus: an {RDMA-enabled} distributed persistent memory file system," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 773–785.

[24] M. Bjørling, J. Gonzalez, and P. Bonnet, "{LightNVM}: The linux {Open-Channel}{SSD} subsystem," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*, 2017, pp. 359–374.

[25] H. Zhu, *Data Plane Development Kit (DPDK): A Software Optimization Guide to the User Space-based Network Applications*. CRC Press, 2020.

[26] "Direct access for files," 2014. [Online]. Available: https://www.kernel.org/doc/Documentation/filesystems/dax.txt

[27] I. Zhang, A. Raybuck, P. Patel, K. Olynyk, J. Nelson, O. S. N. Leija, A. Martinez, J. Liu, A. K. Simpson, S. Jayakar, P. H. Penna, M. Demoulin, P. Choudhury, and A. Badam, "The demikernel datapath os architecture for microsecond-scale datacenter systems," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, ser. SOSP '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 195–211. [Online]. Available: https://doi.org/10.1145/3477132.3483569

[28] N. Moti, F. Schimmelpfennig, R. Salkhordeh, D. Klopp, T. Cortes, U. Rückert, and

A. Brinkmann, "Simurgh: a fully decentralized and secure nvmm user space file system," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–14.

[29] J. Liu, A. Rebello, Y. Dai, C. Ye, S. Kannan, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Scale and performance in a filesystem semi-microkernel," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, ser. SOSP '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 819–835. [Online]. Available: https://doi.org/10.1145/3477132.3483581

[30] M. M. D. Bonnie, B. Ligon, M. Marshall, W. Ligon, N. Mills, E. Q. S. Sampson, S. Yang, and B. Wilson, "Orangefs: Advancing pvfs," in *USENIX Conference on File and Storage Technologies (FAST)*, 2011.

[31] F. Chowdhury, Y. Zhu, T. Heer, S. Paredes, A. Moody, R. Goldstone, K. Mohror, and W. Yu, "I/o characterization and performance evaluation of beegfs for deep learning," in *Proceedings of the 48th International Conference on Parallel Processing*, 2019, pp. 1–10.

[32] "Ceph performance: Benchmark and optimization." Tech. Rep., 2018. [Online]. Available: https://croit.io/blog/ceph-performance-test-and-optimization

[33] Ceph, "Seastore," 2017. [Online]. Available: https://docs.ceph.com/en/quincy/dev/seastore/

[34] R. B. Ross, G. Amvrosiadis, P. Carns, C. D. Cranor, M. Dorier, K. Harms, G. Ganger, G. Gibson, S. K. Gutierrez, R. Latham *et al.*, "Mochi: Composing data services for high-performance computing environments," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 121–144, 2020.

[35] J. Lofstead, G. Markomanolis, J. Kunkel, and J. Bent, "Io500 sc19 lists," Nov 2019. [Online]. Available: https://doi.org/10.5281/zenodo.6462493

[36] S. Moss, "Intel kills off optane memory, writes off 559 million dollar inventory," 2022. [Online]. Available: https://www.datacenterdynamics.com/en/news/intel-kills-off-optane-memory-writes-off-559-million-inventory/

[37] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. Kanaujia, and P. Chauhan, "Tpp: Transparent page placement for cxl-enabled tiered memory," *arXiv preprint arXiv:2206.02878*, 2022.

[38] J. Kunkel, G. F. Lofstead, and J. Bent, "The virtual institute for i/o and the io-500." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2017.

[39] S. Byna, J. Chou, O. Rubel, H. Karimabadi, W. S. Daughter, V. Roytershteyn, E. W. Bethel, M. Howison, K.-J. Hsu, K.-W. Lin *et al.*, "Parallel i/o, analysis, and visualization of a trillion particle simulation," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE, 2012, pp. 1–12.

[40] M. M. A. Patwary, S. Byna, N. R. Satish, N. Sundaram, Z. Lukić, V. Roytershteyn, M. J. Anderson, Y. Yao, P. Dubey *et al.*, "Bd-cats: big data clustering at trillion particle scale," in *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 2015, pp. 1–12.

[41] C. Adams, M. Alrashed, R. An, J. Anthony, J. Asaadi, A. Ashkenazi, M. Auger, S. Balasubramanian, B. Baller, C. Barnes *et al.*, "Design and construction of the microboone cosmic ray tagger system," *Journal of instrumentation*, vol. 14, no. 04, p. P04004, 2019.

[42] H. Devarajan, H. Zheng, A. Kougkas, X.-H. Sun, and V. Vishwanath, "Dlio: A data-centric benchmark for scientific deep learning applications," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid).* IEEE, 2021, pp. 81–91.

LEVERAGING CONTINUATIONS TO INTEGRATE MPI WITH USER LEVEL THREADING

JOHN T. PARRISH[*], JAN CIESKO[†], AND STEPHEN L. OLIVIER[‡]

Abstract. Modern HPC systems rely on both inter-node and intra-node parallelism to achieve required performance. User-level threading (ULT) enables asynchronous intra-node parallelism, while the most typical model for inter-node parallelism is the Message Passing Interface (MPI). *MPI Continuations* is a proposed extension to MPI that would enable completion notification via callbacks as an alternative to direct polling. *MPI Continuations* is a uniquely appropriate model for the integration of the ULT task model and the MPI model because ULT models favor event-driven scheduling over blocking tasks. In this paper, we present an event-based model for non-blocking MPI Requests using the *MPI Continuations* extension, with implementations in both Sandia's Qthreads ULT runtime and OpenMP using OpenMP's tasking constructs. We also present an evaluation of these runtimes that suggests that lower level ULT libraries like Qthreads are less performant than higher-level tasking libraries when implementing task-based applications. Finally, we present ideas for how this work with MPI Continuations can be generalized to provide a middleware API that would allow tasking runtime developers to quickly and easily implement MPI support within their runtime using MPI Continuations.

1. Introduction.

User Level Threading (ULT) and Tasking are two related but distinct models of intra-node parallelism. In both paradigms, code is organized into lightweight units of work that can be easily swapped on and off one of possibly many kernel-level threads. Kernel-level are threads that are visible to the operating system [4]. Because kernel threads are created and managed within the kernel itself, there are significant overheads associated with thread creation and context switching. For this reason, applications written using kernel-level threads are often parallelized at a coarse granularity. To be performant, kernel-level threads should have enough work in them to persist for a long time and should not be frequently swapped in and out. This makes kernel-level threading a poor choice to execute fine-grained parallelism [3]. Many applications are logically organized into small tasks that would naturally be represented as threads, but it is often impractical to create so many kernel threads of potentially quite limited durations. This is where user-level threading can be beneficial [3, 4].

Tasking runtimes provide a convenient utility to represent node-local tasks in a dependency graph structure, but HPC systems are typically multi-node systems. The nodes in these systems are typically linked by a message-passing facility. The most typical framework used in these systems is the Message Passing Interface (MPI). Using MPI, one can efficiently parallelize applications across these nodes. In MPI, dependencies between nodes and processes are represented via messages. If node B requires a result from node A to proceed, then node B will wait on a message from node A containing said result. Using tasking, we can achieve single-node, multi-threaded DAG parallelism, and, using MPI, we can construct multi-node, single-threaded DAG parallelism. Issues arise, however, when we begin to consider how to implement multi-node, multi-threaded task DAGs using both MPI and local tasking runtimes. This is because using a blocking call (like `MPI_Wait`) will block the calling kernel thread without regard for any ULT runtime. Standard MPI is unaware of any user-level threading or tasking that may be utilized on each node, so it will block the kernel thread where the call was made. This is not ideal because blocked kernel threads are detrimental to the performance of ULT/tasking applications. Kernel threads in applications built with ULTs are analagous to CPUs in applications built directly on top of kernel threads, so blocking at the kernel-thread level paralyzes one of the execution units that

[*]Georgia Tech, jparrish34@gatech.edu
[†]Sandia National Laboratories, jciesko@sandia.gov
[‡]Sandia National Laboratories, solivi@sandia.gov

could normally context switch to a new task. In the worst case, blocking the kernel-level thread can lead to unexpected deadlocks [8]. This can happen when one user-level thread depends on another but no additional kernel-level thread is available to schedule the second ULT. If the blocking of the kernel-level thread had instead been replaced by yielding the user-level thread, the deadlock would be avoided.

Sandia National Laboratories maintains Qthreads, a ULT runtime with low-level synchronization primitives. In Qthreads, the main synchronization primitive is the full-empty bit. Users associate a full-empty bit with a memory location and use the bit to synchronize memory accesses. In this paper, we explore the MPI Continuations extension proposed by Schuchart *et al.* [8] can be used to integrate MPI communication with Qthreads and other ULT/tasking runtimes. We present our implementation of MPI support for OpenMP and Qthreads using MPI Continuations; our observations about the state of event support in Qthreads, Argobots, and OpenMP; a performance evaluation of OpenMP and Qthreads using a Gauss-Seidel heat equation solver benchmark; and our ideas for a portable tasking + MPI middleware library built on top of MPI Continuations.

2. Background. Because naively using MPI together with ULT/tasking can lead to performance and deadlock issues, there is a need for a system that will intelligently integrate the two paradigms. Previous solutions to this problem have modularized MPI's threading support to enable MPI to be compiled in a ULT-aware mode. Evans *et al.* [2] added support for the Qthreads and Argobots runtimes to Open MPI. This elevates the blocking from the level of kernel-level threads to the level of user-level threads, but the completion of an MPI request still requires some task to block itself and await the completion of the request by making a blocking call to the MPI library.

Schuhcart *et al.* [9] have proposed an extension to the MPI standard that provides callback-based completion notification called *MPI Continuations.* Using this extension, one can register a callback with the MPI runtime itself. Requests are associated with a callback and then grouped under a special request called a continuation request. By polling the continuation request, the traditional requests with the continuation request are also polled, thereby making progress. On the completion of one of the traditional requests managed by the continuation request, the MPI runtime calls the callback function associated with that request. Our approach uses this continuations API to integrate non-blocking MPI requests with ULT/tasking runtimes.

Tasking and ULT are closely related concepts, but they have important differences. Tasks are a higher-level of abstraction over user-level threads. While user-level threads are typically just the user-space analogue of traditional kernel-level threads, tasks are units of work which may have dependencies [8] on other tasks. Tasking runtimes are optimized to organize tasks by dependencies and perform important functions like temporal sequencing and task throttling. Temporal sequencing means that task schedules are generated based on when a task is created, so if task A and B both output some dependency D, and task C waits on that dependency, then task C will only wait on whichever of the two tasks A and B are created before C. So if the tasks are created in the order A, C, B, then task C will depend on task A but not task B. This is important because the Gauss-Seidel heat equation solver–the primary benchmark we used–relies heavily on this temporal sequencing to ensure correctness. Qthreads is a low-level ULT library that has some facilities for organizing tasks into a dependency structure, but it lacks the rich feature set that OpenMP provides through its task pragmas. Our primary benchmark, the Gauss-Seidel heat equation solver was adapted from an OpenMP-based implementation [5]. Upon porting the benchmark into Qthreads, we noticed significant overheads associated with the scheduling of the ULTs on each node. We believe that these results are a direct consequence of the semantic differences between OpenMP's higher level tasks and Qthreads' lower level ULTs. These observations are discussed further in Sections 6 and 7.

3. Implementation. We implemented MPI support using MPI Continuations in OpenMP, Qthreads, and Argobots. All of our implementations used the version of MPI Continuations found in Joseph Schuchart's fork of the Open MPI project at `https://github.com/devreal/ompi` [7]. We found OpenMP to be the easiest of the three runtimes we examined to integrate with MPI Continuations. In OpenMP, tasks are managed using task constructs, which were added in version 3.0 [11]. We implemented a simple OpenMP example which is shown in Figure 3.1. It consists of a task which initializes the `MPI_Irecv` request, a task which waits on the request to complete and then prints the result, and a task which polls the continuation request. The `MPI_Irecv` request is represented as an OpenMP event. The event is generated by the detach clause. This clause creates the event handle and causes the first task to await the completion of the event after running to completion. The output dependency of the first task will not be satisfied until it has run to completion and the event has been marked completed. The third task shown is the polling task which polls the continuation request. A continuation request with no attached work will test as completed, so the polling task waits to poll the continuation request until `comm_started_flag` is set to 1 to avoid terminating the polling loop prematurely. This example demonstrates the fundamental idea of our approach to integrating MPI Continuations with tasking: MPI requests are represented within the tasking runtime as events and upon completion of the MPI request, the event is marked as fulfilled.

We also joined MPI Continuations with the Argobots [10] and Qthreads [12] frameworks. Unlike OpenMP, Argobots and Qthreads are lower-level ULT frameworks, so they do not have the same support for high-level dependency-graph management as OpenMP. Instead, we had to create and maintain the dependency structures ourselves using Qthreads' full-empty bit semantics and Argobots' eventuals (which are like traditional futures). While we explored both Argobots and Qthreads as potential candidates to join with MPI Continuations, the results in this paper are limited to Qthreads, and MPI Continuations support in Argobots is left as future work.

The lack of proper tasking semantics in Qthreads proved to be a significant challenge when implementing more complex programs than the simple example in Figure 3.1 with more intricate dependency graphs. This became exceptionally problematic when implementing the Gauss-Seidel Heat Equation solver. The synchronization of tasks in Qthreads is dependent on full-empty bit semantics. These act somewhat like futures, with the associated bit being set to full when the data is ready to be used. This simpler system makes the sequencing of tasks less straightforward, and introduces overhead. In Qthreads, tasks are created and launched before they yield to wait on their dependencies to be satisfied, whereas OpenMP tasks are created but not launched until each dependency is satisfied. Qthreads does provide a way to precondition tasks on a collection of full-empty bits, but it doesn't provide a straightforward way to sequence tasks depending on a particular full-empty bit, as OpenMP does with its input/output dependency structure. To replicate this pattern in Qthreads, one either has to create an excess number of full-empty bits to model each dependency or accept that the precondition API cannot be used and perform manual synchronization with the full-empty bits. In our implementation, we chose to forgo the use of the precondition API in favor of keeping the number of full-empty bits lower. Our investigation suggests that in order to properly represent task-based parallelism, task-specific constructs should be provided. A low-level ULT model like Qthreads is not enough to efficiently represent large task-graph structures.

```
  // this task runs first to start the MPI request and detaches to wait for the Irecv to complete
  #pragma omp task depend(out:value) shared(value, comm_started_flag) detach(event)
  {
    // the receive request
    MPI_Request req;
    // start the Irecv
    MPI_Irecv(&value, 1, MPI_INT, 1, 425, MPI_COMM_WORLD, &req);
    // pass the responsibility of the Irecv to the continuation request cont_req
    MPIX_Continue(&req, &release_event, (void *) event, MPI_STATUS_IGNORE, cont_req);
    // let the polling task know that the communication has started
    comm_started_flag = 1;
  }

  // this task can only run once the data is received from the other rank
  #pragma omp task depend(in:value) shared(value)
  {
    printf("RECEIVED %d\n", value);
  }

  // this task runs in parallel to ensure MPI progress
  #pragma omp task shared(comm_started_flag)
  {
    // flag will be set to 1 when the request has completed
    int flag = 0;

    do {
      // only test the request if a communication request has already been attached
      if (comm_started_flag) {
        // poll the communication request and check if the attached request is complete
        MPI_Test(&cont_req, &flag, MPI_STATUS_IGNORE);
      }
      // yield to another task
      #pragma omp taskyield
    } while(!flag);
  }

////////////////////////////////////////////////

// This is the callback that the MPI implementation calls when the MPI_Irecv is ready to complete
void release_event(MPI_Status *status, void *data) {
  // cast the passed pointer to the event handle
  omp_event_handle_t event = (omp_event_handle_t)(uintptr_t) data;
  // fulfill the event
  omp_fulfill_event(event);
}
```

Fig. 3.1: A simple example of completion notification using callbacks

4. Related Work. Task Aware MPI (TAMPI) [6] is a library that integrates OpenMP-style tasking and MPI. TAMPI has been our primary point of comparison, as the heat benchmark we have been using comes from an example implementation using TAMPI [5]. TAMPI tracks MPI requests as tickets. Each ticket corresponds to a request. The ticket keeps track of which task the request belongs to by saving a reference that task's event counter to be decremented later. To attach a request to a task, TAMPI adds a function called `TAMPI_Iwait`. When a request is attached to a task, that task's event counter is incremented, and when the request is completed, the event counter is decremented. A task cannot proceed until its task block has completed execution and its event counter is 0. The primary advantage of our approach over TAMPI is that our approach tracks the requests in the MPI runtime itself, using the built-in events from OpenMP tasking or the full-empty bits from Qthreads. This approach leads to a greater degree of flexibility and extensibility.

The pattern we use to join tasking constructs with MPI Continuations is general enough to be applicable to other ULT/tasking runtimes. As long as the runtime provides some form of event construct, relatively minimal code is required to incorporate MPI using MPI Continuations.

Evans *et al.* [2] present an implementation of flexible threading support for Open MPI. Low level functionality in Open MPI is managed through the *Open, Portable Access Layer* (OPAL) and is customizable via Open MPI's Modular Component Architecture (MCA). Evans *et al.* designed an OPAL MCA component architecture to allow the customization of threading support within MPI by specifying which threading MCA component should be used at compile time. Using this model, the authors implemented support for Pthreads, Qthreads, and Argobots. When compiled with Qthreads or Argobots support, Open MPI's blocking calls will perform proper user-level blocking, rather than blocking kernel threads. Our proposal differs from the approach used by Evans *et al.*, in that we support non-blocking MPI calls by integrating MPI requests as a form of event that tasks can depend on. The modular threading approach allows the tasking run-time to wait on MPI Requests implicitly by waiting for a task that is blocked by a blocking MPI call to unblock. Our approach, however, supports tasks depending on MPI Requests directly. No task is required to block and wait for completion of an MPI request. Rather, the tasking runtime itself is made aware of the request as an event, allowing for the runtime to be flexible in how it handles dependencies on an MPI request compared to dependencies on another task.

Chatterjee *et al.* [1] present a method of supporting MPI calls within the Habanero-C language, which they call *HCMPI.* The authors propose the concept of a *Distributed Data-Driven Future* which extends Habanero-C's concept of a *Data-Driven Future* to multiple nodes. These allow users to perform inter-node synchronization using the semantics of futures. These futures support put, get, and await operations. *HCMPI* ensures progress in the MPI runtime by dedicating one of the kernel-level worker threads that the runtime spawns entirely to polling into the MPI runtime. *HCMPI* shares many conceptual similarities with our approach. HCMPI's DDDF construct is similar to an event in our implementation, and HCMPI guarantees progress similarly by dedicating a kernel-level thread to polling. Rather than specifying a request as a future, we handle them as events, which can be awaited but don't conceptually support get and put operations. Our implementation also encapsulates polling as a task within the runtime rather than as a separate entity which runs inside a different kernel-level thread. The primary difference between our approach and *HCMPI* is that our approach, while similar in concept to *HCMPI*, attempts to be more general. Our general implementation scheme assumes only a minimal set of capabilities of the underlying ULT/tasking runtime compared to *HCMPI*, which is built specifically for the Habanero-C language.

5. Important Observations. Our study of both simple examples and the heat benchmark revealed that the key advantage of using MPI Continuations for this task is that it allows one to link ULT/tasking runtimes with MPI requests using a minimal set of adaptor functions. While porting TAMPI to a new runtime would likely require a port of the ticket system and the event counters, potentially requiring modifications to the runtime itself, enabling MPI support via MPI Continuations only requires some form of event, the ability to wait on events, an event fulfillment function, and a method of polling the continuation request. The first two are present natively in all three of the runtimes we investigated. OpenMP has the detach clause and the `omp_event_handle_t` type. Qthreads has the full-empty bit semantics, and Argobots has its eventuals. Assuming the presence of these first two features, all that is required to port this continuations-based support into a new runtime is to write a callback function which satisfies an event and to ensure that the continuation request is consistently polled. In our implementations, this last requirement took the form of a task in charge of polling that was usually untied so that it could freely be run on any worker/core that became available. In our benchmarks and examples, the polling task was explicitly declared alongside the application code. The polling task does not need to be changed between applications, however, so ULT/tasking library writers could easily include a way to run this polling task automatically.

6. Evaluation. We collected our evaluation data using the Blake cluster at Sandia National Laboratories. Blake is a 40 node Intel Xeon Platinum (Skylake) 8160, 2.1GHz - based cluster. It has two sockets per node and 24 cores per socket connected over the Intel OmniPath interconnect.

To benchmark the various ULT/tasking + MPI Continuations implementations we converted a version of the Gauss-Seidel heat equation solver originally written for TAMPI into OpenMP + MPI Continuations and Qthreads + MPI Continuations versions. We ran these versions in addition to the TAMPI version of the code and a blocking version which used kernel-thread blocking MPI calls within the communication tasks. We ran each version on 1, 2, 4, and 8 nodes (Figures 6.1 - 6.4) using one rank per node and tested grid dimensions of 8192×8192, 16384×16384, 24576×24576, 32768×32768, and 40960×40960. Note that the grid dimension shown in the figures represents one of the dimensions of the square grid, so a data point at having a grid dimension of 8192 was run on an 8192×8192 grid. We ran each configuration for 300 timesteps. We attempted to run each configuration 3 times, but due to time constraints, we were unable to complete all 3 runs on certain configurations. Each running time data point collected is plotted on the graphs, and the lines plotted for each configuration interpolate between the means of the running times for each grid dimension.

As shown in Figure 6.1, we ran each version on 1 MPI node to get a baseline performance. We were unable to obtain data on the Qthreads implementation for a single node due to time constraints, but based on other benchmarks, it seems reasonable to expect that Qthreads incurs significant overhead compared to the OpenMP models. The three OpenMP versions all performed almost identically. This result matches expectation because the only differences between the three versions of the code lies in the communication tasks, of which there should be none on a single node.

As shown in Figures 6.2 – 6.4, Qthreads shows significant overhead compared even to the blocking communication version on all configurations. We believe this overhead results from the way we sequenced the Qthreads using full-empty bits. We created an $n \times n \times 5$ grid where each grid square has 5 full-empty bits corresponding to each of the north, south, east, and west neighbors and the cell itself. Each task awaits these 5 full-empty bits, and because Qthreads' precondition routine does not mark the bits empty after the condition is satisfied, we were unable to properly precondition the tasks using the proper API. Instead, we had to begin each task and then perform a wait on each of the full-empty bits, clearing them immediately after each wait with the *qthread_readFE* function. This was necessary to ensure that only one update was executed on a cell per iteration. Because we were unable to use the precondition API, each of the $n \times n \times t$ tasks has to be spawned and then begin waiting. We believe this creates a significant overhead that scales with the size of the grid, causing Qthreads to perform the worst of all the implementations tested.

Figure 6.4 shows high variability of TAMPI's performance on 8 nodes. We measure the variability of the performance by calculating the range of the 3 running times observed for a given configuration. The average range for TAMPI on 8 nodes was 152.92 seconds compared to 0.64 seconds on 2 nodes and 1.25 seconds on 4 nodes. For additional comparison, the average range on 8 nodes of both the OpenMP and blocking versions combined was 9.38 seconds. While the data we have collected is not able to rule out some form of experimental

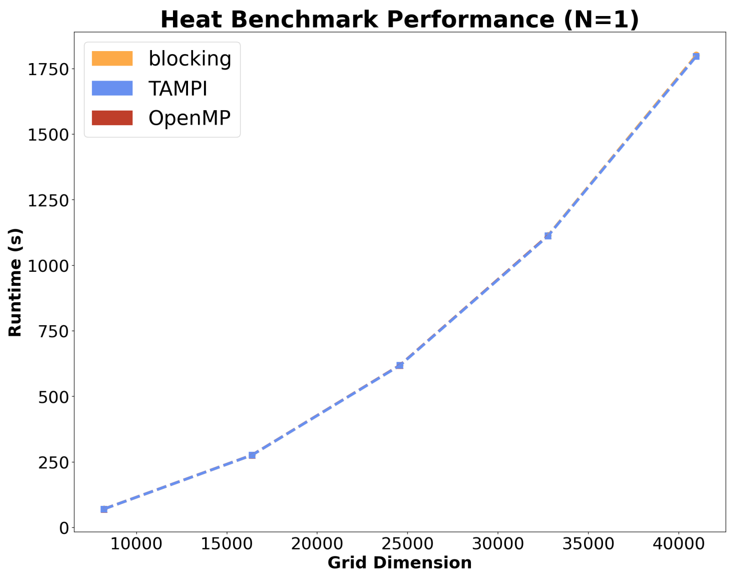


Fig. 6.1: Gauss-Seidel heat equation solver performance results on 1 node

error resulting from the status of the Blake cluster at the time the results were collected, it is interesting that the increased variability is only observed in the TAMPI results. Further experiments are required to determine what caused this increased variability.

The OpenMP + Continuations implementation performs on par with the TAMPI version in every configuration tested, as shown in Figures 6.1 – 6.4. These results show that MPI Continuations is efficient enough to stand in for the ticket system employed by TAMPI, validating the idea that MPI Continuations could form the basis for a generalized framework which can be adapted with relatively low overhead into any runtime with adequate support for tasking and events. The significantly higher runtimes of the Qthreads implementation, however, suggest that libraries with lower-level ULT semantics may need the addition of a dedicated tasking layer to exhibit the performance achieved in OpenMP + Qthreads.

7. Conclusions and Future Work.

The development and evaluation of the various ULT + MPI Continuations implementations yielded several important insights. First, it proved that MPI Continuations is a viable alternative to the runtime-specific style of framework that is present in the TAMPI API. The speed of the OpenMP + MPI Continuations heat implementations suggests that no performance is lost by using the facility of continuation requests rather than TAMPI's ticket system which depends on features specific to the OmpSs runtime [6].

Second, it showed that, with MPI Continuations, the amount of infrastructure required from the ULT/tasking side is minimal and consistent. Instead of building an analogous ticket system in each runtime, developers can pass the management of requests off to the MPI implementation itself. Then all that ULT/tasking runtime developers will need to add is the simple callback function and some sort of polling routine for their runtime. This is possible as long as the runtime supports some concept of events and preconditioning or blocking ULTs/tasks for event completion.

The sub-optimal Qthreads results demonstrated that the synchronization and dependency semantics in Qthreads are not complete enough to easily emulate the tasking semantics

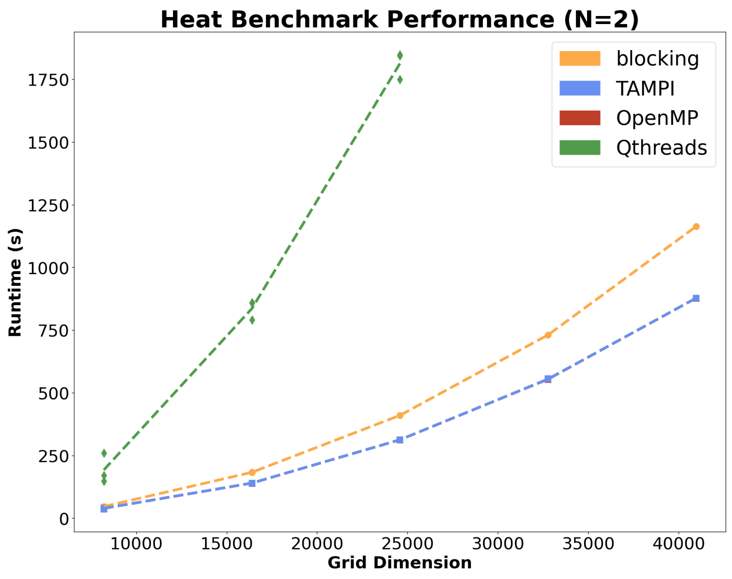


Fig. 6.2: Gauss-Seidel heat equation solver performance results on 2 nodes

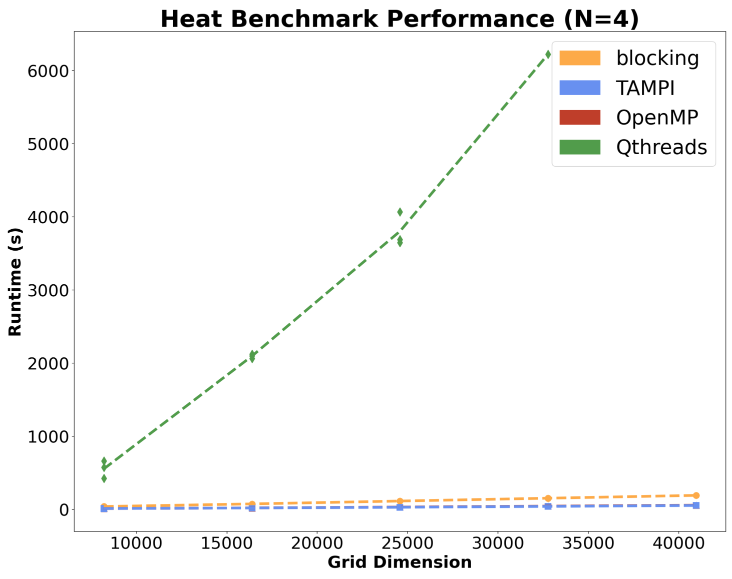


Fig. 6.3: Gauss-Seidel heat equation solver performance results on 4 nodes

of higher-level runtimes like OpenMP. All of these insights have revealed opportunities for future work with both MPI Continuations and Qthreads. The semantics required to connect a tasking runtime to MPI Continuations are minimal and consistent across runtimes. This suggests that a middleware library could be developed to systematize and simplify the integration of MPI Continuations into tasking runtimes. Runtime developers could specify functions that create events, fulfill events, and start the polling task. Then with these mini-

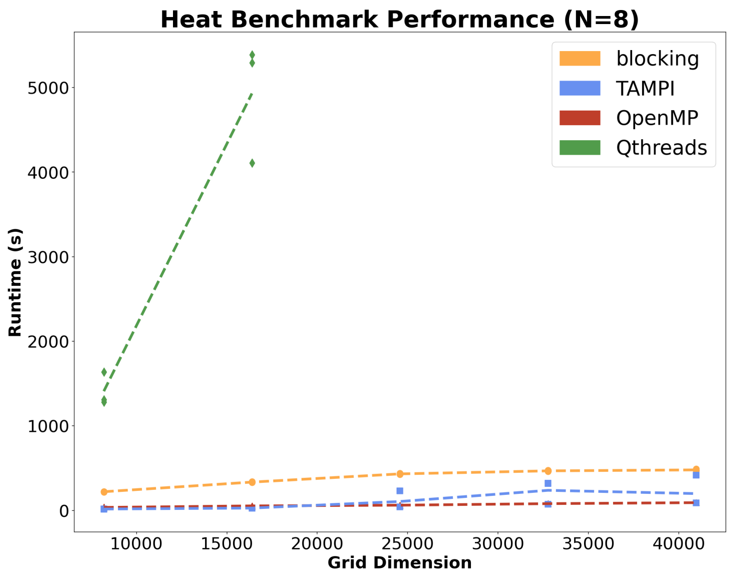


Fig. 6.4: Gauss-Seidel heat equation solver performance results on 8 nodes

mal features specified, the runtime could have access to a library of functions that streamline the management of MPI requests using MPI Continuations.

The fact that Qthreads struggles to match the performance of OpenMP tasks in the heat benchmark suggests that there is a performance difference associated with the semantic difference between ULTs and tasks noted by Schuchart *et al.* [8]. While there may be better performing ways to structure the same benchmark using Qthreads, there is an appreciable gap in the performance of OpenMP and that of Qthreads when it comes to implementing task-based semantics. The difference in overhead and ease of use between Qthreads and OpenMP for the task-based heat equation solver suggests that it may be useful to add tasking semantics on top of lower-level ULT runtimes like Qthreads and Argobots.

REFERENCES

[1] S. Chatterjee, S. Tasirlar, Z. Budimlic, V. Cave, M. Chabbi, M. Grossman, V. Sarkar, and Y. Yan, *Integrating asynchronous task parallelism with mpi*, in 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, IEEE, 2013, pp. 712–725.
[2] N. Evans, J. Ciesko, S. L. Olivier, H. Pritchard, S. Iwasaki, K. Raffenetti, and P. Balaji, *Implementing flexible threading support in open mpi*, in 2020 Workshop on Exascale MPI (ExaMPI), Nov 2020, p. 21–30.
[3] S. Iwasaki, A. Amer, K. Taura, and P. Balaji, *Analyzing the performance trade-off in implementing user-level threads*, IEEE Transactions on Parallel and Distributed Systems, 31 (2020), pp. 1859–1877.
[4] B. D. Marsh, M. L. Scott, T. J. LeBlanc, and E. P. Markatos, *First-class user-level threads*, SIGOPS Oper. Syst. Rev., 25 (1991), p. 110–121.
[5] R. L. Rodríguez, *Acm-bsc-contest.*
[6] K. Sala, X. Teruel, J. M. Perez, A. J. Peña, V. Beltran, and J. Labarta, *Integrating blocking and non-blocking mpi primitives with task-based programming models*, Parallel Computing, 85 (2019), p. 153–166.
[7] J. Schuchart, *Openmpi continuations.* `https://github.com/devreal/ompi`, 2022.
[8] J. Schuchart, C. Niethammer, and J. Gracia, *Fibers are not (p) threads: The case for loose coupling of asynchronous programming models and mpi through continuations*, in 27th European

MPI Users' Group Meeting, 2020, pp. 39–50.

[9] J. Schuchart, P. Samfass, C. Niethammer, J. Gracia, and G. Bosilca, *Callback-based completion notification using mpi continuations*, Parallel Computing, 106 (2021), p. 102793.

[10] S. Seo, A. Amer, P. Balaji, C. Bordage, G. Bosilca, A. Brooks, P. Carns, A. Castelló, D. Genet, T. Herault, S. Iwasaki, P. Jindal, L. V. Kalé, S. Krishnamoorthy, J. Lifflander, H. Lu, E. Meneses, M. Snir, Y. Sun, K. Taura, and P. Beckman, *Argobots: A lightweight low-level threading and tasking framework*, IEEE Transactions on Parallel and Distributed Systems, 29 (2018), p. 512–526.

[11] R. Van der Pas, *Openmp tasking explained*, online: http://openmp. org/wpcontent/uploads/sc13. tasking. ruud. pdf, (2013).

[12] K. B. Wheeler, R. C. Murphy, and D. Thain, *Qthreads: An api for programming with millions of lightweight threads*, in 2008 IEEE International Symposium on Parallel and Distributed Processing, Apr 2008, p. 1–8.

IMPROVING A SOFTWARE USER INTERFACE WITHOUT THE USER

ABIGAIL SPIGARELLI* AND JACOB DAVIS†

Abstract.
This project focuses on upgrading operators' experience using a satellite scheduling software. Improving its User Interface (UI) will reduce the cognitive load and mistakes made, increase confidence in the system and understanding of the information displayed, and lastly enhance the onboarding experience. Many UI evaluations rely on user feedback to improve. Unfortunately, this project doesn't have access to its users. Instead, the UI is evaluated for areas of improvement using cognitive models. Initial experimentation used CogTool (a human predictive performance model), and its accuracy was verified with manual testing. CogTool measured the cognitive load and timed navigation of the UI. These experiments showed that applying principles from the metrics reduces time spent on urgent tasks. Future experiments with other models and tools will need to be performed to measure the quality of the UI.

1. Introduction. Improving the User Interface (UI) is a key aspect of software updates. Usable systems provide "a number of benefits including improved productivity, enhanced user well-being, avoidance of stress, increased accessibility and reduced risk of harm"[5]. As a legacy satellite scheduling software is being revamped, improving the UI is therefore an important concern. A software interface evaluation is dependent on specific restrictions and applications of the project. This satellite scheduling software consists of mostly classified work, limiting access to its operators. The software also handles a large breadth of work, performing numerous complex operations. Lastly, the software has a small number of users, who use the software everyday for years. An effective UI evaluation will be appraised by its ability to decrease the time required for critical tasks and overcome these three challenges: access to its users, its complexity, and its long term effect.

The access to the software's operators presents a difficult challenge. The software is used by a small group of operators that, because of the classified nature of the project, have limited interaction with the software developers. It is possible to receive feedback on design mockups, but this is conducted through an intermediary and takes many days to receive a response. A well researched and common approach to improving the UI is a user based evaluation. Thought to be the bread and butter of UI evaluations, they require access to the users through short surveys, formal interviews and information collected as they interact with the software. Without direct access to the users, a user evaluation based approach is next to impossible for this project.

An alternative solution is to build models and systems to synthetically approximate the user experience. This technique faces a significant roadblock: the complexity of the software. This is a very intricate piece of software with many different sub-applications and controls. The interface is entirely digital, giving each operator a considerable amount of flexibility with their set up. Additionally, there are a variety of roles individual operators perform adding another unknown variable. For example, a standard operator typically utilizes two monitors, but "super" operators use around eight monitors to manage all the information and tasks they need to perform. The variability in the user's setup makes building precise and accurate synthetic models difficult. One technique to overcoming this complexity, is to break down each of the components of the software into concrete and distinct tasks. By increasing the levels of abstraction, most unknown variables are eliminated, contributing to an insightful model.

The customization of the operator's environment is intentional. The operators asked for this feature, and it was provided. In fact, the software is built at the request of its

*Brigham Young University, as2273@byu.edu
†Sandia National Laboratories, jacdavi@sandia.gov

operators. Contrary to typical commercial software, the funding for this software comes not at the expense of the users, but on behalf of its users. There is no concern about customer retention, encouraging the software to overlook short-term aspects, such as it's learning curve, and focus on the expert user's experience. Unfortunately, because internet economy incentives favor short-term software, the long-term effect of a UI is still largely unstudied making it difficult to improve associated interfaces[9].

In addition to overcoming those three challenges, an effective UI evaluation will also need to identify areas where an interface will improve the most. In order to narrow the evaluation's scope, the chosen UI analysis will be measured by its impact in task time. Certain applications of the satellite scheduling software need to handle very time-critical tasks. The chosen UI evaluation will need to address and fulfill this goal of increased time efficiency.

Since most UI evaluations focus on user's feedback and their environment, the lack of interaction with the software users presents the most difficult challenge. Even if there was user feedback, the complexity of the software prevents a comprehensive evaluation. Lastly, the lack of research on the long-term aspects of the UI creates a third roadblock. The satellite scheduling software requires a UI evaluation that needs minimal user feedback, handles the software's complexity, takes into perspective the long term impact, and decreases the time spent on critical tasks.

2. Finding an Appropriate Evaluation. After identifying the challenges, the following section will compare four UI evaluation methods (cognitive models, usability testing, system usability score and a heuristic evaluation) in their ability to improve the time estimates for tasks. Two of the evaluations require user feedback and two provide quantitive results. A method that best addresses the three previously stated challenges and provides time evaluations of tasks will be chosen.

As stated before, although most heavily researched, a user evaluation based approach is not possible for this project. This restriction eliminates usability testing and system usability scores, leaving cognitive modeling and heuristic evaluations. A heuristic based evaluation, requires training to perform and most importantly doesn't measure or officially address improving the time efficiency of particular tasks. The last method, cognitive modeling requires little user involvement, provides time based estimations of the tasks, and simplifies complex interfaces. Cognitive modeling doesn't necessarily focus on expert users, but particular applications of it does. By addressing all three challenges and providing time estimates of the tasks, cognitive modeling is the best evaluation for this project. Table 2.1 summarizes the differences between the methods.

3. Congitive Modeling. A comprehensive analysis of all available cognitive models, their underlying theories, and possible applications to this project is beyond the scope of this paper. However, there is a methodology that address the complexity of the software and provides a long-term approach to this analysis: KLM (Keystroke-Level Model)[2]. KLM is the simplest GOMS (Goals, Operators, Methods, and Selection)[10] technique. Defined over 40 years ago, GOMS provides a top-down approach to decomposing a task, starting at the user's goal and breaking the goal down into sub-goals. These goals are then achieved by applying methods and selection rules. Because of its utilization of abstraction, this cognitive model easily addresses the complexity of the operator's environment. KLM also aims to predict "the time it takes an expert user to preform a given task on a given computer system"[7]. By focusing on expert users, KLM strengthens the cognitive model's approach to improving long-term focused software. KLM is a more restrictive GOMS technique, "with no parallel activities, no interruptions, and no interleaving of goals"[6]. This requires that only simplified tasks are modeled. For this application of KLM, this restriction is an

	Strengths	Weaknesses
Cognitive Modeling	-Low cost -No user involvement -Provides time estimates for tasks	-Might require training -Requires simple tasks
Usability Testing	-User feedback is often exceptionally helpful -Can provide time estimates for tasks -Well researched approach	-Requires contact with users -High cost
System Usability Score	-Provides quantitive results for qualitative aspects of the design	-Requires contact with users -Subjective to evaluator
Heuristic Evaluation	-No user involvement -Can be combined with other evaluations	-Requires trained evaluator -High cost -Subjective to evaluator -Evaluation might find only minor issues -No quantitative results

Table 2.1

asset, helping address the issue of the software's complexity. An additional challenge is the complexity of the model. This typically contributes to time intensive computations and formal training to implement and understand. This problem is overcome with Cogulator and CogTool, two easy to use open source tools that use KLM methods to evaluate interfaces, requiring only informal training to use. Following an analysis of their differences, applying the chosen tool in the context of the software will be discussed.

3.1. Cogulator and CogTool. Cogulator[3] and CogTool[1], were tested and compared against each other in the context of this project. Cogulator provides significantly more modeling flexibility, allowing both physical and digital components. This contributes to a steeper learning curve. Since the satellite scheduling software is entirely digital with no physical aspects, Cogulator's emphasis on comprehensive modeling isn't necessary. On the other hand, CogTool is built to analyze screenshots, a perfect application for the entirely digital UI. In addition, CogTool is easy to use, produces interesting visualizations of the model, and encourages comparing UI mockups. For these reasons CogTool was chosen to create the cognitive models for this UI evaluation. Table 3.1 summarizes the differences between CogTool and Cogulator.

	Cogulator	CogTool
Strengths	-Model flexibility	-Efficient and fast model building -Computes a visualization of the model -Provides ranges for predicted task times -Easy exploration of alternative UI designs
Limitations	-Steeper learning curve -Tedious to create models	-Requires a screenshot of the UI -Inflexible with order of specific tasks -No feature to model specific cognitive tasks, such as mental arithmetic or memorization

Table 3.1

3.2. Research. With CogTool as the chosen cognitive modeling tool, the next step is using it to research the software's interface. As stated earlier CogTool is based on the KLM method, so the first step is identifying a goal or task. Initially four particularly time-sensitive, but simple tasks were chosen. Next, screenshots of the associated UI were imported, each signifying a frame or a state of the device. Widgets or "hot spots" were then highlighted. These represent the interactive controls of the UI and connect each of the frames together creating a storyboard that outlines the steps to conduct a task. To preserve the sensitive nature of the project, a completed storyboard from a different project is shown below.

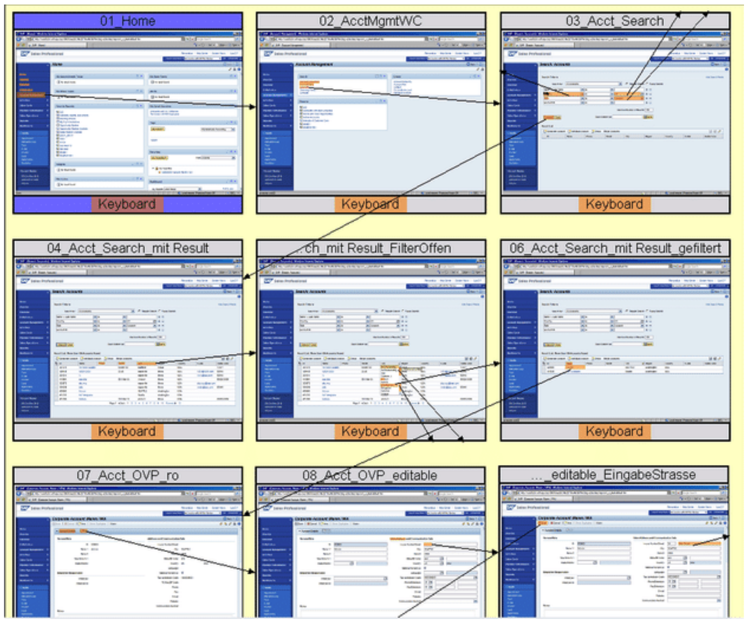

01_Home
02_AcctMgmtWC
03_Acct_Search
04_Acct_Search_mit Result
...ch_mit Result_FilterOffen
06_Acct_Search_mit Result_gefiltert
07_Acct_OVP_ro
08_Acct_OVP_editable
..._editable_EingabeStrasse
Keyboard

FIG. 3.1. *High-level example of a CogTool Storyboard[8].*

After creating the story boards, choosing what would make the most impactful changes to the UI was difficult. Analyzing a UI is often subjective and dependent on the evaluator. One way to avoid this is to use established metrics. One study researched the impact of this approach by creating a web page based evaluator that used a multi-metric solution relying on a corpus of different formulas[11].Their findings show that general principles and guidelines, if followed correctly, can create an effective and systematic UI evaluation. In light of their research, we decided to improve the UI designs according to Fitts's law, a formula based metric. Defined in 1954 by Paul Fitts, this law models human movement. It proves that the time it takes to click an area on a screen is a function of the distance to the target divided by the size of the target[4]. This metric encourages a simple UI with large buttons and important features clustered together.

None of the changes made to the improved UIs were drastic, typically involving only a size-able increase in a particular button. The tasks were already so simplified that there were limited areas of improvement. Figure 3.2 shows CogTool's output comparing the change between the legacy and the improved UI. The task modeled required three clicks. The difference in the tasks time between the two mockup versions is minimal. Figure 3.3 shows the difference in time between the legacy and mockup for three other tasks, revealing similar results.

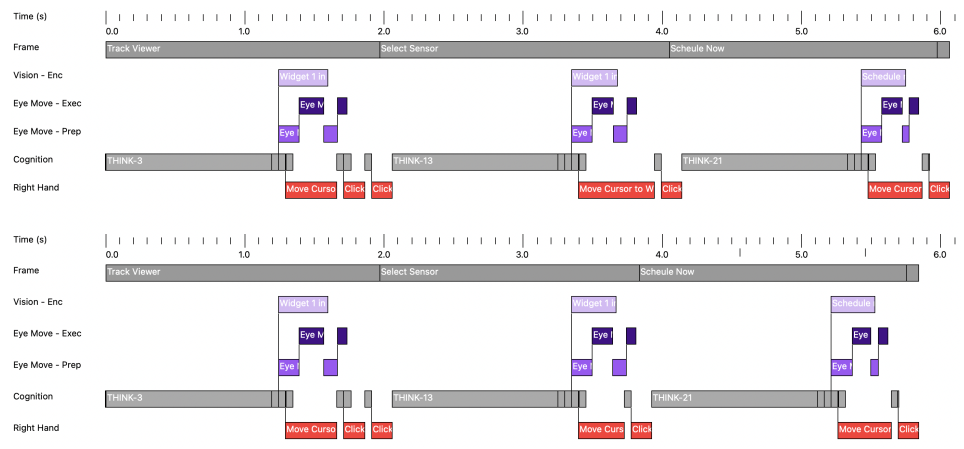


FIG. 3.2. *Comparison between the CogTool timelines for the legacy (top) and improved(bottom) UIs.*

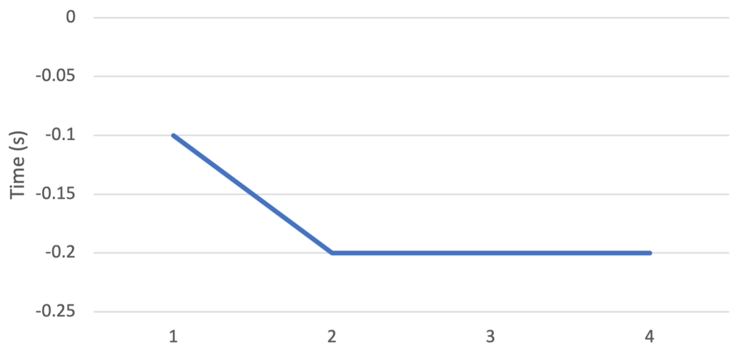


FIG. 3.3. *Difference in time for legacy and improved designs for 4 specific tasks.*

CogTool is a useful cognitive modeling tool, particularly because it provides time estimates for its tasks. While, the time estimates were insightful, they were minimal. The complexity of the software, and the model's required simplification of the tasks, meant that each of the tasks were taken largely out of context with the rest of the software. How often these tasks are conducted in relation to the rest of the possible tasks, could increase the significance of the otherwise minimal improvements. Additionally, to test the accuracy of the time estimates, one author manually timed how long it took to perform the tasks. One task that CogTool estimated would take 13.8 seconds, only took 10.9s to complete on average. Granted, CogTool estimations do include time for cognitive processing, which is difficult to test easily. However, that is a 26% percent difference between the modeled and actual performance times. Lastly the small improvements in terms of time efficiency came at the cost of aesthetics. The larger buttons encouraged by Fitts' law were less visually appealing to multiple developers than the legacy interface. The required simplification of the software, manual testing results, and balance between aesthetics and time efficiency, may imply that the disconnect between reality and cognitive modeling is too great to find any statistically helpful results.

4. Future Research. While the results from our research were inconclusive, analyzing the UI with CogTool is only one of many other tools or approaches that can conquer the three challenges of the project. Addressing different goals of the software and alternative methods of data collection could results in significantly more promising results. Time efficiency isn't the only thing the interface can be measured by. For example, some satellite scheduling tasks favor accuracy over speed. There also might be ways to perform user evaluations with limited impact on the users. Information such as how often operators make mistakes, utilize UI shortcuts, use certain features, click particular buttons etc., could be collected. This data would make future software improvements easier to improve, broadening the horizons of possible areas of analysis. By focusing on different goals and methods of data collection, the research will be more conclusive and informative.

5. Conclusion. The purpose of this research is to show that even with minimal user contact, it is possible to analyze and improve a software's UI. The first step of improving a UI is analyzing the interface and identifying areas of improvement. Measuring the quality of the UI requires a tailored approach. In the context of a satellite scheduling software, this assessment was defined by three challenges: limited user feedback, intricacy of the software, and the long-term focus of the UI. The success of the approach was measured by its ability to improve the time to complete critical tasks. A KLM modeling software, called CogTool, was identified as the chosen approach. CogTool created an evaluation without access to users, responded well to simplified tasks, and measured all task time estimates from the perspective of an expert user. Unfortunately, initial research with CogTool failed to show any significant improvements to the interface. This research used only one solution, CogTool, and one metric, Fitts' law. As stated earlier there are other solutions that could possibly yield even better quantitative results. Following an accurate assessment of the interface, the next step is to implement the changes and improve the UI. It is expected that findings from this and future research will be incorporated into the software design.

REFERENCES

[1] *Cogtool*. `https://www.cogtool.org`. Accessed: 2022-08-12.
[2] S. K. Card, T. P. Moran, and A. Newell, *The keystroke-level model for user performance time with interactive systems*, Commun. ACM, 23 (1980), p. 396–410.
[3] M. Corp, *Cogulator*.
[4] P. M. Fitts, *The information capacity of the human motor system in controlling the amplitude of movement*, Experimental Psychology, 47 (1954), pp. 381–391.
[5] *Ergonomics of human system interaction-part 210: Human-Centered Design for Interactive Systems*, standard, International Organization for Standardization, Switzerland, Jun 2019.
[6] B. E. John and D. E. Kieras, *The goms family of user interface analysis techniques: comparison and contrast*, ACM Trans. Comput. Hum. Interact., 3 (1996), pp. 320–351.
[7] B. E. John, K. Prevas, D. D. Salvucci, and K. Koedinger, *Predictive human performance modeling made easy*, CHI '04, New York, NY, USA, 2004, Association for Computing Machinery, p. 455–462.
[8] C. Kovesdi and J. Joe, *Exploring the use of cognitive models for nuclear power plant human-system interface evaluation*, Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 63 (2019), pp. 2190–2194.
[9] S. Kujala, V. Roto, K. Väänänen-Vainio-Mattila, E. Karapanos, and A. Sinnelä, *Ux curve: A method for evaluating long-term user experience*, Interacting with Computers, 23 (2011), pp. 473–483.
[10] A. N. Stuart K. Card, Thomas P. Moran, *The Psychology of Human-Computer Interaction*, CRC Press, blank, 1983.
[11] M. Zen and J. Vanderdonckt, *Towards an evaluation of graphical user interfaces aesthetics based on metrics*, in - 2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014, pp. 1–12.

REFORMULATING OPTIMIZATION-DRIVEN FACILITY-SCHEDULING TOOLS FOR SANDIA DEVELOPMENT TESTS

MATTHEW P. VIENS*, WILLIAM E. HART†, AND CYNTHIA A. PHILLIPS‡

Abstract. Scheduling under resource and time considerations is a common problem across disparate engineering fields. As a result, scheduling problems have been formulated in many ways for different applications. We consider a facility scheduling application to support strategic planning and operations for Sandia nuclear weapons testing facilities. In this application, we select a set of tasks based on priority using a best-effort approach for fixed time windows. We adapt formulations from the related Resource-Constrained Project-Scheduling Problem. We express the problem as a mixed-integer program, which we solve using a commercial solver. We have evaluated this approach using synthetic data sets, and have successfully applied it to Sandia facility data sets. On Sandia facility data sets, these new formulations can be solved up to 8 times faster than previous methods, and solutions can be generated in $\sim$2 minutes or less.

1. Introduction. Sandia National Laboratories (Sandia) has many testing and evaluation facilities that support critical mission objectives for United States government stakeholders. Challenges for effective facility management include long time horizons, competing priorities for scarce facility resources, and precedence constraints. Facility schedules frequently change to respond to delays in facility operations and critical resources (including key staff), new work, and to adapt to changing priorities and needs of government stakeholders.

The customer programs include many different projects that Sandia must handle efficiently. Projects are comprised of sub-components of minimal length called *tasks*. Historically, teams of subject-matter experts have manually created facility schedules considering two major factors: demand (including tasks durations, task requirements, and tasks priorities) and resources (including facility capacities, working days, and staff). Across all programs, the demand for facility time often exceeds available capacity, so Sandia's scheduling activities have focused on maximizing total task throughput, accounting for a prioritization of programs in consultation with affected customers.

As a manual component in Enterprise Resource Planning (ERP) processes, revising the schedule for large Sandia facilities can take weeks. By introducing discrete-optimization methods, schedules - and schedule revisions - can be generated within minutes. This addition of efficient optimization models to the ERP systems will enable subject matter experts to quickly assess the impact of changes in facility operations and customer requirements and evaluate new schedules in real time. Further, this capability will enable the rapid evaluation of "what if" scenarios to provide customers feedback on the impact of changes in their programs. Finally, this capability will enable higher-confidence assessments of facility capabilities, since discrete-optimization methods can provide mathematical bounds that demonstrate fundamental limitations on facility throughput given available resources. Thus, facility managers will be able to confidently assess when no possible schedule exists to meet some program requirements.

We developed an optimization tool called Enterprise Resource-Planning Optimizer (ERPO). We describe ERPO's models and formulations. ERPO is a scheduling back-end that is part of ongoing development of new user-facing ERP tools. The ERPO tool generates optimal facility schedules using modern Integer Programming (IP) formulations that are expressed with the Pyomo modeling environment [5, 7] and solved using the Gurobi integer programming solver [6].

*University of Wisconsin-Madison Department of Computer Sciences, mviens@wisc.edu
†Sandia National Laboratories, wehart@sandia.gov
‡Sandia National Laboratories, caphill@sandia.gov

Our facility scheduling problem is non-preemptive, where scheduled tasks are run to completion once started. Our formulations are time-indexed integer-programming formulations that discretize the scheduling horizon into daily time intervals. We adapt Artigues' work [2], from the well known Resource-Constrained Project-Scheduling Problem (RCPSP) [4], to our facility scheduling problem.

The organization for the rest of the paper is as follows. In Section 2, we define the ERPO facility-scheduling application. This application is formalized in Section 3, where we describe several integer-programming formulations. In Section 4, we compare the runtime performance of these formulations on large synthetic data sets. Finally, we summarize conclusions and discuss future work in Section 5.

2. Problem Details. Our facility-scheduling problem is non-preemptive and best-effort-based over a fixed time window. Our best-effort approach gives a reward for scheduling each task while allowing the possibility of not scheduling tasks to avoid infeasible schedules. We need a best-effort approach, rather than mandatory scheduling, since our task sets can contain more tasks than are possible with our constraints. We discretize the window into individual days and consider task durations in positive integer days.

We have resource and task constraints. Resource constraints enforce facility availability and capacity. Tasks have earliest/latest start dates and precedence constraints. A *precedence constraint* requires a following task to start after the preceding task completes. All constraints are hard feasibility constraints.

We pick between tasks using *task priority*, positive integers where a larger value indicates higher importance. Task priority is used on our default best-effort objective, or scheduling goal, called *total priority scheduled* where we sum the priorities of all of the scheduled tasks.

Our facility-scheduling problem differs from the RCPSP in two major ways: mandatory scheduling and objectives used. In the RCPSP, all tasks must be scheduled. Our best-effort approach does not schedule all tasks when it is impossible to do so. Best-effort approaches always have feasible solutions since scheduling zero tasks is feasible for best-effort. Thus, the problems have different objectives. The RCPSP uses a makespan objective, i.e. minimize time to run all the tasks, and we use the total priority scheduled objective. Otherwise, the problems share most concepts and definitions. Artigues presents the RCPSP in greater detail.

3. IP Formulations. We created two formulations for the facility scheduling problem using the multiple formulations discussed by Artigues. The impulse formulation was inspired by his Disaggregated Discrete Time (DDT) formulation. The step formulation was inspired by his Step-based Disaggregated Discrete Time (SDDT) formulation. Both formulations are implemented as Pyomo models in the ERPO tool.

3.1. Notation. The formulations share many common elements of notation that are presented below:

Concept	Symbol
Facility Set	R
Tasks Set	J
Tasks in Facility r	J_r
Window Set	T
Precedence Set	E
Task j Feasible Start Set	T_j
Task j Feasible Active Set	$\hat{T_j}$
Task j Duration	n_j
Task j Earliest Start Time	e_j
Task j Latest Start Time	l_j
Task j Priority	π_j

3.2. Impulse Formulation. The impulse formulation is presented below:

$$\max_{x,y} \sum_{j \in J} \pi_j y_j \tag{3.1}$$

s.t.

$$\sum_{t \in T_j} x_{jt} = y_j \qquad \forall j \in J \tag{3.2}$$

$$\sum_{\tau=0}^{t-n_i} x_{i\tau} - \sum_{\tau=0}^{t} x_{j\tau} \geq 0 \qquad \forall\,(i,j) \in E, \forall t \in T \tag{3.3}$$

$$\sum_{j \in J_r} \sum_{\tau=t-n_j+1}^{t} x_{j\tau} \leq 1 \qquad \forall t \in T,\ \forall r \in R \tag{3.4}$$

$$x_{jt} \in \{0,1\} \qquad \forall j \in J \quad t \in T_j$$

$$x_{jt} = 0 \qquad \forall j \in J \quad t \notin T_j$$

$$y_j \in \{0,1\} \qquad \forall j \in J$$

$x_{jt} = 1$ means that task j started at time t; a value of 0 means it did not.

$y_j = 1$ means that task j was scheduled; a value of 0 means it was not.

(3.1a) is the total priority scheduled objective. (3.1b) connects task starting time using x_{jt} to a task being scheduled y_j. Specifically y_j indicates a job is run if and only if it is scheduled to start at a feasible time. (3.1c) is the precedence constraint forcing job j to start after job i completes. If and only if task i ran at $\tau \leq t - n_i$ can task j run at time t. (3.1d) is the resource constraint saying that of all the tasks that use resource r, denoted J_r, at most one of them could be active at any given time. The second sum is taken over all start times for job j that would force j to occupy facility r at time t.The rest of the formulation is domain declarations for the variables and make this a 0-1 (binary) scheduling formulation.

This formulation is distinct from the DDT formulation in three ways. The first difference is the use of the best-effort total priority scheduled objective. The second difference is in (3.1b) where the DDT approach would replace y_j with 1, which would require each task to run. The third difference is in (3.1d) where we rephrase the resource constraint to assume each facility can only have at most one task at a given time. This matches the client data and simplifies the expression.

3.3. Step Formulation. The step formulation is presented below:

$$\max_{z} \sum_{j \in J} \pi_j z_{j,l_j} \tag{3.5}$$

s.t.

$$z_{jt} - z_{j,t-1} \geq 0 \qquad \forall j \in J \quad \forall t \in T \tag{3.6}$$

$$z_{i,t-n_i} - z_{jt} \geq 0 \qquad \forall (i,j) \in E \quad \forall t \in T_j \tag{3.7}$$

$$\sum_{j \in J_r} z_{jt} - z_{j,t-n_j} \leq 1 \qquad \forall j \in J \quad \forall t \in \hat{T}_j \quad \forall r \in R \tag{3.8}$$

$$z_{jt} = 0 \qquad \forall j \in J \quad \forall t < e_j$$

$$z_{jt} \in \{0,1\} \qquad \forall j \in J \quad \forall t \in T$$

$z_{jt} = 1$ means that a task j started at or before time t; a value of 0 means it did not.

(3.2a) is the total priority scheduled objective and demonstrates $y_j = z_{j,l_j}$ across the formulations. (3.2b) is the step constraint that means if $z_{jt} = 1$ then $z_{j\tau} = 1, \tau \geq t$. (3.2c) is the precedence constraint saying if and only if task i started at or before time $t - n_i$ can task j start at or before time t. (3.2d) is the resource constraint saying that of all of the tasks that use resource r, denoted J_r, at most one of them could be activate at any given time. The rest of the formulation is domain declarations for the variables and make this a 0-1 scheduling formulation.

This formulation is distinct from the SDDT formulation in 3 ways and modifies the behavior in a similar way to the differences in the impulse formulation. The first difference is the use of the best-effort total priority scheduled objective. The second difference is the absence of a mandatory scheduling constraint like $z_{j,l_j} = 1, \forall j \in J$. The third difference is in (3.2c) where we rephrase the resource constraint to assume each facility can only have at most one task at a given time.

There is one major difference between the formulation and the Pyomo model. The formulation has z_{jt} defined over all $t \in T$. This leads to the variable count scaling with the size of the time window. In the Pyomo model, we define z_{jt} over just $t \in T_j$ and leverage two simplifying behaviors: $z_{jt} = 0, t < e_j$ and $z_{jt} = z_{j,l_j}, t \geq l_j$. This means that number of z_{jt} variables, for a fixed task j, scales with T_j rather than T, which prevents the number of variables from scaling with the overall scheduling window.

3.4. Initial Formulation. The initial formulation is presented below:

$$\max_{a,x,y} \sum_{j \in J} \pi_j y_j \tag{3.9}$$

s.t.

$$\sum_{t \in T_j} x_{jt} = y_j \qquad \forall j \in J \tag{3.10}$$

$$n_i + \sum_{t \in T_i} t \cdot x_{it} \le \sum_{t \in T_j} t \cdot x_{jt} \qquad \forall (i,j) \in E \tag{3.11}$$

$$\sum_{\substack{s \in \{0,\dots,n_j-1\} \\ t-s \in T_j}} x_{j,t-s} \ge a_{jt} \qquad \forall j \in J,\, t \in \hat{T}_j \tag{3.12}$$

$$\sum_{t \in \hat{T}_j} a_{jt} = n_j y_j \qquad \forall j \in J \tag{3.13}$$

$$\sum_{j \in J_r | t \in \hat{T}_j} a_{jt} \le 1 \qquad \forall t \in T,\, \forall r \in R \tag{3.14}$$

$$x_{jt} \in \{0,1\} \qquad \forall j \in J \quad t \in T_j$$

$$x_{jt} = 0 \qquad \forall j \in J \quad t \notin T_j$$

$$y_j \in \{0,1\} \qquad \forall j \in J$$

$$a_{jt} \in [0,1] \qquad \forall j \in J,\, \forall t \in \hat{T}_j$$

$a_{jt} = 1$ if task j was active at time t and $a_{jt} = 0$ otherwise. It is always integer when the x and y variables are binary, so we can model it as a continuous variable.

We introduce the initial formulation after the impulse and step formulations since it is a more complicated. This formulation comes from previous work at Sandia and was the first formulation implemented in the ERPO tool.

Since we have already presented the other two formulations, we address points of contrast rather than describing all behavior. Precedence constraints (3.3c) compare computed start times. Because the number of precedence constraints (3.3c) does not depend on the number of time periods, it may scale better as the number of time periods grows compared to precedence constraints (3.1c) and (3.2c). Constraints (3.3f) use the activity variables for a resource constraint equivalent to (3.1d). This is redundant when we have the task-start information in x. Our experiments show that the original formulation takes longer to solve than impulse and step formulations. The original formulation's linear-programming relaxation is likely weaker because of the precedence constraints (3.3c) where the binary x variables are scaled by potentially large time parameters.

4. ERPO Performance. To compare the performance of our formulations we used two major categories of tests: real client data tests and synthetic data tests. We compare the initial, impulse, and step formulations using the real client data. The real client data tests established that both the impulse and step formulations outperform the initial formulation. The real client data is only one dataset, which limits the ability to make general statements about performance on real client data tests. We created a tool, `TaskGenerator`, to create many different varieties of task sets to experiment beyond a single client dataset. All of our tests were done with the most recent versions - as of Summer 2022- of the Gurobi solver and Pyomo modeling language. Unless otherwise mentioned, we used the default settings for the Gurobi and Pyomo tools.

While much of our modifications are inherited from Artigues, we cannot directly leverage his performance-based formulation comparisons, which are largely inherited from Bianco [3], since this analysis was specific to RSPCP problems.

4.1. Real Client Data Tests. We had one task set from the client, which did not include precedence information. We considered several different problems to compare formulation performance. All of the results agreed -within solver tolerances- on objective results, so the timing results is the major difference. The first problem was to schedule the task set with the Biased Total Priority objective. The Biased Total Priority objective is the Total Priority Objective with a penalty term added to promote tasks starting as close to Earliest Start Time as possible. This objective breaks ties between schedules that have the same total priority by minimizing delay, which can decrease the number of optimal solutions and lead to increased problem difficulty. The second problem was to schedule the task set after changing, adding, and removing some tasks with the Biased Total Priority Objective. The third problem was the same as the second problem with a larger tolerance setting. The fourth problem was to schedule the original task set with the Total Priority Objective.

In general, the core statistic our users care about is the *end-to-end* time, or the wall clock time between saying run and the ERPO tool returning the schedule. The results of the end-to-end times, rounded down to the nearest second, on the 4 problems are below:

Problem	Initial	Impulse	Step
1	271	108	99
2	951	90	136
3	111	93	36
4	382	91	90

4.1.1. Real Client Data Analysis. On the most difficult problem for end-to-end time, problem 2, both the impulse and step formulations were around 8x faster than the initial formulation with this specific data. This corresponds to the full scheduling problem, both scheduling the highest priority set of tasks possible and minimizing task start delay.

There was one result from model solve time, that is, the time the Gurobi solver takes to solve the constructed Pyomo model, that was interesting for feasibility analysis. On problem 4, the impulse and step formulations outperformed the initial formulation on model solve and end-to-end time. The model solve time here is what would control repeated runs, or sweeps, of "What If" scenarios, where multiple models with minor variations are solved and contrasted, since problem 4 used the total priority scheduled objective. This results in 29x faster (525 vs 11 seconds), on the impulse formulation, for the "What If" sweeps with this specific data.

We also tested the impact of the Gurobi solver's presolve settings, since Achterberg et al. [1] showed the influence of presolve. Default presolve was either comparable or superior to the other possible presolve settings.

Overall, our impulse and step formulations outperformed in every problem in model solve and end-to-end times. This demonstrates clear gains from the impulse and step reformulations for our problem with this specific data.

4.2. Synthetic Data Tests. When we consider synthetic data we can ask two major questions beyond the real client data tests. The first question is what happens when we add precedence constraints. The second is how does runtime vary with task properties. All of the real client data tests used a single set of tasks, modified slightly depending on the problem, and lack precedence constraints. We expect the ERPO tool to be used to schedule many different task sets and to have precedence data.

The synthetic data comes from our `TaskGenerator` utility and leverages Python's pseudorandom-generation tools to create task and precedence data from uniform distributions. Since random generation could lead to infeasible tasks or precedence relationships, we ensure that only feasible tasks and precedence relationships are returned.

We did many different tests varying the number of tasks generated from 50 to 600, and the length of the scheduling window from 1 to 3 years.

We used the Biased Total Priority scheduling goal, the same as most of the real client data tests. Priority values came from a uniform distribution of 1 to 10. Precedence constraints occurred on approximately 50 percent of the tasks.

We recorded the end-to-end time for each test on both the impulse and step models and present those times, rounded to the nearest tenth of a second, from an average of 3 tests:

4.2.1. Impulse Formulation Tests.

Tasks	1 Year	2 Years	3 Years
50	2.8	2.4	4.8
100	2.6	5.1	8.7
200	4.7	11.1	18.0
400	10.3	24.3	42.5
600	18.6	39.2	65.5

4.2.2. Step Formulation Tests.

Tasks	1 Year	2 Years	3 Years
50	0.5	0.7	1.0
100	0.9	1.3	1.7
200	1.7	2.5	3.5
400	3.9	4.4	5.4
600	16.4	6.6	10.0

4.2.3. Analysis. After the introduction of frequent precedence constraints, the step formulation outperforms the impulse formulation. Also, the end-to-end times for the synthetic data are markedly faster than the results from the real client data tests. While it is hard to exactly compare the problems, there are many factors that could account for this difference including use of precedence constraints and structure from the `TaskGenerator` differing from real client data structure. Between the two elements, we can continue to expect that the step formulation will outperform the impulse formulation when precedence relationships are common. We must continue our testing with synthetic data to further consider changes in number of facilities, task duration, and task priority.

5. Conclusions. Our ERPO tool provides significant improvements to the facility scheduling process for Sandia. The decrease in end-to-end scheduling time, over both manual and previous automated methods, provides the ability to quickly schedule on real client exemplar data in 2 minutes or less. Additionally, our tests on synthetic task data demonstrate the value and power of including the precedence information in the data inputs, which simultaneously causes further matching of the ERPO results to manual scheduling behavior and restricts our feasible solution set causing apparent runtime gains.

We have many avenues for future work on the ERPO falling into two categories: functionality improvements and performance testing. The functionality improvements include presentation of multiple near-equivalent scheduling options, what-if modeling for facility capacity variations, and additional solver setting modifications. The performance testing has two directions: fine-tuning the data generation parameters and runtime testing. For the data generation parameters, we presently use uniform distributions for all our synthetic data, but the real data may conform better to other distributions. For the runtime testing, we showed that it is feasible to quickly schedule hundreds of tests over years. Seeing where and why the facility scheduling problem goes from easily tractable, with our formulations, to intractable is of interest to our clients.

Overall, our ERPO tool is capable of marked gains in facility scheduling versus our previous scheduling automation attempts with gains of up to 8x and 29x on full scheduling and what-if analysis respectively and serves an ongoing client need for robust and fast facility scheduling at Sandia as a core component of the ERP application.

REFERENCES

[1] T. Achterberg, R. Bixby, Z. Gu, E. Rothberg, and D. Weninger, *Presolve reductions in mixed integer programming*, INFORMS Journal on Computing, 32 (2019).

[2] C. Artigues, *A note on time-indexed formulations for the resource-constrained project scheduling problem.* 15 pages, June 2013.

[3] L. Bianco and M. Caramia, *A new formulation for the project scheduling problem under limited resources*, Flexible Services and Manufacturing Journal, 25 (2011).

[4] P. Brucker and S. Knust, *Complex scheduling, second edition*, 01 2012.

[5] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, vol. 67, Springer Science & Business Media, third ed., 2021.

[6] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2022.

[7] W. E. Hart, J.-P. Watson, and D. L. Woodruff, *Pyomo: modeling and solving mathematical programs in python*, Mathematical Programming Computation, 3 (2011), pp. 219–260.

III. Applications

Articles in this section discuss the application of computational techniques to simulate physical systems. This includes the use of molecular dynamics, multiscale models, or machine learning to provide insights into materials, high-energy applications, atmospheric phenomena, and structural mechanics.

1. *Ma, Karoui, Karoui* and *Sikorski* enhance the process of carbon nanotube characterization by utilizing molecular dynamics to develop a carbon nanotube database for future machine learning predictive materials work.
2. *Reyes, Smith* and *Severa* use recurrent neural networks to determine leading jet membership in a heavy ion colliding application.
3. *Sema* and *Wood* use machine-learned interatomic potentials to simulate molecular dynamics and predict vapor-liquid equilibria and phase transitions.
4. *Torchinsky* and *Taylor* develop new interfaces for the Non-Hydrostatic High-Order Methods Modeling Environment for global atmosphere modeling, with applications to a tropical cyclone test case.
5. *Wilson* and *Silling* adapt the peridynamic numerical method for simulation of colliding objects to use a viscous contact force to avoid interpenetration errors.

S.K. Seritan
J.D. Smith

November 11, 2022

THE PROCESS OF RESEARCHING CARBON NANOTUBES AND ITS PROPERTIES

BRANDON MA*, ABDENNACEUR KAROUI†, FOZIA SAHTOUT KAROUI‡, AND EMBER SIKORSKI§

Abstract. Carbon nanotubes (CNTs), particularly single-walled carbon nanotubes (SWCNTs) in this case, are tubular nanostructures possessing nanoscale properties that allow the nanostructure to be utilized in a wide variety of applications. However, traditional experimentation with hybrid piezoelectric composites, or artificial piezoelectric materials, and their properties such as electric, mechanical, and thermal properties are costly and time-consuming. Furthermore, there is a scarce amount of complete CNT databases and publications on CNT's mechanical properties and the impact of chirality on its mechanical properties. This leads to the primary goal of this research; to develop a complete, open-source database of CNTs properties by utilizing molecular dynamics (MD). With MD, the materials and properties of interest can be simulated and explored without traditional constraints. However, the vast resulting data regarding many materials' properties are too numerous to be fully analyzed manually hence the adoption of machine learning (ML) aspect in this research as well as automation. Once the CNT database has been sufficiently populated and its entries verified, the data will be used to develop a machine-learning model that will predict the CNT's electric and mechanical properties with the hopes that it will eventually lead to the development of new materials or properties.

1. Introduction. Possible applications that CNTs are suitable for are bio-filtering [6], nano-filtration, water desalination, and sensors. All of which can provide a significant technological breakthrough. The lack of a reliable and complete CNT database hinders the engineering and research of the material, expending time and resources. Additionally, the importance of a CNT's chirality and how it influences its properties requires more observation and research. The above-mentioned circumstances and the absence of data on CNT's piezoelectric and mechanic properties add further setbacks as those properties are a crucial aspect of understanding CNTs and their potential.

To address this issue, we turn to molecular dynamics (MD) as our mainstay with the addition of machine learning to enhance the research. A CNT of varying chirality and length is generated using a molecular modeling and visualization computer program that will be used as an input file for LAMMPS. The input file will then be properly prepared and subjoined by dedicated LAMMPS scripts for that CNT to run through LAMMPS, and the resulting LAMMPS output files will be extracted and analyzed. This process will repeat until the data accumulated has been deemed sufficient which the machine learning aspect can begin. As mentioned earlier, the amount of data that needs to be prepared, processed, and analyzed is very time-consuming.

To alleviate some of the data intensity, a sequence of our data will be omitted and subjected to the machine learning model to predict and calculate possible properties of the CNT. Machine learning (ML) is a field of study where computers are programmed to learn from the data and subsequently improve as they accumulate experience in performing their assigned tasks and is a key aspect of the research as it progresses. Initial predictions will be limited to the likes of mechanical and will progress to other properties like thermal and piezoelectric. In addition to predicting the material's properties, the ML model will be used to generate data that will virtually produce new materials or properties. To develop a high-accuracy ML model, a large amount of data, computational data-driven methods, and numerous testing are required to ensure a reliable model.

*North Carolina Central University, bma@eagles.nccu.edu
†North Carolina Central University, akaroui@nccu.edu
‡North Carolina Central University, fkaroui@nccu.edu
§Sandia National Labs, elsikor@sandia.gov

The integration of LAMMPS and machine learning will provide the necessary data, tools, and methods to predict and eventually develop new materials and properties. This will lead to a pattern detected by the ML model giving insight into what kind of properties a CNT will have; how certain CNTs of varying chirality and length will produce a specific spectrum of properties, serving as a guide to determine the properties of a CNT. Precaution must be taken with the data generated from the ML model and initial training data sets must be verified for accuracy before they could be entered into the CNT database. Subsequently, the CNT database will become an open-source database that allows users to access and search for particular properties of interest to them. Another alleviation of the time consumption of preparing the data is through automation.

Using Python, an automation script will quickly modify and develop the necessary scripts and requirements for the data to run in LAMMPS [7]. Additional calculations can be performed using the automation script such as direct measurement of the CNTs length and diameter to ensure that the CNT is not too short or long than the estimated projections. Furthermore, the post-processing of the LAMMPS data after a successful run can be generated using automation to find the stress-strain curve, size, volume, and temperature. A comma-separated file (CSV) and graphs of each parameter will be supplementing the post-processing results, generated by the automation script. All these tools, methods, and data are an important part of this research leading up to the creation of the CNT database.

This database will allow users to search for particular properties ranging from mechanical properties to thermal properties. At the time of this paper, the machine learning aspect is still in early development. As such, automation and other subjects will be the most prevalent in this paper.

2. Creating the prototype. The CNT structure and properties selection will stem from the chiral indices (n,m), which in turn influence the diameter and length. Both of which are measured and calculated in nanometers (nm). The CNT diameter, d, is calculated using the following equation:

$$d = 0.0783 * (((n+m)^2 - n * m))^{0.5} \tag{2.1}$$

The length, l, is calculated by a simple equation derived from the CNT diameter and aspect ratio of 10.00:

$$l = 10.00 * d \tag{2.2}$$

Another condition to adhere to is that the structure must have at least a chiral index of 4 or it will not be a nanotube. For example, a structure can have the chiral indices of (n,m) = (1,4), but cannot have chiral indices of (n,m) = (1,1) or (n,m) = (2,3) as these chiral indices will create a structure that is not a nanotube. Should all the proper conditions be met, a prototype will be created using Visual Molecular Dynamics (VMD) to create the structure's atom coordinates into three file formats: program database (.pdb), XYZ (.xyz), and a dump file (.dump) that will be modified into a data file (.dat). Both .xyz and .pdb are recognized and/or universally adopted for data and serve as a reference in the database. Meanwhile, .dump is converted to .dat which serves as one of the input files for LAMMPS.

2.1. File processing automation. The initial process of converting the .dump file into a .dat file is a manual process. This extended to the CNT's directory, input script, and bash script; all of which were repetitive and tedious to handle manually. Furthermore, the scripts were to be modified in accordance with the number of atoms, the structure's chiral indices, and diameter leaving potential room for human error. To offset the margin of human errors and streamline the task, automating the process will greatly decrease the time and resources expended on the file processing task. A high-level and general-purpose programming language called Python was chosen to create the automation script as there is a vast amount of existing and immediate resources for automation using Python [4].

Multiple Python scripts were created with each performing a specific function for ease of debugging and testing. These scripts possess imports from Python's existing libraries such as os [1], sys [3], and shutil [2] for the automation to handle file names, paths, and directories. All scripts are tested and debugged using various inputs to develop a more robust automation script that can handle or ignore unexpected and/or incorrect inputs. These scripts will then be integrated into an encompassing Python script and tested with each additional function per iteration for any issues before continuing any more integration. This cycle of testing, debugging, and integration resulted in a suitable Python automation script that can handle, copy, modify, and relocate files within seconds whereas it would have taken minutes, possibly an hour to manually process even a handful of files.

2.2. Using automation to resolve issues during LAMMPS. There are occasions in the research where additional processes are needed to address the issues that have occurred in the MD simulation. Addressing those issues manually is very time-consuming and error-prone. Hence, the solutions to these problems are implemented as additional features in the Python automation scripts. There are two notable problems that the research team encountered during the research. One such issue is the boundary box holding the structure will try to close in on the CNT.

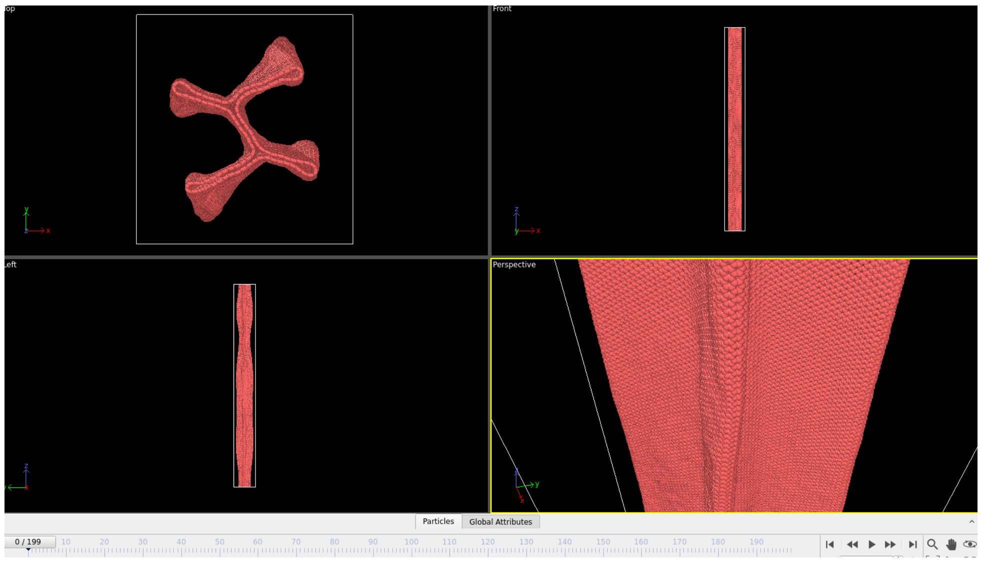


FIG. 2.1. *CNT(99,99) with a diameter of 13.43 nm is greatly deformed at zero step*

However, the 'closing in' motion can cause the CNT to flatten and a solution was to have the box boundary size be two angstroms longer than the CNT in the X and Y directions. To do this, the automation script seeks out the maximum and minimum atom coordinate in the X and Y directions then adds or subtracts two angstroms to those coordinates and appends the new box boundaries to the data file. While this solution has resolved numerous MD simulations, it does not address cases where the structure is already greatly deformed as seen in Figures 2.1 and 2.2 above. This often occurs in structures containing a large number of atoms. This behavior is likely due to the properties of carbon atoms, their chirality,

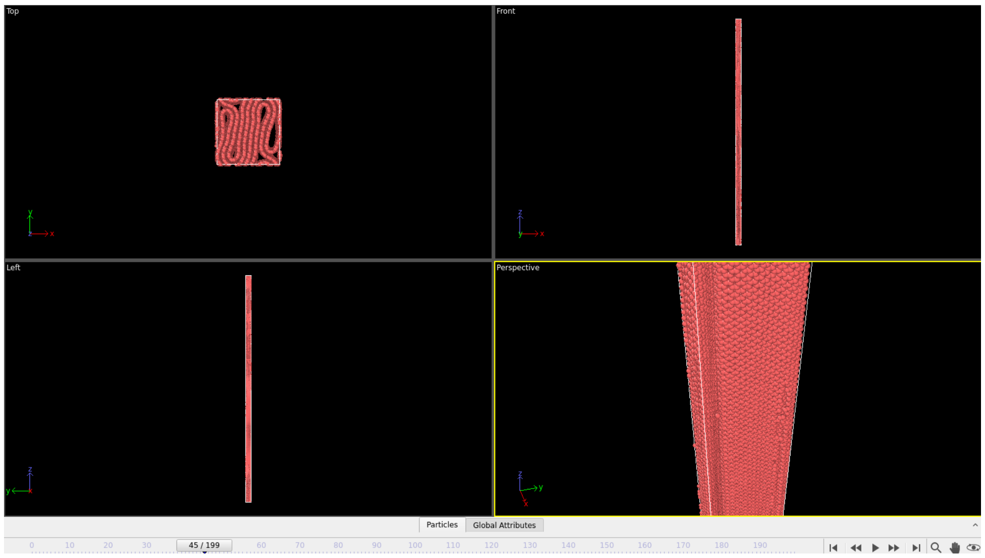


Fig. 2.2. *Deformed CNT (99,99) during tensile test*

and/or the CNT simply does not exist in reality.

Determining the true diameter and length of a CNT has also been a center point for generating the appropriate data for the CNT databases. The formulas, as shown in Section 2, used to calculate the diameter and length of the structure are but an estimate of the CNT's true diameter and length due to factors like atomic bond lengths. Therefore, a direct measurement of the structure is required to confirm whether the diameters and lengths are accurate enough following the aspect ratio of 10 as dictated by the research team. The initial solution of utilizing the distance formula was erroneous due to not understanding the proper approach in a three-dimensional structure. After adjusting the script, it records the min and max of the X, Y, and Z directions, then calculates the dimensions using the distance formula.

Once the direct measurements have been calculated, the results are generated into a text file for reference.

2.3. Post-processing tensile test. After a successful molecular dynamic calculation in LAMMPS, the output data is to be post-processed for data analysis. The trajectory file is inputted into a software called OVITO to visualize the CNT being pulled in the Z direction as seen in 2.3. Previously, the output was manually copied and pasted into Excel if the simulation did not yield any unusual behaviors. The Excel workbooks performs the calculations to yield plots of various parameters of interest such as temperature, stress-strain curve, potential and kinetic energy, size and volume, etc. However, this process is tedious as it requires manual adjustments of graphs to fit the data and these values can vary greatly between CNTs.

To make post-processing simpler, a Python script was created to automate the necessary calculations and plotting. The automation script will create a .csv file in conjunction with the graphs and compile the files into their respective folders. Figure 2.4 and 2.5 are one of the many post-processed graphs of a carbon nanotube, particularly a CNT with chiral index of (1,6) and diameter of 0.51nm.

In certain instances, specific graphs will include a title detailing and highlighting important information as shown in Figure 2.6. As one can see in most of the post-processing

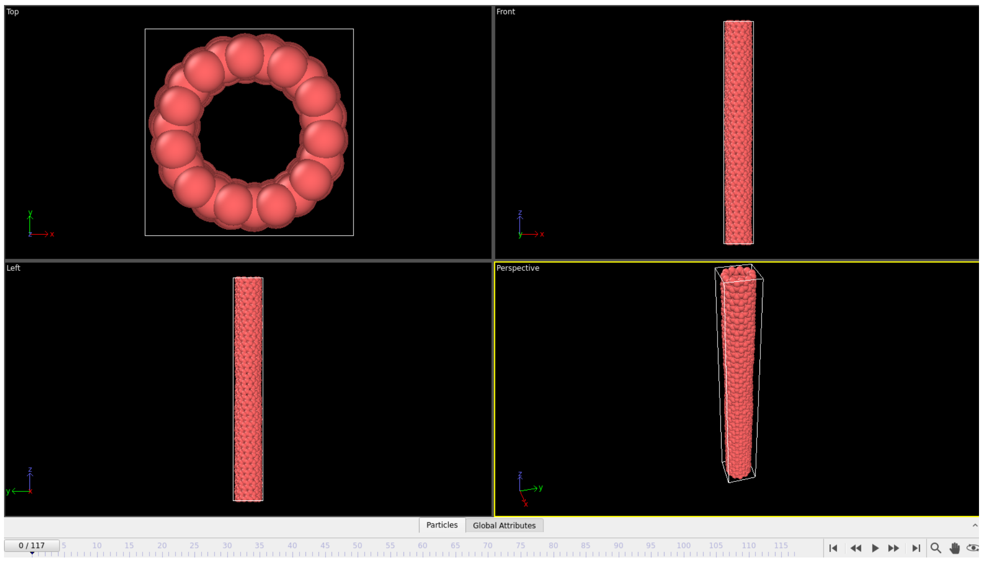


Fig. 2.3. *CNT visualized and about to be pulled*

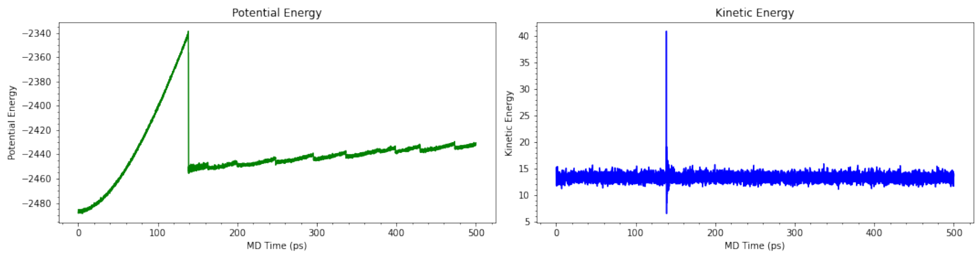


Fig. 2.4. *Plot of CNT potential and kinetic energy*

graphs, there is a noticeable uptick or downtick in the plot when the CNT fractures during the tensile test simulations. A massive drop in potential energy, and a large spike in kinetic energy and temperature when it fractures; these pivotal moments help the team analyze and understand the CNT's properties.

3. Thermal conductivity. LAMMPS does not calculate thermal conductivity directly and instead provides the necessary data to perform the calculation for thermal conductivity. There are several methods to calculate thermal conductivity. Using auto-correlation of the heat flux after the simulation is one method. Another is finding the temperature gradient to extract and post-process. However, some of the methods require significant modification with LAMMPS scripts making them ill-suited in these instances.

In this case, we use heat flux and auto-correlation to find the thermal conductivity. Drew Rohskopf has provided LAMMPS scripts to the research team for calculating the phonon density of states and heat fluxes, but the post-process aspect required some adjustments to suit our needs. Instead of post-processing the data through LAMMPS, the calculations are done in Python as it is quicker and more accessible. Furthermore, we can modify and plot the data with more ease than in LAMMPS. Collaborating with Drew, the script does the following. It extracts the heat versus time from LAMMPS output after a successful MD

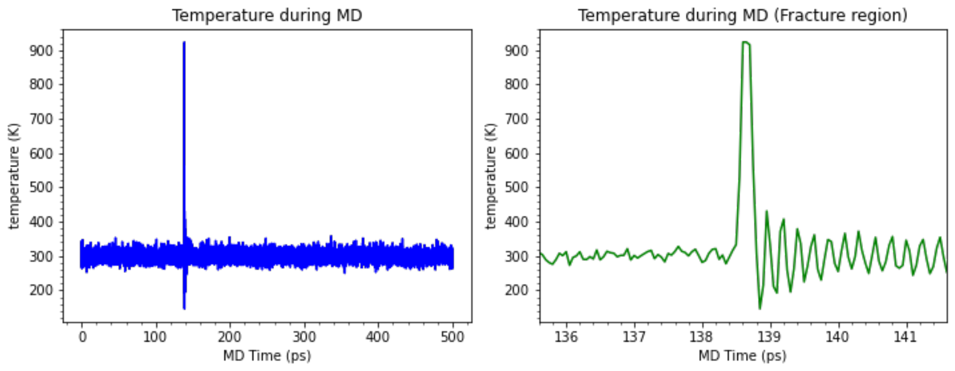


FIG. 2.5. *Plot of temperature during the simulations*

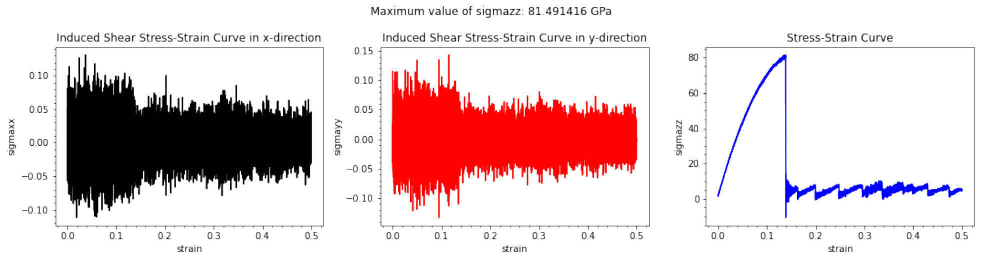


FIG. 2.6. *Induced shear stress-strain curve plot*

run; plots heat vs time and mean flux to show that the system is in equilibrium; lastly, the script calculates heat flux autocorrelation to measure the heat flux signal. One of the results can be seen in Figure 3.1; the heat flux and time of CNT (1,19) with a diameter of 1.53nm.

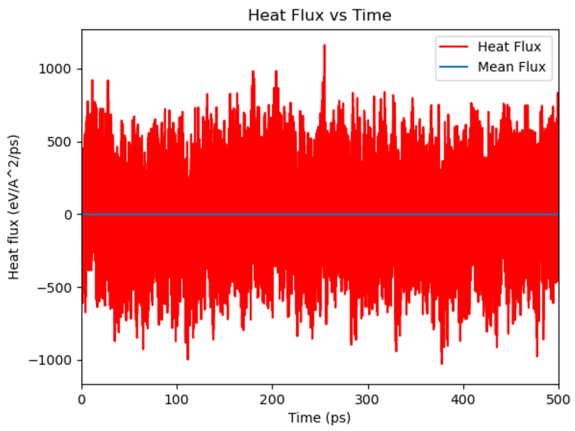


FIG. 3.1. *Graph of the heat flux and time*

4. Density functional theory of CNT. While density functional theory (DFT) has yet to be addressed by the research team, it is an area that will be included in the

CNT database. To compute the energies and forces of carbon nanotubes, we use SeqQuest, a general-purpose electronic structure code for computing energies and forces of periodic surfaces (slabs), solids, and finite molecules to simulate density functional theory [5]. Unlike previous CNT structures, we limit the number of atoms to around 20-40 atoms as the structures will be interlocking on top of one another for the simulation. The pseudo-potential used for the simulation is carbon, particularly carbon long, and the input script is modified accordingly to the structure such as primitive lattice vectors, grid dimensions, number of atom types, atom coordinates, etc. Upon completing a successful run, an output file will be created, providing a detailed list of the occurrence and its results.

5. Results and preliminary analysis of tensile test. The tensile test is a material characterization where we apply strain on the material in LAMMPS. We then plot the response of the material and get a graph like Figure 5.1. The data analysis focuses on mechanical properties where we extract the elastic modulus, yield strength, and fracture strength as well as other data from the test. However, this data analysis has yet to be explored in-depth due for the time being. The following is an observation of the data that has been post-processed.

Post-processing the simulated CNT yields data that has been mostly consistent throughout all CNT tensile test simulations though the impact of their polymer characterization, i.e. armchair, zigzag, and chiral, requires more in-depth analysis. The plotting of data shows that there is a noticeable behavior of CNT with a smaller number of atoms and those with a large number of atoms. Though what constitutes a small or large system is currently up for debate, one can make an educated guess by looking at the chirality. The higher the chirality, the higher the number of atoms there are. Structures containing lesser amounts of atoms often have a sharp and considerable fluctuation whereas the size variation has a relatively stable fluctuation in Figures 5.2 and 5.3.

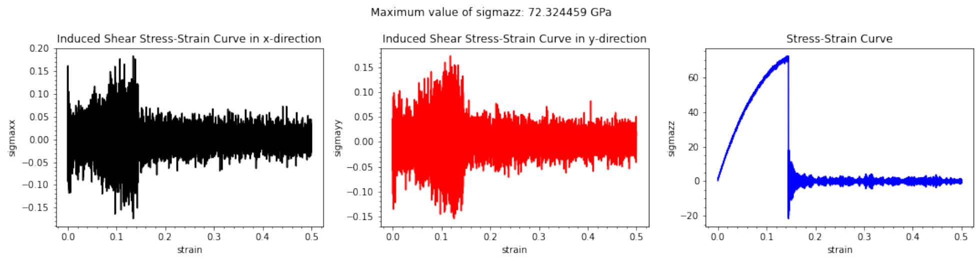


FIG. 5.1. *Stress-strain curve of CNT (0,4) with a diameter of 0.31 nm*

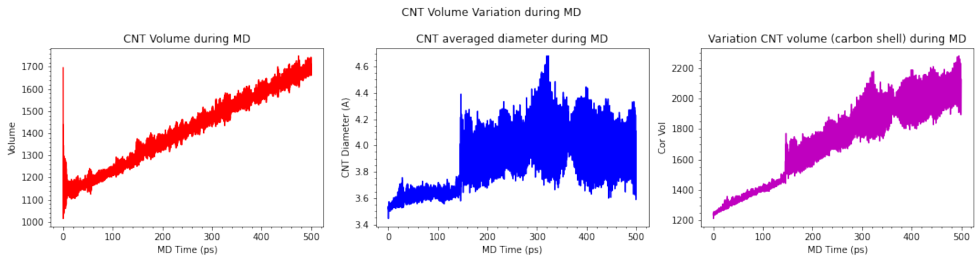


FIG. 5.2. *Volume variation of stress-strain curve of CNT (0,4) with a diameter of 0.31 nm*

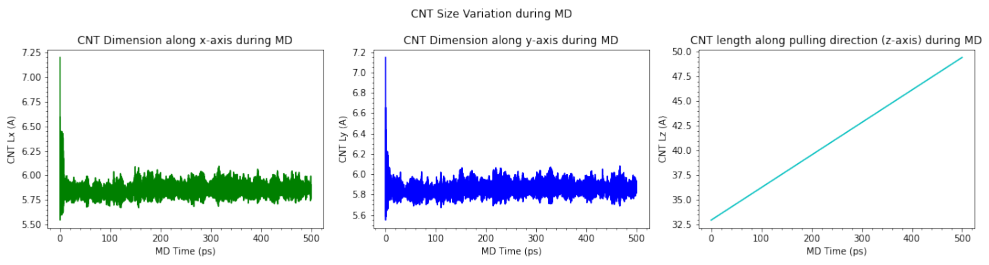


FIG. 5.3. *Size variation of stress-strain curve of CNT (0,4) with a diameter of 0.31 nm*

A similar result can be seen in Figure 5.4 whose structure also has a low count of atoms. Structures with a higher number of atoms differentiate themselves when observing the volume and size plots. The plot of volume variation initially starts steeply and fluctuates gradually. Meanwhile, the plot detailing the variation of size begins with a dramatic flutter that sharply increases or decreases depending on the structure as seen in Figure 5.5.

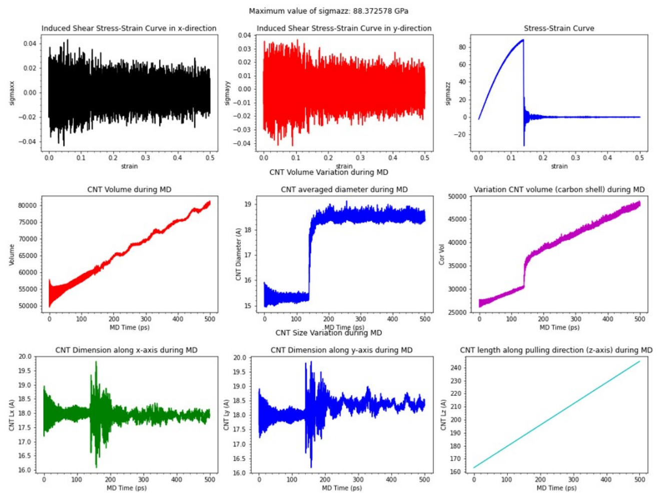


FIG. 5.4. *Plots of CNT (1,19) with a diameter of 1.53 nm*

Once again, the plot of volume and size of CNT (0,90) with a diameter of 7.05 follows a similar pattern. It is important to note that the size graphs in Figure 5.7 could be regarded to be remotely opposite of its counterpart in Figure 5.5. The reason for this is currently unknown at the time of this report and will be discussed as the analysis continues.

Recall Figures 2.1 and 2.2 of a CNT (99,99) with a diameter of 13.43 nm. Because the CNT starts deformed during the simulations, it has affected the post-processing results. This oddity is likely because of the general structural properties and chirality as described in Lu-Chang Qin's article [5]. Likely, CNT (99,99) might not even be possible to create hence the strange behavior during the simulation. However, there is not sufficient evidence for any of the aforementioned, and will require further analysis and tests. The results from the data analysis of CNT (99,99) can be seen in Figure 5.8

6. Conclusions. Preliminary analysis of the outputs has given mostly consistent results of how CNT behaves during the tensile test simulations. This is a good sign that the research is on the correct track, but more testing and analysis of the CNT are needed

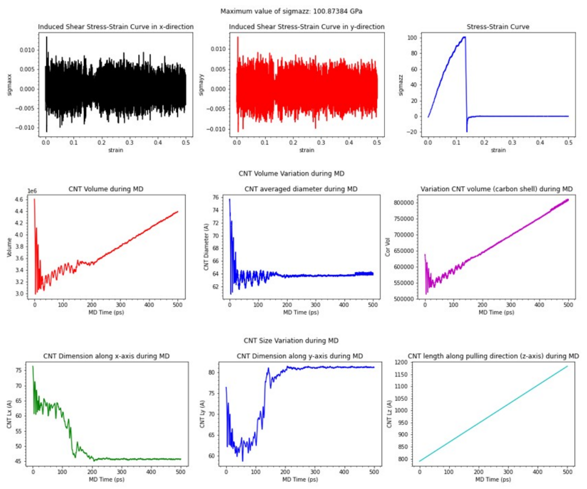


Fig. 5.5. *Plots of CNT (1,94) with a diameter of 7.40 nm*

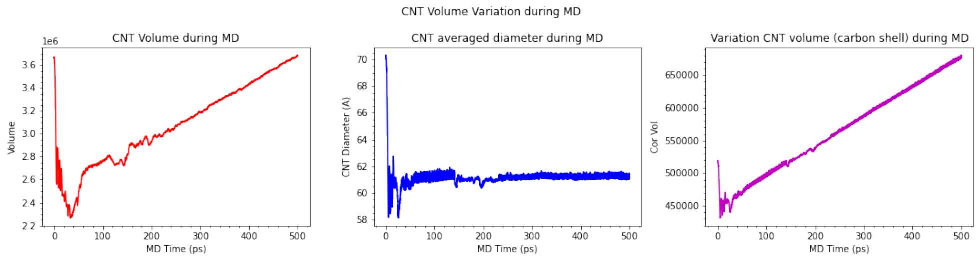


Fig. 5.6. *Volume variation of stress-strain curve of CNT (0,90) with a diameter of 7.05 nm*

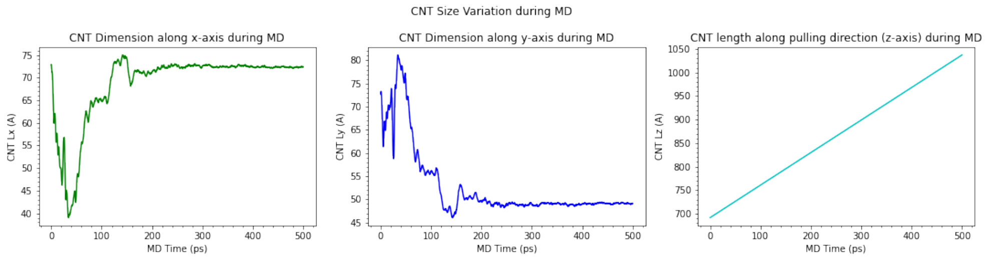


Fig. 5.7. *Size variation of stress-strain curve of CNT (0,90) with a diameter of 7.05 nm*

to solidify research holdings. As it stands, the CNT has a discernible behavior during the tensile tests. However, there are a few outliers concerning CNT with large systems that will require further investigation. Once the outliers have been addressed, the research team should be able to revise their approach and limitations.

The process of researching CNTs and their properties requires several tools and methods to obtain the desired results. There were several ways to address the lacking areas in this research. The tools and expertise that are available dictated the direction of the process and have yielded substantial results. What was once tedious, repetitive, and manual tasks have been automated to streamline the process within seconds or minutes. Calculations

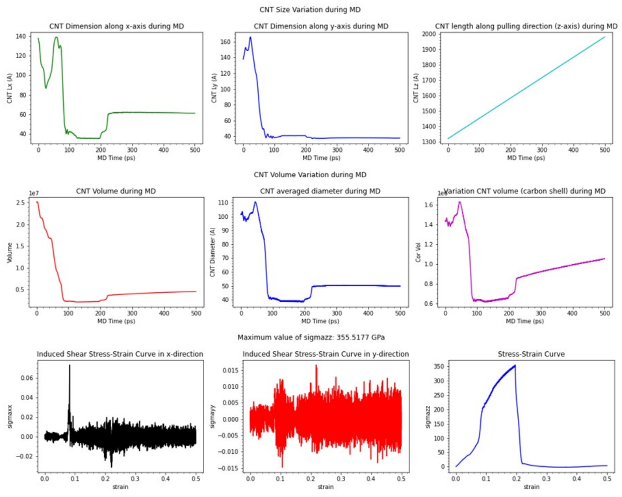


Fig. 5.8. *Graphs of CNT (99,99) with a diameter of 13.43 nm*

that require careful examinations of their inputs can be done during the automation process and integrated into other series of calculations that influence the simulations.

Although the research is ongoing and far from over, the tools developed to assist in the research will save time and resources. Time and resources that the research team can allocate to other crucial work. It also cannot be understated that the room for human errors has decreased considerably with the introduction of automation tools, thus improving the accuracy and quality of the data outputted though it will still be reviewed and compared to existing data. As the research progresses, changes and adjustments are likely to occur requiring an overhaul of the research team's approach. These can be adapted into the automation scripts ensuring that the research continues without significant delays.

Acknowledgments. This work is supported by the National Nuclear Security Administration award NA0003979, and the NSAM-ML project. I would like to thank Abdennaceur Karoui [1] and Fozia Sahtout Karoui [2] for their advice and feedback. Megan McCarthy [3] for her Python and OVITO expertise; Drew Rohskopf [4] for assisting me in conducting thermal properties and thermal conductivity calculations; Peter A. Schultz [5] for teaching me SeqQuest; Jonas Albert Actor [6] for his expertise on machine learning, and David John Littlewood [7] for giving me this wonderful opportunity at Sandia National Labs. Thank you all.

REFERENCES

[1] P. S. Foundation, *os — miscellaneous operating system interfaces*, n.a.
[2] ———, *shutil — high-level file operations*, n.a.
[3] ———, *sys — system-specific parameters and functions*, n.a.

[1]North Carolina Central University, akaroui@nccu.edu
[2]North Carolina Central University, fkaroui@nccu.edu
[3]Sandia National Labs, megmcca@sandia.gov
[4]Sandia National Labs, adrohsk@sandia.gov
[5]Sandia National Labs, paschul@sandia.gov
[6]Sandia National Labs, jaactor@sandia.gov
[7]Sandia National Labs, djlittl@sandia.gov

[4] ———, *What is python? executive summary*, n.a.
[5] P. A. Schultz, *Seqquest electronic structure code*, 2018.
[6] P. K. D. S. A. J. Shiv Kumar Prajapati, Akanksha Malaiya, *Biomedical applications and toxicities of carbon nanotubes, drug and chemical toxicology*, Taylor and Francis Online, (2020).
[7] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, *Lammps - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Comp. Phys. Comm., 271 (2022), p. 108171.

TOWER AND JET PREDICTION USING MACHINE LEARNING, A HEAVY ION COLLIDING APPLICATION

CLARISSA REYES*, J. DARBY SMITH†, AND WILLIAM SEVERA‡

Abstract. In heavy ion colliding physics, vast amounts of data are generated rapidly. Storage of all generated data is not feasible or advised as much data produced is not of interest. Special circuits and detectors are designed to record only data that is considered of interest. This process, termed jet triggering and jet finding, can be accelerated through intelligent algorithms. This work seeks to build an initial machine learning approach toward jet finding through the use of recurrent neural networks. Training on real experimental ion collision data and on simulated ion collision data, we are able to predict whether cells detecting energy levels belong to the leading jet in a collision with over 97% accuracy for both types of data. Additionally, we use total energy as a means to identify an entire jet pattern, finding that our algorithm correctly picks out the leading jet within 20 GeVs.

1. Introduction. Particle accelerators and colliders have a deep history spanning nearly a century [18]. The utility of these experiments has been paradigm shifting, resulting in the discovery of various subatomic particles [10] and providing a means to study the conditions shortly after the big bang [21]. Related to the latter, a primary goal of the Relativistic Heavy Ion Collider (RHIC) at Brookhaven National Laboratory is to produce and analyze a rare form of matter: the Quark-Gluon Plasma (QGP) . Experiments in this realm found that some measured properties deviated strongly from prediction based on the expected underlying theory, perturbative Quantum Chromodynamics (pQCD/QCD), changing assumptions on the gas state [22].

Quantum chromodynamics, in general, has accelerated understanding of the proton, driven by the study of its components and how those components contribute toward the spin of the proton. The spin of a proton is a quantized value (in half-integer multiples of $\hbar$) representing the intrinsic angular momentum. Prior to a 1987 experiment by the European Muon Collaboration, it was expected that the spin of the proton would be carried by its valence quarks (two up quarks, one down quark). However, the experiment determined that the quarks carried little-to-none of the proton spin [14].[1] Following this discovery, termed the proton spin crisis, physicists have turned their lens toward the other components of a proton in search of the missing momentum: gluons and quark/anti-quark pairs. Current theory suggests that proton spin includes quark or gluon orbital angular momentum [16].

Both our understanding of the transport properties in the pQCD regime and the components carrying proton spin are active areas of research. These require data on the gluon and quark/anti-quark components of a proton in order to study transport properties, like spin asymmetries and cross sections (a probability measure for particular colliding processes to occur) [5]. The required experimental data can be challenging to collect. For the purposes of discussion, consider the collision of two protons at RHIC. Beams of protons are accelerated to relativistic speeds, traveling in opposite directions along a 2.4-mile track. At intersections along this track, protons traveling in opposite directions will have a chance to collide. If a collision occurs in just the right way, the quarks and gluons are liberated and undergo a hadronization process whereby they transform or decay into hadrons. The distribution of these hadrons are captured by a detector, which can be a hadron calorimeter

*University of Texas at El Paso, cvalles7@miners.utep.edu

†Sandia National Laboratories, jsmit16@sandia.gov

‡Sandia National Laboratories, wmsever@sandia.gov

[1]The expectation was that since quark and anti-quark pairs should cancel each other out in terms of spin, the only remaining contributor to spin should be the valence quarks that aren't paired. Two ups plus one down should yield the expected $\hbar/2$ spin. That this was not the case was a very shocking result.

or an electromagnetic calorimeter [7, 6, 4]. Data must be able to be acquired at the rate proton beams pass through the intersection points, which is about 10 MHz, or about 10 million events per second. This is obviously quite extreme and would produce a lot of data if ran for any amount of time. Not all data taken will come from a collision producing hadrons. In order to save only the most important data, a trigger is put in place capable of examining the data and determining whether to send it off for analog-digital conversion and saving. A so-called fast trigger capable of examining data at 10 MHz is the STAR trigger [3].

Once data is acquired, energy patterns representing hadrons must be separated from underlying events. This process is called jet finding, where a jet is an energy pattern representing one or more hadrons. This process can be visualized as in Figure 1.1. We see a hypothetical example of a hadron energy pattern where cells are color coded by their energy level. If the energy of a cell reaches a certain threshold, it is labeled a tower. Prominent clusters of towers represent jets. The collision can produce several jets at a time. Jet finding algorithms determine which towers belong to jets and which belong to an underlying event. A prominent jet finding algorithm is called the anti-k_t algorithm [9].

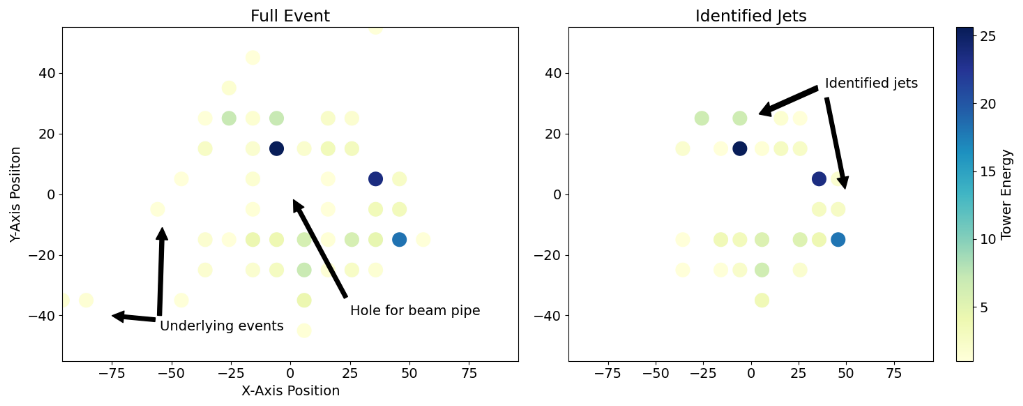


Fig. 1.1. *Hypothetical visualization of cell energy levels in GeV. In this illustrative event, we can see how the data in full event (left) is filtered to retain the most relevant parts through jet finding (right).*

The process of data acquisition and jet finding currently happen separately or on different time scales [3]. Ideally, the saving, or triggering, of data and the jet finding could happen at the same time or could be done with more intelligence, targeting explicit hadrons of interest. Such an approach would require probabilistic computing and machine learning approaches. However, any such approach is doomed to fail if machine learning cannot identify whether towers belong to jets as opposed to underlying events. A successful demonstration that a machine learning approach can identify towers is a step towards identifying jet patterns and providing a machine learning jet finding algorithm.

In this work, we demonstrate such a machine learning approach for determining tower membership in the jet and lay the groundwork for future jet finding results through statistical analysis. The remainder of this paper is organized as follows. In Section 2, we discuss the need for a probabilistic approach for jet triggering and finding, and further motivate our initial machine learning algorithm. In Sections 3 and 4 we describe and visualize our data, then detail our jet finding machine learning algorithm and statistical analysis. Finally, in Section 5, we discuss our future directions.

2. Jet Finding and the Need for Probabilistic Computing. As previously described, jet finding is a complicated process. First, data is taken at the speed of proton beam crossings (10 MHz). Then, the data must be classified on the fly as to whether or not it is a gluon or quark jet so that the data can be saved. Then, once data is saved, it must be analyzed through use of a jet finder algorithm in order to determine which parts of the saved data constitute jets.

Once jets have been found, physicists would like to categorize the jets as gluons or various types of quarks. Each category has a particular distribution, but not all are able to be easily discriminated. For instance, distinguishing between gluons and light quarks can be quite challenging and has been a focus of research [19, 12].

Ideally, jet triggering, finding, and sorting could occur all at once and at the speed of data, reducing the need for a multi-step process and perhaps alleviating data woes from experiments that can produce petabytes of data per year. Since the data is inherently distributional, any such all-in-one approach would require probabilistic tools.

Researchers have recently noted the need for a new probabilistic computing paradigm [1, 20, 17]. The current computing stack is engineered to be deterministic to a fault, with pseudo-randomness injected as an afterthought. As such, random number generation and use can be surprisingly expensive and/or slow. In addition, 10 million data recordings per second is an insanely fast rate. New technology must be developed that can take data at this speed and determine whether or not to save it based on distributions of interest. A fast, all-in-one, jet detector, finder, and classifier is an ideal application for a future probabilistic computer. Should a researcher only be interested in gathering gluon data, for instance, a fast probabilistic trigger could save data that comes from a gluon distribution with estimates on match. That is, it could save an identified jet with an estimate of 92% of being a gluon. Such an approach could be realized with appropriate fast triggering along with a probabilistic machine learning solution that leverages a computer that can compute with distributions.

Though, it must be stated that **if a machine learning algorithm cannot identify jets or even identify whether or not a tower belongs to the most prominent, or leading jet, then a probabilistic machine learning approach is doomed to fail**. Without the ability to distinguish jets, no further solution involving distributions will be viable.

Already, the particle colliding community have begun to look toward machine learning approaches for jet finding. One such approach treats jets as images and adds color and intensity based on energy and momentum [15]. Another approach learns a decision tree using experimental data points [11].

Contrasting these approaches, we utilize data measurements like energy, momentum, location, and azimuthal angles without converting them into images. We aim to identify directly, based on these quantities, whether or not a tower belongs to the most prominent jet in an image. That is, we seek a direct machine learning identification approach rather than a decision tree process and without the use of image-converted data.

In the next sections, we describe the data we use and detail our tower classification approach. We then extend our method to identify the entire leading jet given a data point containing a jet.

3. Understanding the Data. Summarizing the previous discussion, jets are produced when two particles collide. Jets are recorded as data from a calorimeter, and these data are often classified as towers based on energy thresholds (see Figure 1.1). Given that a jet occurred and was recorded (jet triggering), the process of jet finding is determining which towers belong to individual jets.

For our application, we target a more specific subtask: identifying which towers belong to the leading, or most prominent, jet. We had two varieties of data for our application. One set of data is true experimental data taken from proton-proton collision experiments at RHIC. The remaining bulk of our data was generated from an identical proton-proton setup using the simulation software PYTHIA [2]. Out of the 22,392 events in our data set, 980 were obtained through collision experiments and the remaining 21,412 through simulation software.

In each case the data is separated into events where towers are recorded. Within each event, towers are listed as belonging to the leading jet. Each tower has specific data listed (see Table 3.1) and in addition, a summary row is listed for the leading jet containing aggregate information for the jet. In using a machine learning solution to identify which towers belong to the leading jet, we focus the bulk of our attention on the tower specific data and less on the aggregate jet information data.

Table 3.1

Tower and Jet Data Description. † indicates that the value for the aggregate jet information is an energy weighted average of tower values. The first three values, zvert, nrec, and m1, are given for an entire data point. The fourth value, m2, is given for each jet within an event. The final three, r, c, and Q, are not present for aggregate jet information.

Label	**Description**	**Units**
zvert	z-vertex of entire event	cm
nrec	number of towers and jets in the event	
m1	leading jet number	
m2	ID of parton type	
px†	x-component of momentum	GeV/c
py†	y-component of momentum	GeV/c
pz†	z-component of momentum	GeV/c
en	energy	GeV
x†	x position	cm
y†	y position	cm
eta†	pseudorapidity, a function of the polar angle	
phi	azimuthal angle	
modc	calorimeter mode 1 = Electromagnetic 2 = Hadron	
r	tower row number	
c	tower column number	
Q	tower analog-digital-converter value	GeV

For data visualization, we conducted exploratory data analysis using reports generated by the **Pandas Profiling** module in Python [8]. We were able to observe the variable types, distributions, counts, correlation matrices, and verify that there should be no missing data due to initial data cleaning. The histograms generated by the module helped us compare the observed range and distributions of our variables with their theoretical behaviors based on physics knowledge. In Figure 3.1, we see a histogram that shows the multimodal distribution of the variable phi. Histograms of other variables like x and y, which are continuous quantities showed discrete behaviors since they were discretized and binned during collection. From context, we treated these as continuous variables, though it is critical to note that the correlation analysis could differ if we treated them as categorical. In subsection 4.1 we discuss different correlation measures and when they are appropriate to use depending on the type of variable. Figure 3.2 shows a heat map of the Phi-K correlation matrix for both discrete and continuous data variables. Observing these correlations validated our understanding of the physics application. It also gave us an initial insight into which variables may be correlated to event accuracy - which we define as the proportion of towers in the test set that were correctly classified and will be further addressed in Section 4. We can see that the heat map shows a slight correlation between event accuracy and the leading jet number (variable m1).

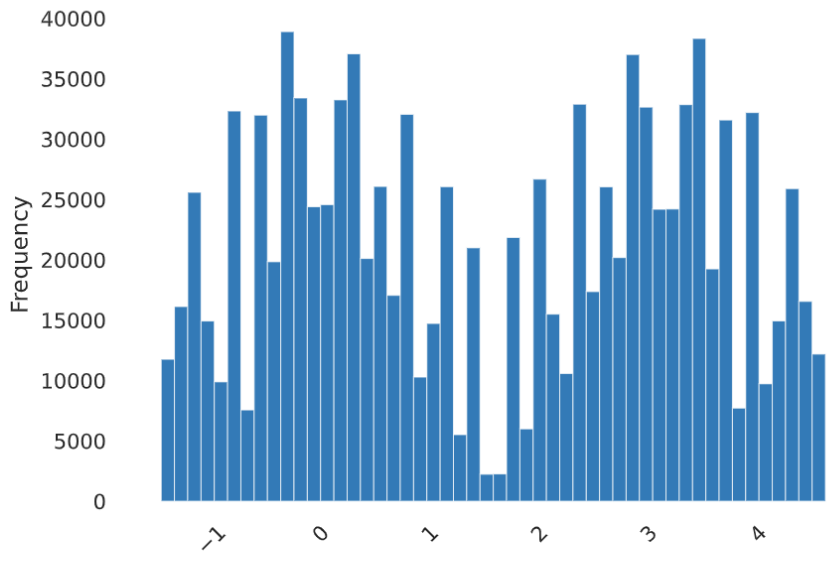


Fig. 3.1. *Histogram showing the frequency distribution of the variable phi (azimuthal angle), we can see that phi appears to have a multimodal distribution. Our data contained values of phi between -1.4657 and 4.6073.*

This report also generated counts for categorical variables, which we were able to compare with the knowledge we have about experimental setup. In performing this comparison, we discovered only 142 towers were measured with an electromagnetic calorimeter, while 1,075,229 towers were measured by a hadron calorimeter. The experimental setup had supposedly removed the electromagnetic calorimeter away from the jet production region, so the existence of data on the electromagnetic calorimeter caused concern that those particular runs were faulty. Hence, the data collected by the electromagnetic calorimeter should be omitted. Although it was not removed for this analysis, it will be for future analyses.

4. Tower Classification and Prediction Analysis. The ultimate goal of our application is to intelligently identify data containing jets (jet triggering) and to save the jet data (jet finding). So, the task at hand is to use machine learning to identify which towers belong to the leading jet given a data event that contains a jet. We approach this task by designing a recurrent neural network (RNN) model to determine whether each tower belongs to the leading jet. All variables in the tower data set are initially fed as predictors into the RNN model. The RNN design is the result of a hyperparameter optimization process which resulted in a 78-dimensional embedding layer followed by 5 bidirectional LSTM units [13].

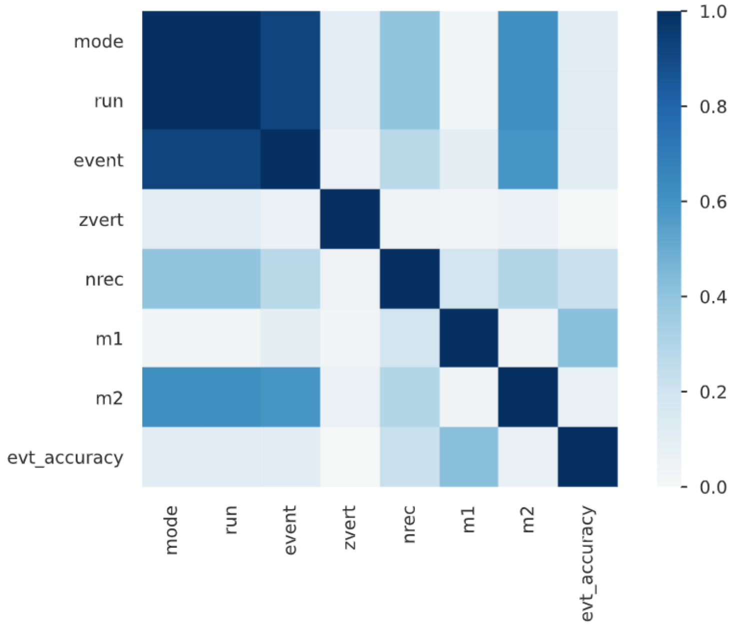


Fig. 3.2. *Heatmap of the Phi-K correlation matrix for the event data variables. We can reference Table 3.1 for variable definitions.*

The model used 2.3 million parameters, and increasing parameter efficiency is the subject of on-going work. See Fig. 4.1 for a diagram of our RNN.

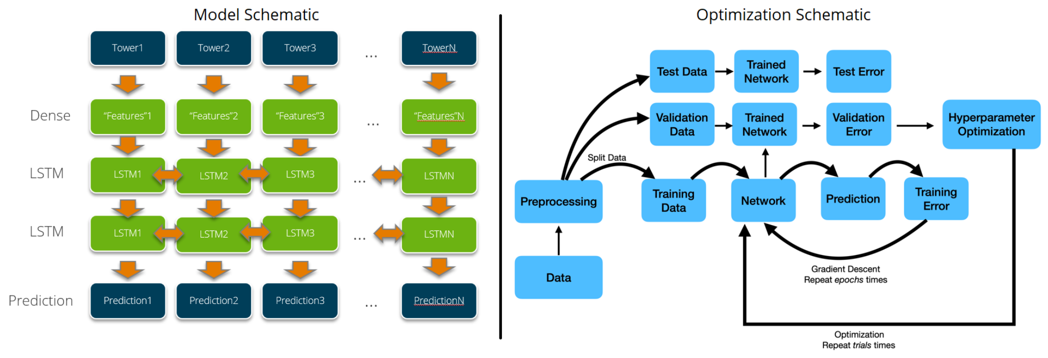


Fig. 4.1. *RNN model schematic. Data for towers within an event are fed into a dense layer with Gaussian error activation functions. These pass into 5 bidirectional LSTM units (2 pictured) to produce a prediction on whether or not towers belong to a leading jet. Model optimization follows a standard procedure, though optimization and parameter efficiency are the subject of an on-going investigation.*

As is standard, our model training method requires both a training (approximately 80% of all data) and a validation set (10%). Once we have a trained model, we evaluate accuracy of prediction using the test set (10%). This test set (containing information from 1,075,441 towers) is what we used to conduct the different analyses in this section.

An initial look at our prediction accuracy shows that 98.6% of the towers in the test set were correctly classified. In Table 4.1, we can see a comparison of accuracy between experimental data, which is obtained by colliding the particles, and simulated data, obtained using the PYTHIA software. We can see that simulated data has a slightly higher accuracy percentage than experimental data. Later in this analysis we will see statistical tests indicating that the mode of data generation does in fact appear to have an association with prediction accuracy. The percentage of true positives, sensitivity, appears to have the smallest difference between experimental and simulated data. Conversely, the percentage of true negatives, specificity, has the largest difference. This suggests that our classification algorithm has a slightly higher chance of falsely predicting that a tower belongs to the leading jet, for experimental data compared to simulated data.

Table 4.1

In this side-by-side comparison of accuracy, sensitivity, and specificity for experimental and simulated data, we see that simulated data has a slightly higher values for all three measures.

	Experimental	Simulated
Accuracy	97.3%	98.7%
Sensitivity	98.2%	98.9%
Specificity	95.4%	98.2%

In the next subsections we will evaluate the relationship between tower variables and prediction accuracy, and test the distribution and association between the correctly and incorrectly predicted subsets of the data.

4.1. Correlations and Associations. In order to better assess correlations in data with mixed types of variables, it is important to use certain measures when appropriate. In Table 4.2 we list the different tools that were considered in this analysis, the types of variables they are appropriate for, and a scope of their purpose.

Table 4.2

Several correlation and association measures were considered when conducting this analysis. This visual shows which types of data each measure is appropriate for and other information to help interpret the values of each.

Name	**Analysis Type**	**Value Range**	**Variable Type**	**Purpose**
Pearson's r	Correlation	$[-1,1]$	Continuous	Measures the strength and direction of linear relationship.
Spearman Rank	Correlation	$[-1,1]$	Continuous	Measures the monotonicity of the relation between two variables.
Φ_k	Correlation	$[0,1]$	Any	Correlation coefficient that also captures non-linear dependency.
Cramer's V	Association	$[0,1]$	Categorical	Measures the strength of association.
Wilcoxon Rank Sum	Test	$(0,1]$	Continuous	Compares distributions and medians.
Point Biserial	Test	$(0,1]$	Continuous vs. Binary	Measures the strength of association.
Pearson χ^2	Test	$(0,1]$	Categorical	Assesses independence.

Based on our correlation analysis and tests of association and independence, we found that there was in fact an association between momentum in the x direction, the mode of data generation (experimental or simulated), and the label number of the leading jet and tower prediction accuracy. Since energy is proportional to momentum, this aligns with physical expectation.

4.2. Using Total Energy as a Measure for Event Accuracy. For this application, we defined event accuracy as the fraction of towers correctly identified as members of the leading jet. However, subject experts may consider other modes of accuracy more informative and relevant to tower classification. Two such measures are: the fraction of total energy correctly predicted, and the difference in the actual average location of an event and the predicted average location of an event. In this work, we explore one of these, namely the fraction of total energy correctly predicted, defined in the equation below.

$$\frac{E_{pred}}{E_{total}} = \frac{\sum_{i \epsilon X'} e_i}{\sum_{j \epsilon X} e_j} \tag{4.1}$$

To define the fraction of total energy correctly predicted, let us first consider the set of events that are known to belong to the leading jet, we denote this as X. Next, we consider the set of events in X that were correctly predicted by the model to belong to the leading jet, and denote this set as X'. In equation (4.1), we can see that the fraction of total energy correctly predicted is the ratio of the corresponding sum of energies for the events in each of these two sets.

Using this as our new target for accuracy, we can see that the more towers that are accurately classified, the closer our running sum of energy gets to the actual total energy value. Also taking into consideration that some towers have higher energies than others, we can then see how it would be more important to accurately predict the membership of these higher-energy towers—since they make up a higher percentage of the total event energy. This differs from our previous definition of accuracy, which placed an equal importance on correctly classifying each tower.

Using this alternate definition of accuracy, we looked at the simple differences and squared differences in an event's actual energy value and its predicted energy value. Looking at a histogram and box plot (see Figure 4.2a) of the simple differences we can see that they are symmetrically distributed with a mean of zero. We were able to see that 99% of events had simple differences between −15.8 and 13.9 GeVs. Looking at a histogram and box plot of the squared differences we can see that they follow a skewed distribution with a long, narrow tail to the right. In the box plot 4.2b, we can see that after the tail, there are nine events with much higher squared error than the rest. If we look at the scatter plot of squared error vs event energy, we can see the same nine outlying observations with much higher squared error than the rest. However, it is notable that it does not appear that they have a higher squared error due to overall higher event energy values. The event energy values for these nine observations range from about 60 to 110 units, which is a typical range for total event energy. In future analysis, we will characterize these observations and see how this new knowledge relates to our existing knowledge of the physics application.

5. Conclusion. Conducting exploratory data analysis and a correlation assessment allowed us to verify known relationships among variables in our physics application, check for bugs in our data collection process, and identify relationships that we would be interested in exploring further. This knowledge will help shape the direction of our future areas of analysis.

Going forward, we will retrain the model after excluding the events captured by the electromagnetic calorimeter to see the effects on the tower prediction accuracy of our experimental data. Since the amount of such data was small relative to the full data set, removing the data might not make a measurable difference. However, it is also possible that since all such observations are part of the experimental data, it may improve the overall accuracy values for the experimental data.

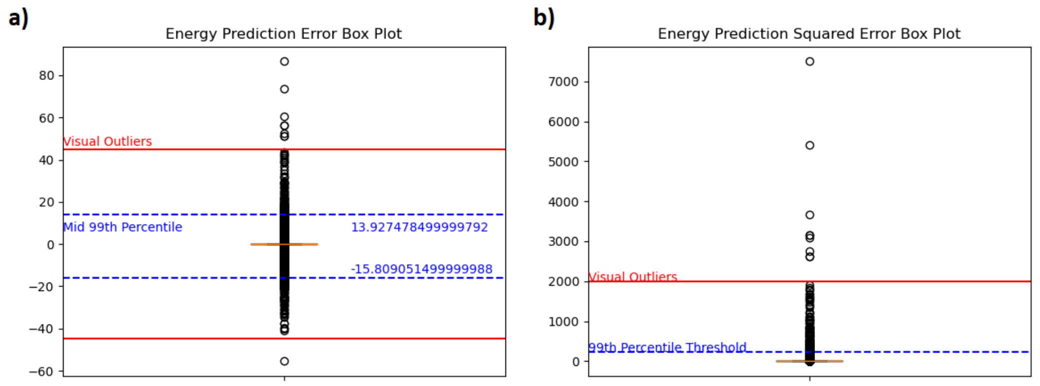


Fig. 4.2. ***a)*** *This box plot of energy prediction error shows the simple error is approximately normally distributed with a mean at 0 and labels the 99th percentile bounds.* ***b)*** *This box plot of squared energy prediction error correspondingly shows a skewed distribution with a clear gap between the tail values and the observations labeled as visual outliers.*

In the future, we seek to measure accuracy based on the average location of a jet. Changing the definition of accuracy may create a different set of outliers. We will be looking to see which events are outlying in this measure – and how their characteristics compare to those of the events outlying by total event energy. It is possible that some types of jets may be more susceptible to misclassification based on the type of accuracy used. For example, it could be that hadrons are identifiable from gluons in this context. Once these experiments are complete, we will replicate the analysis with a data set very similar to, but about twenty times larger than, the one used in this study. Given that the current experimental data set is small compared to the simulated set, further analysis on new data will accelerate understanding and insight on critical jet finding tasks.

Acknowledgment. This work was made possible through the support of the Department of Energy Office of Science (DOE/SC) Co-Design for Microelectronics Project: CO-designed Improved Neural Foundations Leveraging Inherent Physics Stochasticity – COIN-FLIPS.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, J. M. Martinis, G. Finocchio, and K. Y. Camsari, *Massively parallel probabilistic computing with sparse ising machines*, Nature Electronics, (2022), pp. 1–9.

[2] C. Bierlich, N. Desai, B. Road, L. Gellersen, I. Helenius, P. Ilten, L. Lönnblad, S. Mrenna, S. Prestel, C. Preuss, T. Sjöstrand, P. Skands, M. Utheim, and V. Rob, *Pythia*. `https://pythia.org/`.

[3] F. Bieser, H. Crawford, J. Engelage, G. Eppley, L. Greiner, E. Judd, S. Klein, F. Meissner, R. Minor, Z. Milosevich, et al., *The star trigger*, Nuclear Instruments and Methods in

Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 499 (2003), pp. 766–777.

[4] L. Bland, *Spin physics at rhic*, in AIP Conference Proceedings, vol. 675, American Institute of Physics, 2003, pp. 98–111.

[5] L. Bland, E. Brash, H. Crawford, A. Derevschikov, K. Drees, J. Engelage, C. Folz, M. Jones, E. Judd, X. Li, et al., *Cross sections and transverse single-spin asymmetries in forward jet production from proton collisions at s= 500 gev*, Physics Letters B, 750 (2015), pp. 660–665.

[6] L. Bland, E. Brash, H. Crawford, A. Derevschikov, K. Drees, J. Engelage, C. Folz, E. Judd, X. Li, N. Minaev, et al., *Observation of feynman scaling violations and evidence for a new resonance at rhic*, arXiv preprint arXiv:1909.03124, (2019).

[7] L. Blanda, H. Crawfordb, and A. Quinteroc, *Forward dijets at rhic relevant details.*

[8] S. Brugman, *Pandas profiling.* `https://pandas-profiling.ydata.ai/docs/master/index.html`.

[9] M. Cacciari, G. P. Salam, and G. Soyez, *The anti-kt jet clustering algorithm*, Journal of High Energy Physics, 2008 (2008), p. 063.

[10] C. Campagnari and M. Franklin, *The discovery of the top quark*, Reviews of Modern Physics, 69 (1997), p. 137.

[11] M. Draguet, *Machine Learning in Particle Physics Quark versus Gluon Jet Discrimination*, PhD thesis, University of Oxford (GB).

[12] J. Gallicchio and M. D. Schwartz, *Pure samples of quark and gluon jets at the lhc*, Journal of High Energy Physics, 2011 (2011), pp. 1–19.

[13] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.

[14] R. L. Jaffe, *Where does the proton really get its spin?*, Physics today, 48 (1995), pp. 24–33.

[15] P. T. Komiske, E. M. Metodiev, and M. D. Schwartz, *Deep learning in color: towards automated quark/gluon jet discrimination*, Journal of High Energy Physics, 2017 (2017), pp. 1–23.

[16] K.-F. Liu, *From nuclear structure to nucleon structure*, Nuclear Physics A, 928 (2014), pp. 99–109.

[17] S. Misra, L. Bland, S. Cardwell, J. Incorvia, C. James, A. Kent, C. Schuman, J. Smith, and A. JB, *Probabilistic neural computing with stochastic devices*, Advanced Materials (In Press), (2022).

[18] W. Panofsky, *Evolution of particle accelerators*, SLAC Beam Line, 27 (1997), pp. 36–44.

[19] J. Pumplin, *How to tell quark jets from gluon jets*, Physical Review D, 44 (1991), p. 2025.

[20] L. Rehm, C. C. M. Capriata, M. Shashank, J. D. Smith, M. Pinarbasi, B. G. Malm, and A. D. Kent, *Stochastic magnetic actuated random transducer devices based on perpendicular magnetic tunnel junctions*, arXiv preprint arXiv:2209.01480, (2022).

[21] M. Riordan and W. A. Zajc, *The first few microseconds*, Scientific American, 294 (2006), pp. 34A–41.

[22] E. Shuryak, *What rhic experiments and theory tell us about properties of quark–gluon plasma?*, Nuclear Physics A, 750 (2005), pp. 64–83.

PREDICTING VAPOR-LIQUID EQUILIBRIA AND PHASE TRANSITIONS WITH MACHINE-LEARNED INTERATOMIC POTENTIALS

D. G. SEMA* AND M. A. WOOD†

Abstract. A long-standing goal in classical molecular dynamics is to achieve 'transferability' of an interatomic potential, meaning the model can remain accurate in out-of-domain applications and reproduce physical properties over a wide range of phase (ρ,T) space. With the increasing popularity and promise of machine learned interatomic potentials (MLP), this goal is finally within reach and has been recently demonstrated in extreme environments of carbon and iron. The present work has trained an E(3) equivariant deep learning model for Al over a wide range of states ($0.2-3.0g/cm^3$ and $933-10,000K$). We demonstrate the data efficiency of the equivariant atomic representations by reaching quantum-mechanical accuracy in a small fraction of the dataset and then perform large-scale molecular dynamics simulations to predict the vapor-liquid phase equilibrium. Predictions of the vapor dome critical point, as well as critical exponents of this spontaneous phase transition are discussed with respect to model uncertainties. Finally, we focus on the out-of-domain liquid-solid phase transition and demonstrate the efficacy of these models as truly predictive simulations.

1. Introduction.

Understanding the atomistic structure-properties relation of materials has been an integral part in computational material science and has accelerated the advancement of numerous research fields. High-throughput drug discovery, protein folding, and high technology applications like modeling nuclear fusion reactors and designing high entropy alloys are some areas that have benefited from accurate modeling of interatomic interactions [1–3].

Quantum mechanical modeling of molecules and materials rests on the shoulders of electronic structure computations, commonly carried out within the framework of Density Functional Theory (DFT). Despite the widespread use of DFT, these first-principle modeling methods are limited to calculations of hundreds of atoms and short time scales making large scale fist-principles molecular dynamics simulations of functional materials intractable [4]. On the other hand, interatomic potentials have been commonly based on empirical representations of chemical bonds with complex functional forms that are tedious to parameterize and require extensive domain-level expertise to develop (ReaxFF, MEAM, COMB) [5–7].

In recent years, machine learning has emerged as a promising approach to create interatomic potentials from predominantly data-driven and material agnostic approaches that bridge length and time scales while still maintaining quantum-accurate interactions. Kernel based models like the Gaussian Process Regression (GAP) [8], feed-forward neural networks (DeepMD) [9] and descriptor based models, most notably the spectral neighbor analysis potential (SNAP) [10] and the atomic cluster expansion (ACE) [11] are built on input features that have permutational, rotational and translational invariance. These ML potentials (MLP) have been developed rapidly and demonstrated to reproduce physically meaningful atomic interactions and dynamics [12–14]. More recently, the E(3)-equivariant message-passing graph neural networks (MPGNNs) and deep neural networks (NequIP, Allegro, GemNet, BOTNet, MACE) [15–19] aim to encode the local atomic environment with both invariant and equivariant feature representations and have shown great promise to achieve quantum chemical accuracy and transferability [15, 16, 18, 19]. In the present work, we train an equivariant deep learning model for Al using Allegro and demonstrate the data efficiency of these machine-learned interatomic potentials (MLP) by reaching chemical accuracy by only utilizing a small fraction of the original dataset. Furthermore, we have performed large

*Massachusetts Institute of Technology, Mechanical Engineering, dsema@mit.edu
†Sandia National Laboratories, mitwood@sandia.gov

scale molecular dynamics that predict the vapor-liquid phase equilibrium (VLE) as well as on out-of-domain applications(those that are not captured in training), like melting and crack formation in slabs.

2. Al Training Set. The Al dataset contained in total of 11529 configurations (each with 256 atoms) and covered a wide range of the phase diagram, $(\rho, T) = (0.2{-}3.0 g/cm^3, 933{-}10,000K)$. The data were sampled from short AIMD trajectories in NVT(canonical ensemble) and divided into groups for ease of optimization. Examples of these groups include liquid-vapor slabs and some bulk crystals created from a face-centered cubic structure based on their density and temperature; there were 11 such groups in total. The calculations were performed in VASP with a kinetic energy cutoff of 500eV and Gamma-point k-sampling using the PBEsol functional and Projector Augmented Wave method (PAW) as pseudopotentials, while maintaining a good balance of computational efficiency and accuracy. [20–23]

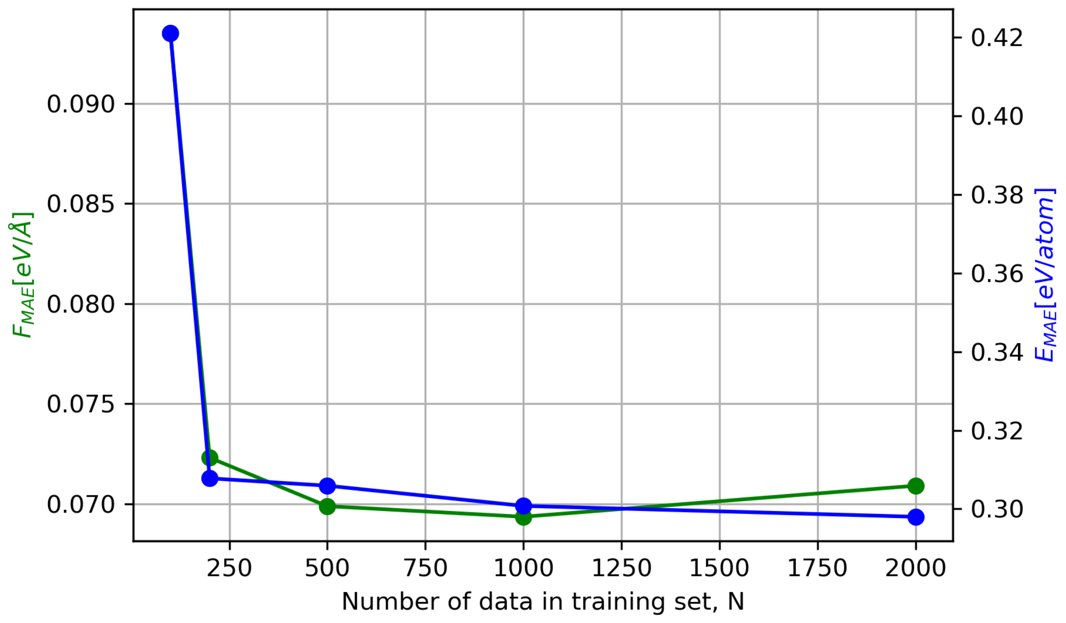


Fig. 3.1: Mean Absolute Error (MAE) for the forces and potential energy of the Al dataset as a function of the number of data in each training split. Series in green correspond to the left vertical axis, while series in blue is plotted against the right vertical axis. The Allegro model saturates at a very small fraction of the available dataset, requiring 200 configurations.

3. E(3)-equivariant Interatomic Potentials. To assess the data efficiency of these networks we will begin by training a few Allegro models with small fractions of the available dataset. All Allegro models were trained using 2 layers (tensor products), 1 feature of even parity and $l_{max} = 1$, this involves scalar and vector features. The 2-body latent MLP consists of 2 hidden layers with depths [16, 32] and uses the SiLU nonlinearity. We note that this nonlinearity is analogous to the use of a nonlinear radial basis in descriptor-based methods like ACE [11]. Allegro models of interatomic interactions are sequentially chained networks, each roughly representing some part of the atomic basis functions utilized in descriptor MLP. A second latent MLP has 1 hidden layer of dimension 32 with a SiLU nonlinearity. The embedding weight projection MLP also has 1 hidden layer of dimension 32 and SiLU as a nonliearity. The final edge energy MLP that converts the final features to the atomic pairwise energies consists of 1 hidden layer of dimension 16 and uses the SiLU nonliearity. Four MLPs are initialized with the Glorot method according to a uniform distribution such that the weights of each layer have the same variance [24]. We select 8 non-trainable Bessel functions with a polynomial envelope function of degree $p = 6$ and a radial local cutoff radius, $r_{cut} = 4.7$Å. The models were trained jointly on energies and forces with an importance ratio of 5%/95% in the loss function respectively. This ratio of loss function wights was chosen after a coarse sampling for model stability in MD[25]. The Adam optimizer was used with a starting learning rate of 2e-3 and a batch size of 1, while the on-plateau scheduler was used to reduce the learning rate based on the validation loss with a patience of 3 epochs and a decay factor of 0.8. [26] The training and evaluation of the potentials was done on a single V100 GPU.

In order to reduce the compute time spent training these MLP, the dataset was split into smaller subsets in the following way. For each split, we randomly sample a fraction of each group of data in order to ensure information from different parts of the phase space is included and create datasets with 100, 200, 500, 1000 and 2000 configurations. We train the Allegro models with each of these subsets with a 90/10% split into training and validation sets. The potentials are then evaluated on the entire dataset of 11529 configurations and the results are presented in figure 3.1. This demonstrates that Allegro is able to capture the physical phase space with a very small number of configurations; in this case we observe that 200 configurations (180 for training and 20 for the validation sets) are enough data to get a converged potential that has reached an acceptable accuracy for MD since the MAE error of 70meV/Å is <1% of the maximum forces that appear in these configurations.

We note that the entire process for the development of this potential, including the generation of the amount of DFT data that is needed as shown from Figure 3.1, training the potential and starting large-scale production molecular dynamic simulations takes less than 24 hours, thus enabling the development of potentials-in-a-day. This metric of efficiency is arbitrary, but it represents a significant speed up in the development of interatomic potentials which for decades have been developed iteratively by domain experts. Alleviating the bottleneck through select research groups for model development and deployment will greatly accelerate insight gained through MD simulations in the near future.

4. Vapor-Liquid Phase Equilibrium Prediction. Using the developed potential, we perform large scale($> 10^6$ atoms ,$> 10^3 ps$) molecular dynamics simulations to predict the vapor-liquid phase equilibrium (VLE) of Al. Of interest are the locations of phase boundaries in this low-density regime of Al as these are states of the material exhibited in the shock-release experiments performed at the Sandia Z-Machine. Mapping these phase boundaries from MD would prove a valuable prediction that experimental investigations can use to guide new experiments and interpret complex instrumentation signals. The peak of the liquid-vapor phase boundary is known as the critical point, and is inaccessible directly from DFT calculations, and difficult to identify from experiments. A set of isothermal DFT simulations can probe the critical point, but are not free of finite size effects. Starting from a density of $\rho = 0.75$, which is chosen as it is close to the experimental critical density, we perform simulations designed to exhibit phase separation in NVT. The starting simulation box consists of $2 \cdot 10^5$ atoms and was prepared by randomly placing atoms such that they have ≥ 2Å distance from each neighbor. This condition was chosen based on the radial distribution function of fcc Al. We start at 2000K with steps of 250K until we reach the

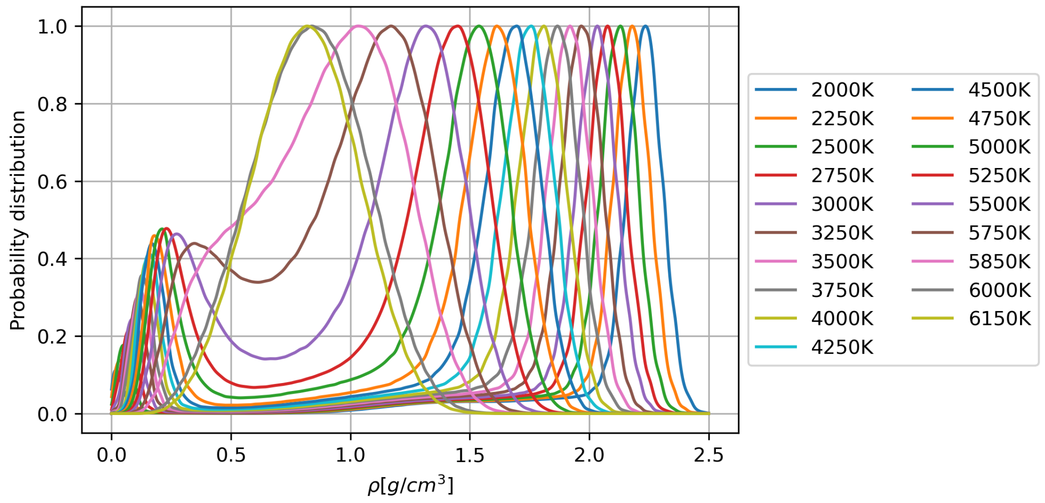


Fig. 4.1: Density profiles of the liquid (right peaks) and vapor (left peaks) phases as probability distributions for different temperatures as predicted from phase separation molecular dynamics simulations using the Allegro model. The critical point is predicted to be $(\rho_c, T_c) = (0.745 g/cm^3, 5978K)$ which is in quantitative agreement with the experimental observations.

conditions at each where the Al has become a supercritical fluid(above the critical point). Figure 4.1 shows the density distributions at various temperature conditions and the distinct peaks corresponds to the either vapor and liquid densities. Figure 4.2, shows the prediction of the liquid-vapor dome in comparison to the EAM potential and linear SNAP. The linear SNAP model created by Sandia collaborators was trained on the entire Al dataset. This heatmap represents the density distributions of the two phases for the Allegro model and showcases the uncertainty of the quantity of interest as we approach the critical point.

The critical point can be estimated from the rectilinear rule and the phase-coexistence method from equations:

$$\rho_l - \rho_g = A(T_c - T)^b \Rightarrow (\rho_l - \rho_g)^{1/b} = A^{1/b}T_c - A^{1/b}T, \tag{4.1}$$

$$\frac{\rho_l + \rho_g}{2} = \rho_c + C(T_c - T) \Rightarrow \frac{\rho_l + \rho_g}{2} = \rho_c + CT_c - CT, \tag{4.2}$$

where A, C are fitting parameters, $b \simeq 0.236$ is the Ising critical exponent of a two state system and (ρ_c, T_c) is the critical point. Using our computational results from 2000-5850K we can estimate the critical point by performing a linear fit from 4.1 and then 4.2 to be at $(\rho_c, T_c) = (0.745 g/cm^3, 5978K)$. From our simulations captured in Figure 4.1 and 4.2 we find the theoretical result to be in quantitative agreement and also within 1% of the experimental critical density and 8% of the experimental critical temperature. These results are encouraging for future use of Allegro models used in MD as experimentally relevant predictions can be made from limited amounts of training data.

5. Solid-Liquid phase transition. Now we turn our attention to the section of the phase diagram that is underrepresented in our training data. We employ our Allegro MLP to

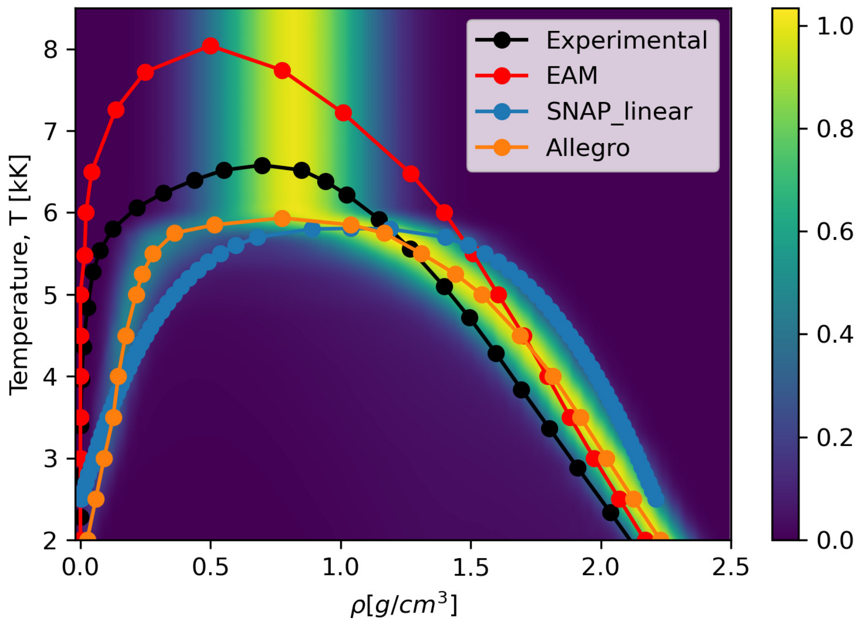


Fig. 4.2: Vapor-Liquid dome predictions for Allegro, linear SNAP and EAM. The heatmap represents the density distributions of the two phases for the Allegro model and showcases the uncertainty of the quantity of interest as we approach the critical point. Both Allegro (E(3)-equivariant deep neural network) and SNAP (bispectrum descriptors) are able to predict the vapor-liquid dome with a critical point closer to the experimental observations than the EAM potential [27].

estimate the melting point of bulk Al at $2.72 g/cm^3$ by performing a simulation with $2.56 \cdot 10^5$ atoms in NVT. We start at room temperature and gradually ramp the temperature past the experimental melting point. These Allegro MD simulations predict a melting point at ≈1754K, which agrees with the experimental phase space (Figure 5.1) [27].

Finally, we perform an NVT simulation where we allow the material to undergo a thermal shock via rapid quenching in order to qualitatively measure the materials physical behaviour predicted from this MLP. We start from an fcc slab with $1.05 \cdot 10^5$ atoms and perform a rapid melt-quench process. During heating the slab melts and start to become amorphous without completely losing its crystalline structure. During the quenching phase, cracks and nanocrystals start to form throughout the material. As the cracks propagate through the slab, the nanocrystals merge into bigger ones and rearrange into an fcc orientation (Figure 5.2).

6. Conclusions. In this work, we developed an E(3)-equivariant deep neural network that is able to predict a wide range of the phase space of Al, including VLE phase equilibrium, the critical point and supercritical fluid dynamics. These are important result in Sandia's effort to provide modeling and simulations capabilities that support the unique pulsed power experiments done onsite. Mapping phase boundaries from MD aides in providing viable equations of state to continuum modeling efforts, furthering our predictive

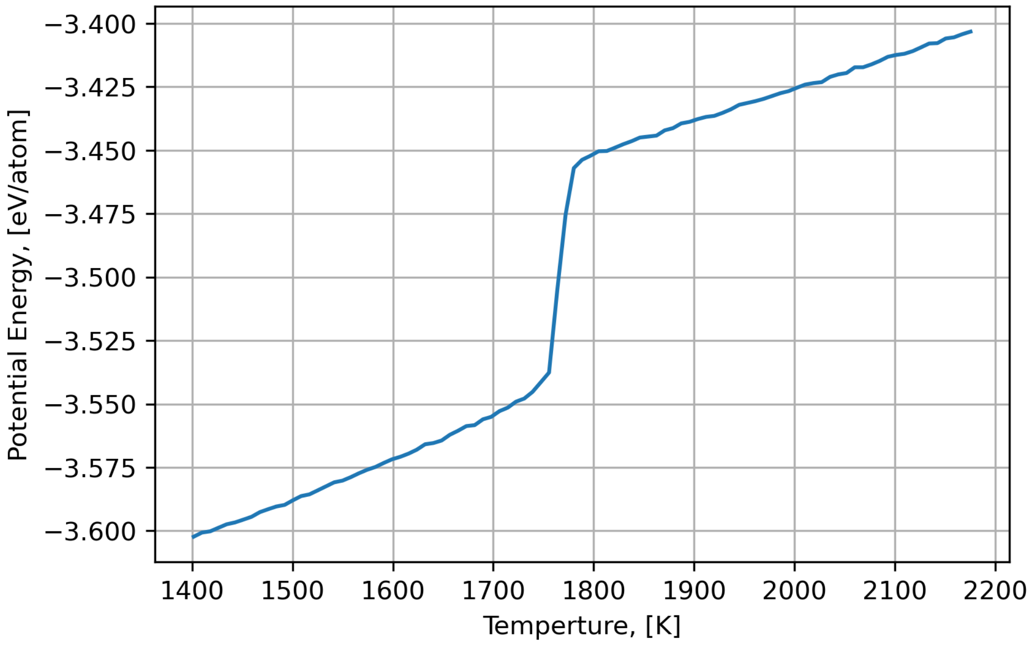


Fig. 5.1: Prediction of the melting point (rapid rise in Potential Energy) of bulk Al at $2.72 g/cm^3$ by performing an NVT simulation with 256k atoms. The fcc Al(s) transitions into an amorphous liquid at ≈1754K.

capability via multi-scale modeling. In addition, we showed that Allegro can predict solid-liquid phase transitions as well as crack formation, propagation and surface reconstruction in slabs with physically meaningful characteristics.

The data efficiency and chemical accuracy of these MLPs provide a significant milestone in developing potentials for materials in a day, that have the capability to predict dynamics and physical quantities of interest in extreme environments where experimental predictions are inaccessible. Accelerating this critical model development step will enable broadly reaching simulation predictions, and bring valuable comparisons between experiment and theory efforts.

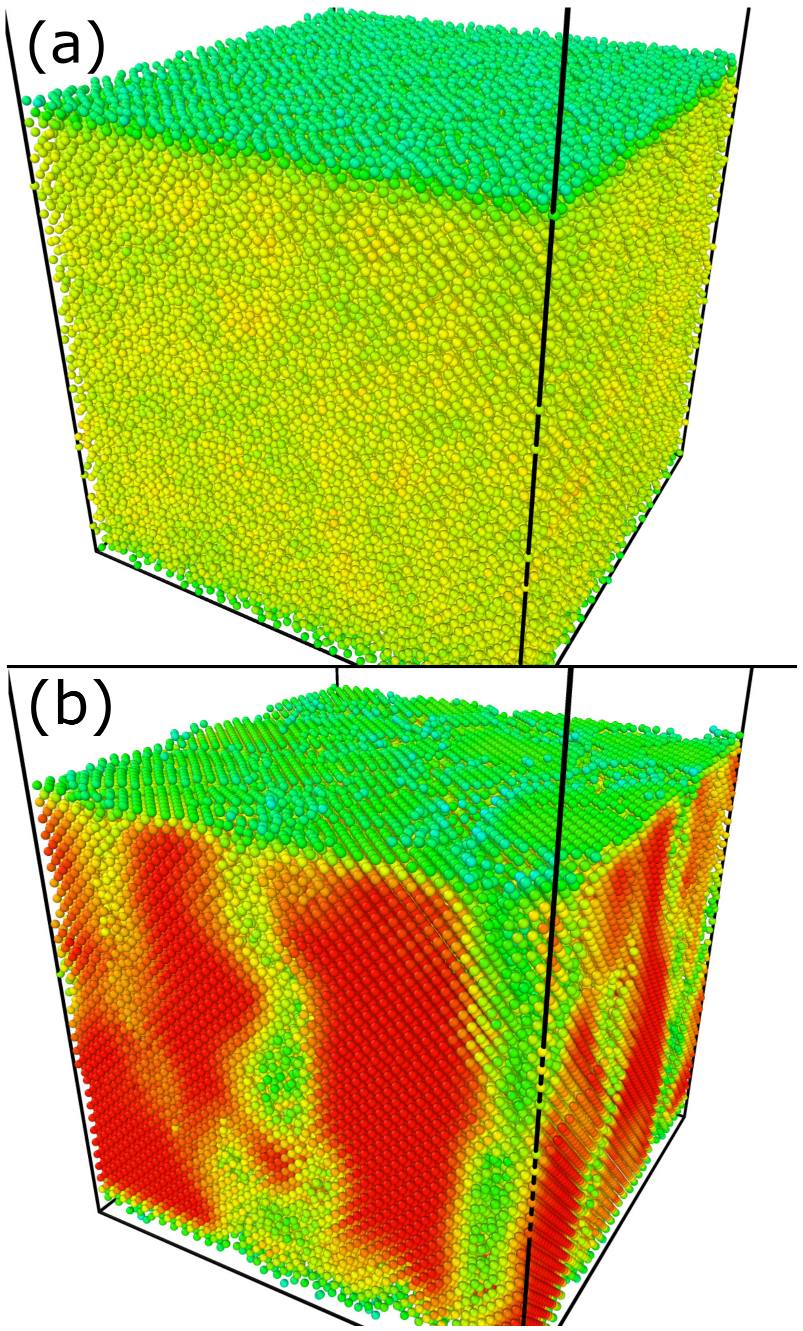


Fig. 5.2: Crack formation and propagation during a melt-quench process of a slab. (a) The Al becomes an amorphous liquid and (b) during the quenching process, the slab begins to crystallize leading to the formation of nanocrystals that merge and rearrange into fcc structures.

Acknowledgements. I sincerely want to thank Mitch Wood for his mentorship and his valuable insight and support on the project. I also want to thank Aidan Thompson, David Montes, Ember Sikorski, Megan McCarthy, James Goff and Drew Rohskopf for their useful discussions and meetings.

References.

[1] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[2] H. J. Kulik, "Making machine learning a useful tool in the accelerated discovery of transition metal complexes," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 10, no. 1, p. e1439, 2020.

[3] L. Chanussot, A. Das, S. Goyal, T. Lavril, M. Shuaibi, M. Riviere, K. Tran, J. Heras-Domingo, C. Ho, W. Hu *et al.*, "Open catalyst 2020 (oc20) dataset and community challenges," *ACS Catalysis*, vol. 11, no. 10, pp. 6059–6072, 2021.

[4] A. J. Cohen, P. Mori-Sánchez, and W. Yang, "Challenges for density functional theory," *Chemical Reviews*, vol. 112, no. 1, pp. 289–320, 2012.

[5] A. C. Van Duin, S. Dasgupta, F. Lorant, and W. A. Goddard, "Reaxff: a reactive force field for hydrocarbons," *The Journal of Physical Chemistry A*, vol. 105, no. 41, pp. 9396–9409, 2001.

[6] M. S. Daw, S. M. Foiles, and M. I. Baskes, "The embedded-atom method: a review of theory and applications," *Materials Science Reports*, vol. 9, no. 7-8, pp. 251–310, 1993.

[7] K. Choudhary, T. Liang, A. Chernatynskiy, Z. Lu, A. Goyal, S. R. Phillpot, and S. B. Sinnott, "Charge optimized many-body potential for aluminum," *Journal of Physics: Condensed Matter*, vol. 27, no. 1, p. 015003, 2014.

[8] G. Sivaraman, A. N. Krishnamoorthy, M. Baur, C. Holm, M. Stan, G. Csányi, C. Benmore, and Á. Vázquez-Mayagoitia, "Machine-learned interatomic potentials by active learning: amorphous and liquid hafnium dioxide," *npj Computational Materials*, vol. 6, no. 1, pp. 1–8, 2020.

[9] L. Zhang, J. Han, H. Wang, R. Car, and E. Weinan, "Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics," *Physical review Letters*, vol. 120, no. 14, p. 143001, 2018.

[10] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker, "Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials," *Journal of Computational Physics*, vol. 285, pp. 316–330, 2015.

[11] R. Drautz, "Atomic cluster expansion for accurate and transferable interatomic potentials," *Physical Review B*, vol. 99, no. 1, p. 014104, Jan. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.99.014104

[12] M. A. Wood and A. P. Thompson, "Extending the accuracy of the snap interatomic potential form," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241721, 2018.

[13] J. T. Willman, A. S. Williams, K. Nguyen-Cong, A. P. Thompson, M. A. Wood, A. B. Belonoshko, and I. I. Oleynik, "Quantum accurate snap carbon potential for md shock simulations," in *AIP Conference Proceedings*, vol. 2272, no. 1. AIP Publishing LLC, 2020, p. 070055.

[14] D. P. Kovács, C. v. d. Oord, J. Kucera, A. E. Allen, D. J. Cole, C. Ortner, and G. Csányi, "Linear atomic cluster expansion force fields for organic molecules: beyond rmse," *Journal of Chemical Theory and Computation*, vol. 17, no. 12, pp. 7696–7711, 2021.

[15] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari,

T. E. Smidt, and B. Kozinsky, "E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials," *Nature Communications*, vol. 13, no. 1, pp. 1–11, 2022.

[16] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, "Learning local equivariant representations for large-scale atomistic dynamics," *arXiv preprint arXiv:2204.05249*, 2022.

[17] J. Gasteiger, F. Becker, and S. Günnemann, "Gemnet: Universal directional graph neural networks for molecules," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6790–6802, 2021.

[18] I. Batatia, S. Batzner, D. P. Kovács, A. Musaelian, G. N. Simm, R. Drautz, C. Ortner, B. Kozinsky, and G. Csányi, "The design space of e (3)-equivariant atom-centered interatomic potentials," *arXiv preprint arXiv:2205.06643*, 2022.

[19] I. Batatia, D. P. Kovács, G. N. Simm, C. Ortner, and G. Csányi, "Mace: Higher order equivariant message passing neural networks for fast and accurate force fields," *arXiv preprint arXiv:2206.07697*, 2022.

[20] G. Kresse and J. Hafner, "Ab initio molecular dynamics for liquid metals," *Phys. Rev. B*, vol. 47, no. 1, p. 558, 1993.

[21] G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set," *Phys. Rev. B*, vol. 54, no. 16, p. 11169, 1996.

[22] P. E. Blöchl, "Projector augmented-wave method," *Phys. Rev. B*, vol. 50, no. 24, p. 17953, 1994.

[23] G. Kresse and D. Joubert, "From ultrasoft pseudopotentials to the projector augmented-wave method," *Phys. Rev. B*, vol. 59, no. 3, p. 1758, 1999.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[25] X. Fu, Z. Wu, W. Wang, T. Xie, S. Keten, R. Gomez-Bombarelli, and T. Jaakkola, "Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations," *arXiv preprint arXiv:2210.07237*, 2022.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] M. E. Povarnitsyn, V. B. Fokin, P. R. Levashov, and T. E. Itina, "Molecular dynamics simulation of subpicosecond double-pulse laser ablation of metals," *Physical Review B*, vol. 92, no. 17, p. 174104, 2015.

THERMODYNAMIC CONSISTENCY OF PHYSICS-DYNAMICS COUPLING IN THE HOMME ATMOSPHERE DYNAMICAL CORE

JASON L. TORCHINSKY*, MARK A. TAYLOR†, AND OKSANA GUBA‡

Abstract. Global atmosphere models depend on physical parametrizations to capture sub-grid-scale processes, such as the formation of clouds and precipitation. These parametrizations are typically developed independently and later integrated into the global atmosphere model. It is easy to introduce errors during this integration process; for example, a cloud microphysics parametrization implicitly assumes that thermodynamic processes occur at constant-pressure while the global climate model assumes they occur at constant-volume. We develop a general interface that will correctly support both constant-volume thermodynamics in a vertical height coordinate and constant-pressure thermodynamics in a vertical pressure coordinate for the High-Order Methods Modelling Environment. We compare six combinations of physics interfaces and vertical coordinates using a tropical cyclone test case.

1. Introduction.

Modelling the Earth's atmosphere is an immense task, both mathematically and computationally. Consider the evolution of clouds. There has been great effort put into understanding how water condenses onto aerosol particles, how those droplets grow and/or freeze, and eventually fall to the ground. (e.g., [11]) Such a particle-based approach is not feasible for global atmosphere models, whose resolution is on the order of kilometers, not microns. (e.g., [4]) Many atmosphere models instead utilize a bulk parametrization of sub-grid-scale processes. These parametrizations, referred to as the 'physics', are typically formulated and developed independently. They are later coupled to the atmosphere model, referred to as the 'dynamics'. Typically, the state is updated according to the dynamics (the 'dynamics-update') and then according to the physics (the 'physics-update').

Errors may arise when the presumptions of the physics differ from those of the dynamics. For example, when an atmosphere model uses a vertical height coordinate, each grid cell should have constant volume during the physics-update. The Reed-Jablonowski cloud microphysics parametrization assumes that the thermodynamic processes occur in a constant-pressure environment [10]. Hence, the Reed-Jablonowski parametrization should not be used with an atmosphere model that uses a vertical height coordinate without further interventions.

The primary purpose of this paper is to investigate the impact of mismatched physics and dynamics. We provide the mathematical impetus for the significance of such a mismatch in hydrostatic models (Section 2), which has not been presented before. We demonstrate the significance by comparing the current physics parametrization used in the High-Order Methods Modeling Environment (HOMME) with novel constant-pressure and constant-volume interfaces (Section 3). To make the comparison, we analyze their impact on a tropical cyclone test case (Section 4). The development and comparison of these new interfaces are one of our primary contributions.

2. Conservation of Specific Local Energy.

In this section, we show that the conservation of specific local energy of an air parcel is different between isobaric (constant-pressure; Subsection 2.1) and isochoric (constant-volume; Subsection 2.2) processes, and quantify this difference (Subsection 2.3).

The specific local energy E of an air parcel in a hydrostatic environment is given by the

*University of Wisconsin-Madison, Department of Mathematics, jason.torchinsky@wisc.edu, also funded by the DOE Computational Science Graduate Fellowship DE-SC0020347

†Sandia National Laboratories, mataylor@sandia.gov

‡Sandia National Laboratories, mataylor@sandia.gov

sum of its kinetic energy K, potential energy P, and internal energy U

$$\begin{aligned} E &= K + P + U \\ &= \frac{1}{2}\,\vec{u}\cdot\vec{u} + \phi + (H - p\,\alpha)\,. \end{aligned} \tag{2.1}$$

Here, $\vec{u}$ is the horizontal velocity of the air parcel, ϕ its geopotential height, H its enthalpy, p its pressure, and α its specific volume.

The enthalpy is given by the sum of the heat energies associated with: (i) the difference between the temperature of the air parcel T and a reference temperature T_r and (ii) the latent heat release of water vapor condensing and freezing,

$$H = \underbrace{c_p^*\,(T - T_r)}_{(i)} + \underbrace{(L_{vr} + L_{fr})\,q_v + L_{fr}\,q_l}_{(ii)} + \underbrace{q_d\,R_d\,T_r}_{(iii)}\,. \tag{2.2}$$

The (iii) term simply shifts the zero of H. The thermodynamic constants are the specific heat of the air parcel at constant pressure c_p^*, the gas constant for dry air R_d, the latent heat of vaporization L_{vr} at T_r, and the latent heat of freezing L_{fr} at T_r. Further, q_d, q_v, and q_l are the ratios of the mass of dry air, water vapor, and liquid water, respectively, to the total mass of moist air in the parcel [2].

We consider two vertical coordinates: a hydrostatic pressure coordinate π and a height coordinate z. These vertical coordinates are related by the hydrostatic relation

$$\frac{\partial \pi}{\partial z} = -\frac{g}{\alpha}, \tag{2.3}$$

where g is the acceleration due to gravity.

Using the hydrostatic relation, we may obtain the specific local energy of an entire layer of air E^{tot} in either vertical coordinate

$$E^{tot} = \int_{z_1}^{z_2} \frac{E}{\alpha}\,dz = -\frac{1}{g}\int_{\pi_1}^{\pi_2} E\,d\pi. \tag{2.4}$$

To simplify our calculations for the conservation of specific local energy in isobaric (Subsection 2.1) and isochoric (Subsection 2.2) processes, we make the following assumptions:

i) the air parcel is in a hydrostatic environment, i.e., the total pressure p is equal to the hydrostatic pressure π;
ii) the effect of the layer expansion on the kinetic energy of each parcel is negligible, and;
iii) the pressure-volume work performed by a cell goes into lifting the cells above it. In an isobaric process, the work done by the cell W is equal to

$$W = -\frac{1}{g}\,\frac{\partial\,[\pi\,\phi]}{\partial \pi}. \tag{2.5}$$

We obtain this equation by calculating the change in potential energy of a cell undergoing an isobaric process, which is the integral of the work done on the cell

$$\int_{\pi_1}^{\pi_2} \delta W\,d\pi = \delta P = -\frac{1}{g}\,(\pi_{top}\,\delta\phi_{top} - \pi_{bot}\,\delta\phi_{bot})\,. \tag{2.6}$$

2.1. In an Isobaric Process. In an isobaric (constant-pressure) process, the change in specific local energy of an air parcel is due to the pressure-volume work performed and the external heating δQ.

To ease our calculations, we rewrite the specific local energy (Eqn. 2.1) using the identity $d\,[\pi\,\phi] = \pi\,d\phi + \phi\,d\pi$

$$\begin{aligned} E &= \frac{1}{2}\,\vec{u}\cdot\vec{u} + \phi + (H - p\,\alpha) \\ &= \frac{1}{2}\,\vec{u}\cdot\vec{u} - g\,W + H. \end{aligned} \tag{2.7}$$

With this, we see that the change in energy δE for an isobaric process is given by

$$\left.\delta E\right|_p = 0 - g\ \left.\delta W\right|_p + \left.\delta H\right|_p. \tag{2.8}$$

The first law of thermodynamics states that

$$\left.\delta E\right|_p = -g\ \left.\delta W\right|_p + \left.\delta Q\right|_p. \tag{2.9}$$

Using the definition of enthalpy (Eqn. 2.2), we obtain the isobaric specific local energy conservation equation

$$c_p^*\ \left.\delta T\right|_p + (L_{vr} + L_{fr})\ \left.\delta q_v\right|_p + L_{fr}\ \left.\delta q_l\right|_p = \left.\delta Q\right|_p. \tag{2.10}$$

2.2. In an Isochoric Process. In an isochoric (constant-volume) process, an air parcel performs zero pressure-volume work, and so the change in specific local energy is due entirely to the external heating.

To obtain the equation for conservation of energy of an isochoric process, we note the equation of state

$$p\,\alpha = R^*\,T \tag{2.11}$$

and the relation between the specific heat at constant pressure with the specific heat at constant volume c_v^*

$$c_v^* = R^* - c_p^*, \tag{2.12}$$

where R^* is the gas constant for the air parcel [2].

Using these, we rewrite the specific energy as

$$E = \frac{1}{2}\,\vec{u}\cdot\vec{u} + \phi + c_v^*\,T + \left(q_d\,R_d - c_p^*\right)\,T_r + (L_{vr} + L_{fr})\,q_v + L_{fr}\,q_l. \tag{2.13}$$

In an isochoric process, both α and the thickness of the layer are constant in time. Therefore, the change in specific local energy of a z-layer of air undergoing an isochoric process is given by

$$\left.\delta E^{tot}\right|_\alpha = \int_{z_1}^{z_2} \frac{1}{\alpha}\ \left.\delta E\right|_\alpha\,dz. \tag{2.14}$$

For specific local energy to be conserved for any arbitrary z-layer, the above integrand must be equal to the external heating applied to the cell. Therefore, assuming that c_v^* is constant, the conservation of specific local energy for an air parcel undergoing an isochoric process is equivalent to

$$c_v^*\ \left.\delta T\right|_\alpha + (L_{vr} + L_{fr})\ \left.\delta q_v\right|_\alpha + L_{fr}\ \left.\delta q_l\right|_\alpha = \left.\delta Q\right|_\alpha. \tag{2.15}$$

2.3. Isobaric Versus Isochoric Energy Conservation. Many physics parametrization schemes assume that the air parcels undergo thermodynamics processes isobarically (constant-pressure) [5, 6, 7, 10]. This assumption is compatible with a vertical pressure coordinate; in which each grid cell has a constant pressure. However, this assumption is not compatible with a vertical height coordinate; in which the pressure of each grid cell changes and the volume of each grid cell is constant. This naturally leads to the question

> Is there some way to convert an isobaric (constant-pressure) physics parametrization to be isochoric (constant-volume)?

The answer lies in the similarity between the specific local energy conservation relations for isobaric (Eqn. 2.10) and isochoric (Eqn. 2.15) processes.

For simplicity, we assume that the difference in the phase changes of water and external heating between an isobaric and an isochoric process are negligible, i.e.,

$$\delta q_v|_p = \delta q_v|_\alpha \,, \quad \delta q_l|_p = \delta q_l|_\alpha \,, \quad \text{and} \quad \delta Q|_p = \delta Q|_\alpha \,. \tag{2.16}$$

We may then compare Eqn. 2.10 and Eqn. 2.15 to find that

$$c_v^* \;\delta T|_\alpha = c_p^* \;\delta T|_p \,. \tag{2.17}$$

Given a physics parametrization that describes the change in temperature and water content in an air parcel, we use Eqn. 2.17 to convert the prescribed temperature change from that of an isobaric process to that of an isochoric one. For example, we would multiply the isobaric temperature change by $c_p^*/c_v^* \approx 1.4$ to obtain the corresponding isochoric temperature change.

3. Physics Interface Descriptions. Here we outline the three different physics interfaces we qualitatively examine in Section 4:

(i) the physics interface provided by the 2016 Dynamical Core Model Intercomparison Project (DCMIP 2016; [16, 17]). This is currently used by the High-Order Methods Modelling Environment (HOMME) for the DCMIP 2016 test cases. We refer to this as the 'original physics' of HOMME (Subsection 3.1);

(ii) a novel isobaric (constant-pressure) version of (i), which we henceforth refer to as the 'isobaric physics' of HOMME (Subsection 3.2), and;[1]

(iii) a novel isochoric (constant-volume) version of (i), obtained by applying Eqn. 2.17 to the isobaric temperature change given by (ii), which we henceforth refer to as the 'isochoric physics' of HOMME, (Subsection 3.2).[1]

3.1. Original Physics of HOMME. The High-Order Methods Modelling Environment (HOMME) currently utilizes the height vertical coordinate physics interface provided by DCMIP 2016 (`dcmip_physics_z_v1.f90`, available at [17]). In short, the physics interface is composed of three parts:[2]

1. **Large-scale precipitation:** Utilize either the Kessler [5] or the Reed-Jablonowski cloud microphysics parametrization [10] to update the potential temperature and moisture variables (wet mixing ratio of water vapor, cloud droplets, and rain droplets).[3] With these values, calculate the new moist density and pressure.

[1]We refer to isobaric and isochoric versions of the original physical of HOMME as 'novel' as they are new to HOMME. Isobaric or isochoric updates are widely use in models that utilize a pressure vertical coordinate (e.g., [13]) or a height vertical coordinate (e.g., [12]), respectively.

[2]In all tests, we reduce the bulk transfer coefficient for water vapor C_E and the bulk heat transfer coefficient C_H to one-tenth of their standard value. This is to minimize the effect of the surface flux and planetary boundary layer parametrizations, as they are essentially the same for each physics scheme tested here.

[3]In the main body of this work, we utilize only the Kessler microphysics parametrization. We show results for both the Kessler and Reed-Jablonowski microphysics parametrizations in Appendix A.

2. **Surface fluxes:** Utilize the Reed-Jablonowski surface flux physics parametrization [10] to update the surface horizontal wind speed, wet mixing ratio of water vapor, and temperature. Using these values, update the surface dry mixing ratio of water vapor, moist density, pressure, and potential temperature.
3. **Planetary boundary layer:** Utilize the Reed-Jablonowski planetary boundary layer physics parametrization [10] to update the horizontal wind speeds, potential temperature, and wet mixing ratio of water vapor. With these values, update the dry mixing ratio of water vapor, moist density, virtual potential temperature, and pressure.

These new values are used to find the change in horizontal wind speeds, non-hydrostatic pressure perturbation, temperature, and moisture variables.

Both the Kessler microphysics and Reed-Jablonowski physics parametrizations are formulated to be isobaric, meaning that the final temperature update should be isobaric. It is unclear to us that the pressure should be re-calculated throughout the physics update.

3.2. Novel Isobaric and Isochoric Physics of HOMME. As opposed to the original physics of High-Order Methods Modelling Environment (HOMME), in these versions we do not update the pressure while calculating the isobaric (constant-pressure) temperature change and the change in moisture variables (wet mixing ratio of water vapor, cloud droplets, and rain droplets). In the isochoric (constant-volume) version, the isobaric temperature update is converted to an isochoric temperature update using Eqn. 2.17.

We now describe the physics interface common to both the isobaric and isochoric version, italicizing where it differs from the original physics.[2]

1. **Large-scale precipitation:** Utilize either the Kessler [5] or the Reed-Jablonowski cloud microphysics parametrization [10] to update the potential temperature and moisture variables (wet mixing ratio of water vapor, cloud droplets, and rain droplets).[3] *With these values, calculate the new temperature, moist density, as well as the wet and dry mixing ratios of water vapor. The new moist density is used to update the geometric height levels.*
2. **Surface fluxes:** Utilize the Reed-Jablonowski surface flux physics parametrization [10] to update the surface horizontal wind speed, wet mixing ratio of water vapor, and temperature. *Using these values, update the surface wet and dry mixing ratios of water vapor, moist density, and potential temperature. The new moist density is again used to update the geometric height levels.*
3. **Planetary boundary layer:** Utilize the Reed-Jablonowski planetary boundary layer physics parametrization [10] to update the horizontal wind speeds, potential temperature, and wet mixing ratio of water vapor. *With these values, update the wet and dry mixing ratio of water vapor, moist density, virtual potential temperature. The new moist density is used once more to update the geometric height levels.*

The final potential temperature and moisture variables are used to calculate the total change in horizontal wind speeds and moisture variables, as well as the isobaric temperature change. In the isochoric physics interface, the isobaric temperature change is converted to the isochoric temperature change via Eqn. 2.17.

4. Comparison of Physics-Vertical Coordinate Combinations.

To compare the quality of each the original, isobaric, and isochoric physics schemes described in Section 3, we examine their impact on the tropical cyclone test case of the 2016 Dynamical Core Model Intercomparison Project (DCMIP 2016; [16]).

The DCMIP 2016 tropical cyclone test case consists of an analytic vortex initialized on an Earth-size aqua planet in a background environment suited for the growth of tropical cyclones. We include example output of the surface pressure using the original physics of the

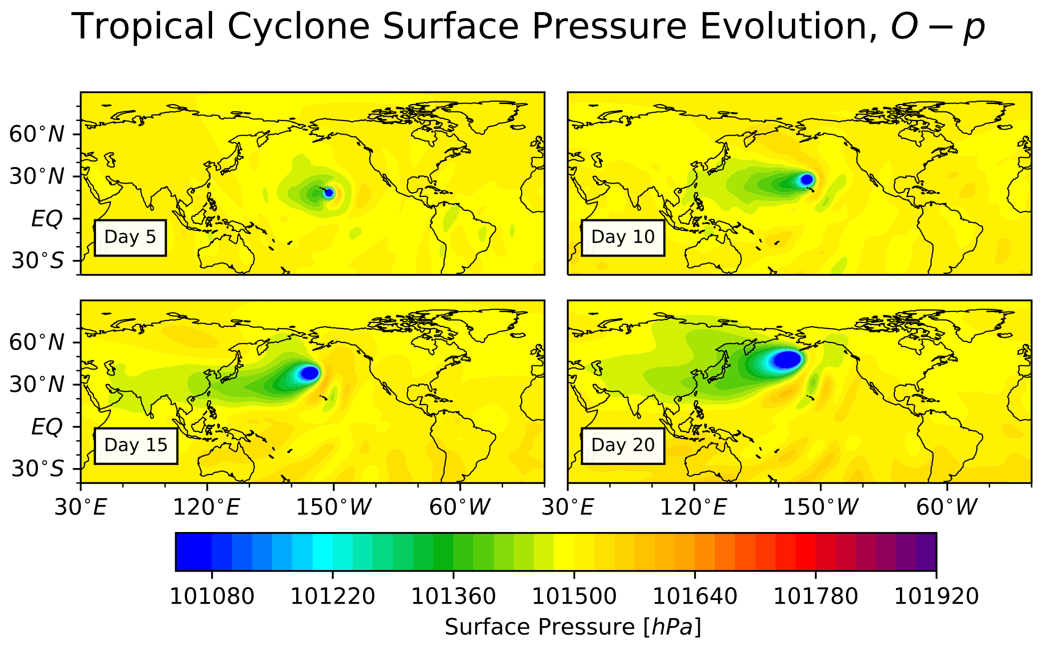


Fig. 4.1. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the original physics of HOMME and a vertical pressure coordinate (combination $O-p$). This is the combination currently used in HOMME. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

High-Order Methods Modelling Environment (HOMME) in a Lagrangian vertical pressure coordinate in Figure 4.1. This is the combination currently used in HOMME. A detailed description of the test case is available in [16]. Further, we utilized thirty vertical levels, a horizontal resolution of approximately $3° \times 3°$, and a maximum time-step size of 200 seconds.

We compare six combinations of physics-vertical coordinate combinations - the original, isobaric, and isochoric physics in both Lagrangian height and pressure vertical coordinates. For shorthand, we denote these combinations in the form $X - x$, where the first letter represents the physics (O for original, P for isobaric, and V for isochoric) and the second represents the vertical coordinate (p for pressure, z for height). For example, $P-z$ represents the combination of isobaric physics and a height vertical coordinate.

As mentioned in Section 3, each physics scheme utilizes the Kessler [5] and/or Reed-Jablonowski [10] cloud microphysics parametrization, which assume that thermodynamic processes occur isobarically [5, 10]. The original physics utilize the values obtained from these parametrizations to update the pressure and use the isobaric temperature update from the isobaric conservation of specific local energy (Eqn. 2.10). The consistency of the original scheme is unclear to us.

On the other hand, the novel isobaric physics does not update the pressure. Hence, the $P - p$ combination is consistent, as each cell in a pressure vertical coordinate has constant pressure. The $V - z$ combination is likewise consistent, as each cell in a height vertical coordinate has constant volume.

Using the $P - z$ combination would result in a temperature change approximately 1.4 times less than that of the $V - z$ combination, resulting in a weaker cyclone. In contrast, using the $V - p$ combination would result in a temperature change approximately 1.4 times

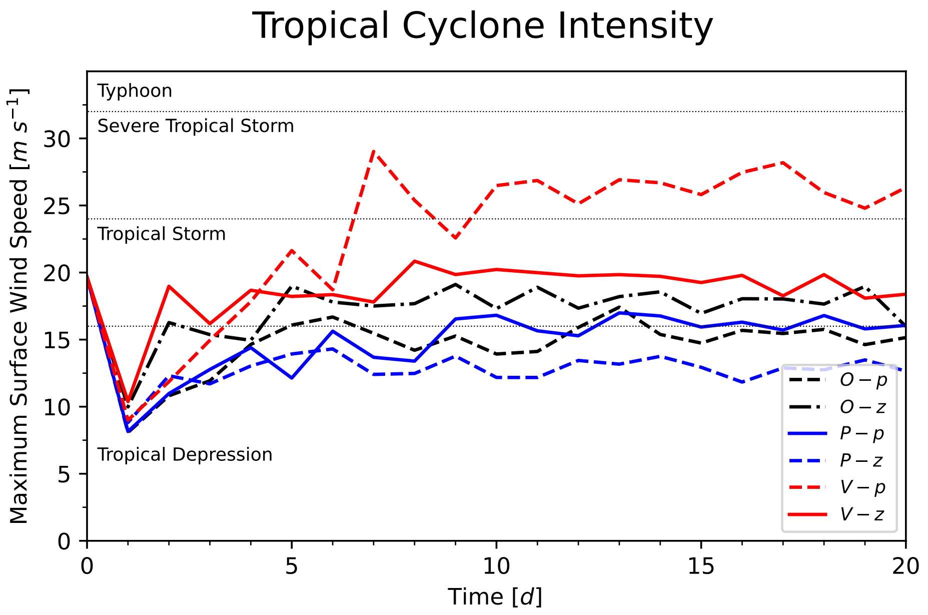


Fig. 4.2. *Evolution of cyclone intensity (maximal surface wind speed) in the DCMIP 2016 tropical cyclone test case for each physics-vertical coordinate combination. The solid lines represent consistent physics-vertical coordinate combinations. The dashed and dot-dashed lines represent physics-vertical coordinate combinations which are not consistent. The thin horizontal dotted lines represent the demarcations of cyclone intensity categories, which are labelled on the left.*

greater than that of $P-p$ combination, resulting in a stronger tropical cyclone.

To compare cyclone intensity, we utilize the classification system of the World Meteorological Organization [18]. The system has five categories, although only four are relevant to our purposes

(ii) **Tropical depression:** The central position of the tropical cyclone can be identified, and the maximum sustained surface wind speed does not exceed 33 knots (16.98 $m\,s^{-1}$).

(iii) **Tropical storm:** The maximum sustained surface wind speed is between 34 knots (17.5 $m\,s^{-1}$) and 47 knots (24.18 $m\,s^{-1}$).

(iv) **Severe tropical storm:** The maximum sustained surface wind speed is between 48 knots (24.69 $m\,s^{-1}$) and 63 knots (32.41 $m\,s^{-1}$).

(v) **Typhoon:** The maximum sustained surface wind speed is at least 64 knots (32.92 $m\,s^{-1}$).

Figure 4.2 shows the evolution of the intensity of the cyclone in each combination over a 20-day period.

As Figure 4.2 illustrates, the choice of physics-vertical coordinate combination can lead to a tropical storm decaying into a tropical depression or intensifying into a severe tropical storm.

In both consistent combinations ($P-p$ and $V-z$), the intensity of the tropical cyclone remains relatively consistent throughout the simulation. Indeed, the intensities of the cyclone in each combination approach each other as time progresses. Figures 4.3 and 4.4 show the evolution of surface pressure for these combinations.

As expected, the cyclone in the $P-z$ combination is consistently less intense, than the $V-z$ combination, quickly decaying from a tropical storm to a tropical depression. The

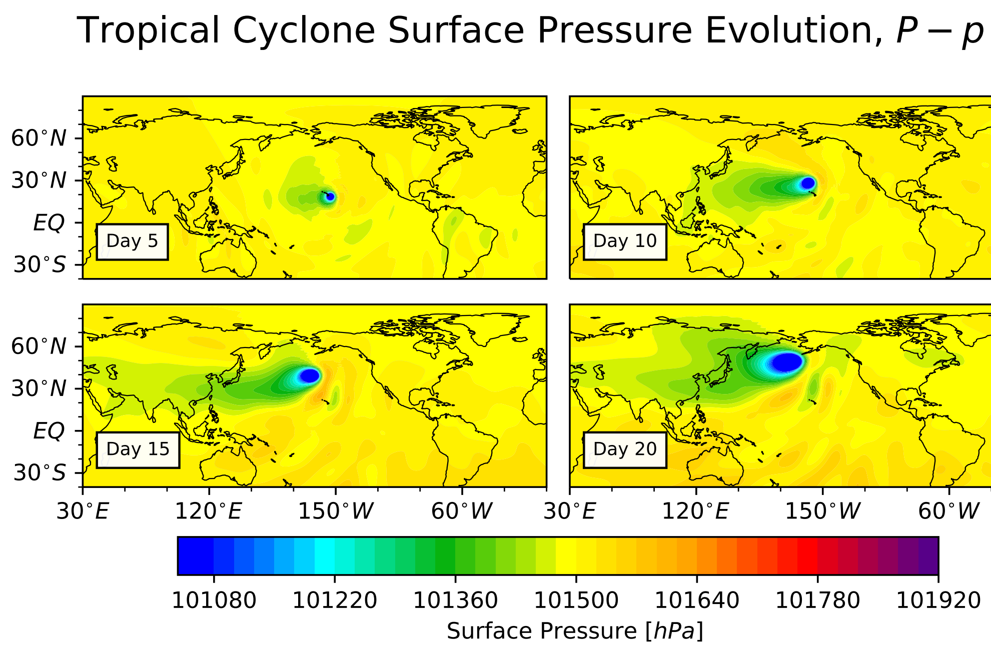


FIG. 4.3. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isobaric physics of HOMME and a vertical pressure coordinate (combination $P-p$). Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

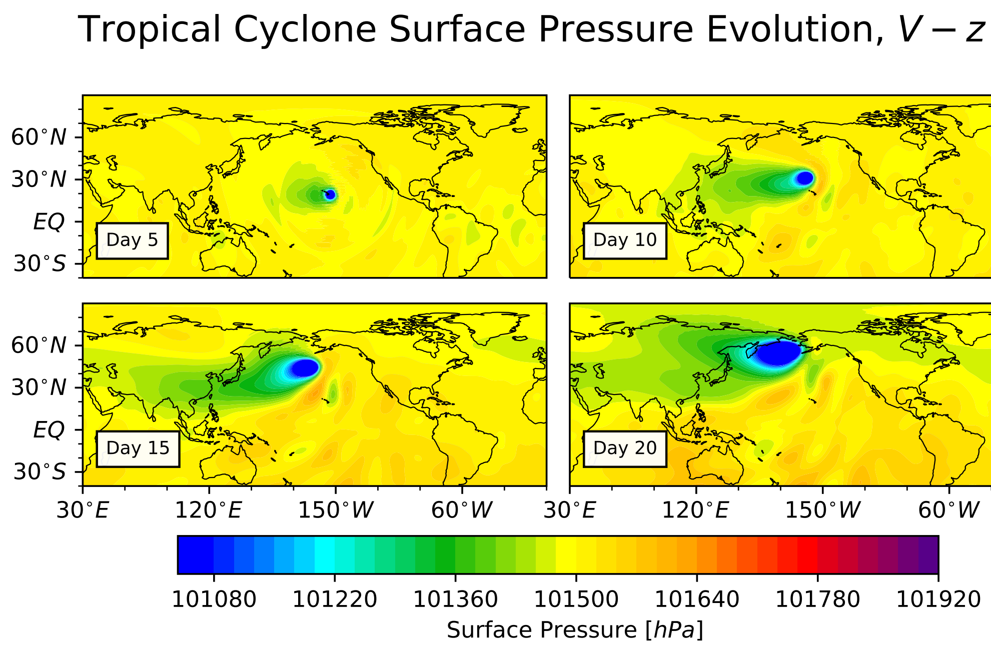


FIG. 4.4. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isochoric physics of HOMME and a vertical height coordinate (combination $O-p$). Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

cyclone in the $V-p$ combination quickly grows into a severe tropical storm due to the greater temperature change from the isochoric physics compared to the isobaric.

The cyclone in the $O-p$ combination and the $P-p$ combination have a similar intensity, as do the cyclone in the $O-z$ and $V-z$ combinations. This suggests that the original physics of HOMME are a sort of hybrid between the isobaric and isochoric physics interfaces.

5. Related Works. Physics-dynamics coupling in weather and climate modelling is a highly active area. For a broad overview on physics parametrizations, from soil-vegetation-atmosphere to cloud microphysics, we recommend [14]. Gross et. al. present a comprehensive review of progress in the field in [3]. Lauritzen and Williamson present a thorough examination of physics-dynamics coupling on energy conservation in the Community Atmosphere Model in [9, 8]. Zhang et. al. build on this work and compare various physics parametrization-vertical coordinate combinations in a multiscale dynamical model in [19].

For more information on the Kessler cloud microphysics parametrization, we direct the reader toward both the original publication [5], as well as the restatement, analysis, and implementation by Klemp et. al. [6, 7]. Although the original parametrization was introduced over 50 years ago, it continues to be generalized and developed. Recently, Tissaoui et. al. generalized the Kessler microphysics parametrization to be non-column-based in [15].

The primary reference on the Reed-Jablonowski physics parametrizations is [10]. Reed and Jablonowski present a mathematical development of the parametrization and analyze its performance in an idealized tropical cyclone test case.

6. Conclusions and Future Work. Our primary contribution was the introduction and comparison of novel isobaric (constant-pressure) and isochoric (constant-volume) versions of the physics interface provided by the 2016 Dynamical Core Model Intercomparison Project (DCMIP 2016) [17]. We compared each of these three physics interfaces in combination with a height and a pressure vertical coordinate in a tropical cyclone test case in the High-Order Method Modelling Environment (HOMME).

We showed that the two 'consistent' combinations (isobaric physics with a pressure vertical coordinate and isochoric physics with a height vertical coordinate) largely agreed on the predicted intensity of the cyclone. The two 'inconsistent' combinations (isobaric physics with a height vertical coordinate and isochoric physics with a pressure vertical coordinate) over- and under-estimated the intensity of the cyclone in a predictable manner.

The original physics interface of HOMME produced a cyclone with an intensity similar to that of the consistent isobaric and isochoric interfaces, regardless of the choice of vertical coordinate. This suggested that the original physics interface was somewhere between the consistent isobaric and isochoric interfaces.

There are two predominant future directions to explore:

1. The primary assumption of our work was that of a hydrostatic environment. How do the physics interfaces affect this quantity? What is the conservation equation for this energy, and is there a simple way to convert between the expected isobaric and isochoric temperature changes?
2. During our investigations, we found that the physics parametrization for the planetary boundary layer had a significant impact on the stability of the simulation. In almost every height-coordinate combination (including one combination involving the original physics), several smaller cyclones were generated along the equator. Eliminating these cyclones required reducing the coefficients for the transfer of water vapor and heat from the sea to the atmosphere by a factor of 10, which is much less than observed values [1].

REFERENCES

[1] P. Black, E. D'Asaro, W. Drennan, J. French, P. Niiler, T. Sanford, E. Walsh, and J. Zhang, *Air-sea exchange in hurricanes: Synthesis of observations from the coupled boundary layer air-sea transfer experiment*, Bulletin of the American Meteorological Society, 88 (2007), pp. 337–374.

[2] C. Eldred, M. Taylor, and O. Guba, *Thermodynamically consistent versions of approximations used in modelling moist air*, Quarterly Journal of the Royal Meteorological Society, (2022), pp. 1–27.

[3] M. Gross, H. Wan, P. Rasch, P. Caldwell, D. Williamson, D. Klocke, C. Jabolonowski, D. Thatcher, N. Wood, M. Cullen, B. Beare, M. Willett, F. Lemairé, E. Blayo, S. Malardel, P. Termonia, A. Gassmann, P. Lauritzen, H. Johansen, C. Zarzycki, K. Skaguchi, and R. Leung, *Physics-dynamics coupling in weather, climate, and earth system models: Challenges and recent progress*, Monthly Weather Review, 146 (2018), pp. 3505–3544.

[4] W. Hannah, C. Jones, B. Hillman, M. Norman, D. Bader, M. Taylor, L. Leung, M. Pritchard, M. Branson, G. Lin, K. Pressel, and J. Lee, *Initial results from the super-parametrized E3SM*, Journal of Advances in Modeling Earth Systems, 12 (2020).

[5] E. Kessler, *On the Distribution and Continuity of Water Substance in Atmospheric Circulations*, American Meteorological Society, Boston, MA, 1969, pp. 1–84.

[6] J. Klemp, W. Skamarock, and S. Park, *A simple model of the hurricane boundary layer revisited*, Journal of Advances in Modeling Earth Systems, 7 (2015), pp. 1155–1177.

[7] J. Klemp and R. Wilhemson, *The simulation of three-dimensional convective storm dynamics*, Journal of the Atmospheric Sciences, 35 (1978), pp. 1070–1096.

[8] P. Lauritzen, N. Kevlahan, T. Toniazzo, C. Eldred, T. Dubos, A. Gassmann, V. Larson, C. Jablonowski, O. Guba, B. Shipway, B. Harrop, F. Lemarié, R. T. an dA. R. Herrington, W. Large, P. Rasch, A. Donahue, H. Wan, A. Conley, and J. Bacmeister, *Reconciling and improving formulations for thermodynamics and conservation principles in Earth System Models (ESMs)*, Journal of Advances in Modeling Earth Systems, 14 (2022).

[9] P. Lauritzen and D. Williamson, *A total energy error analysis of dynamical cores and physics-dynamics coupling in the Community Atmosphere Model (CAM)*, Journal of Advances in Modeling Earth Systems, 11 (2019), pp. 1309–1328.

[10] K. Reed and C. Jablonowski, *Idealized tropical cyclone simulations of intermediate complexity: A test case for AGCMs*, Journal of Advances in Modeling Earth Systems, 4 (2012), pp. 1–25.

[11] R. Rogers and M. Yau, *A Short Course in Cloud Physics*, Pergamon Press, 1989.

[12] W. Skamarock, J. Klemp, M. Duda, L. Fowler, S. Park, and T. Ringler, *A multiscale nonhydrostatic atmosphere model using centroidal Voronoi tesselations and C-grid staggering*, Monthly Weather Review, 140 (2012), pp. 3090–3105.

[13] W. Skamarock, J. Klemp, J. Dudhia, D. Gill, D. Barker, W. Wang, and J. Powers, *A description of the advanced research WRF version 2*. Electronically, Jun. 2005.

[14] D. Stensrud, *Parametrization Schemes: Keys to Understanding Numerical Weather Prediction Models*, Cambridge University Press, Cambridge, UK, 2007.

[15] Y. Tissaoui, S. Marras, A. Quaini, F. de Brangaca Alves, and F. Giraldo, *A non-column based, fully unstructured implementation of Kessler's microphysics with warm rain using continuous and discontinuous spectral elements*. Electronic, arXiv, Jul. 2022.

[16] P. Ullrich, C. Jablonowski, K. Reed, C. Zarzycki, P. Lauritzen, R. Nair, J. Kent, and A. Verlet-Banide, *Dynamical core model intercomparison project (DCMIP2016) test case document*. Electronic, technical report, 2016.

[17] P. Ullrich, A. Verlet-Banide, P. Lauritzen, K. Reed, M. Taylor, and 'poijqwef', *Dynamical core model intercomparison project test case document 2016*. Electronic, public code repository, 2016.

[18] World Meteorological Organization, *Typhoon Committee Operational Manual: Meteorological Component*. Electronic, technical report, 2015.

[19] Y. Zhang, J. Li, R. Yu, Z. Liu, X. Li, and X. Huang, *A multiscale dynamical model in a dry-mass coordinate for weather and climate modeling: Moist dynamics and its coupling to physics*, Monthly Weather Review, 148 (2020), pp. 2671–2699.

Appendix A. Results for All Physics-Vertical Coordinate Combinations.

For completeness we include the evolution of surface pressure for all physics-vertical coordinate combinations, including the use of both the Kessler microphysics parametrization [5] and Reed-Jablonowski microphysics parametrization [10].

We show the evolution of the intensity of the tropical cyclones using the Kessler microphysics parametrizations in Figure A.1. Figures A.2, A.3, A.4, A.5, A.6, and A.7 show the evolution of the surface pressure across a 20-day simulation for all six combinations of these combinations. In each combination, the tropical cyclone drifts north-northwest and grows

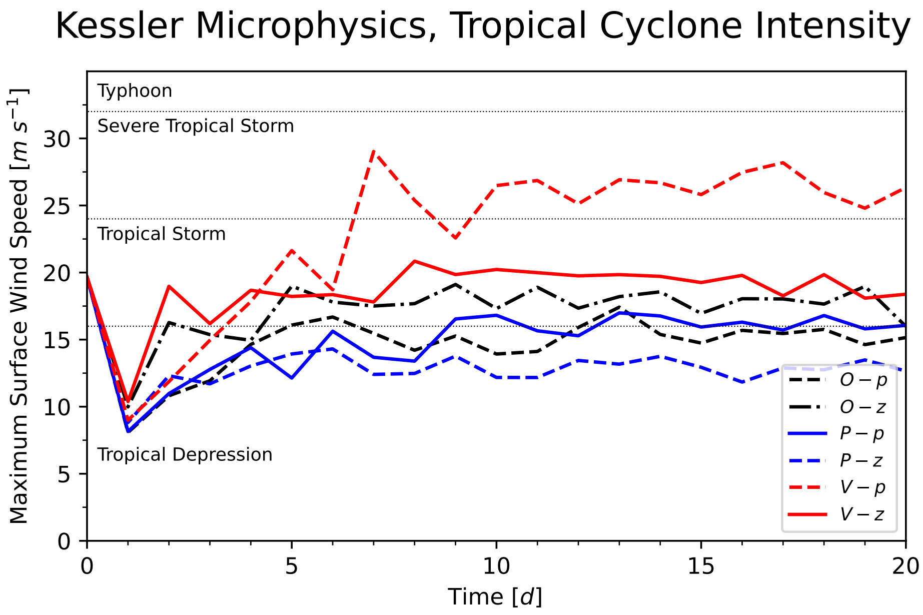


Fig. A.1. *Evolution of cyclone intensity (maximal surface wind speed) in the DCMIP 2016 tropical cyclone test case for each physics-vertical coordinate combination, using the Kessler microphysics parametrization. The solid lines represent consistent physics-vertical coordinate combinations. The dashed and dot-dashed lines represent physics-vertical coordinate combinations which are not consistent. The thin horizontal dotted lines represent the demarcations of cyclone intensity categories, which are labelled on the left.*

in size. The speed at which the cyclone progresses and the amount ii grows is dependent on the physics-vertical coordinate combination, ranging from the small, slow cyclone in the $P - z$ combination to the large, fast cyclone of the $V - p$ combination.

We show the evolution of the intensity of the tropical cyclones using the Reed-Jablonowski microphysics parametrizations in Figure A.8. Figures A.9, A.10, A.11, A.12, A.13, and A.14 show the evolution of the surface pressure across a 20-day simulation for all six combinations of these combinations. Unlike all other tests, the original physics of HOMME using Reed-Jablonowski microphysics required the true value of the bulk transfer coefficient for water vapor C_E and the bulk heat transfer coefficient C_H to sustain the tropical cyclone [1]. In each combination, the tropical cyclone drifts north-northwest and grows in size. The speed at which the cyclone progresses and the amount it grows is dependent on the physics-vertical coordinate combination, ranging from the small, slow cyclone in the $P - z$ combination to the large, fast cyclone of the $V - p$ combination.

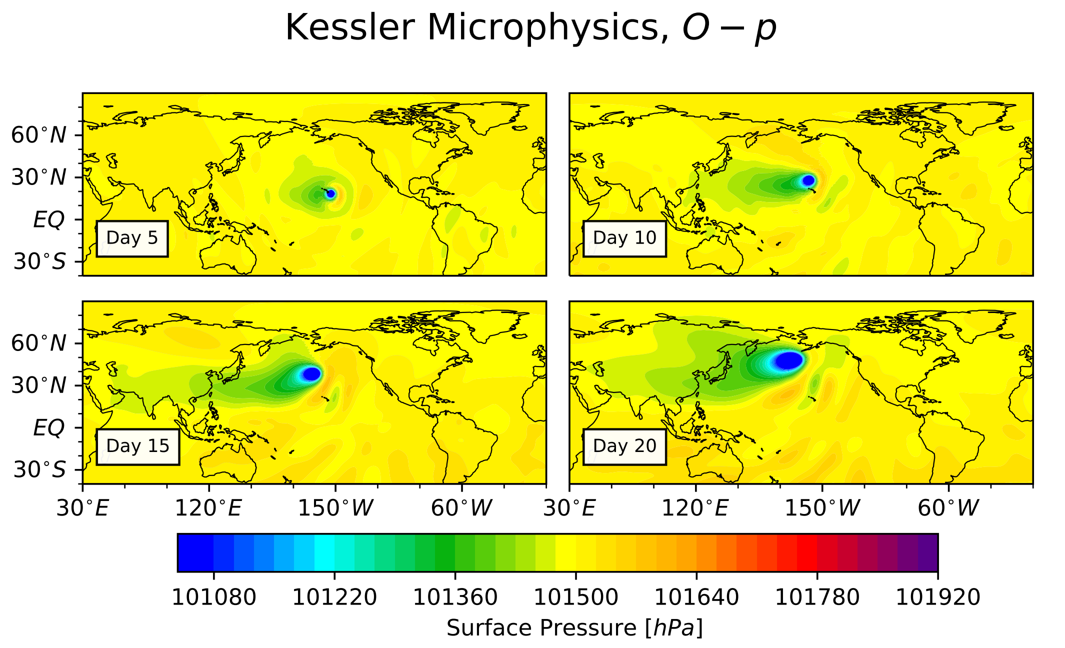


FIG. A.2. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the original physics of HOMME and a vertical pressure coordinate (combination $O-p$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

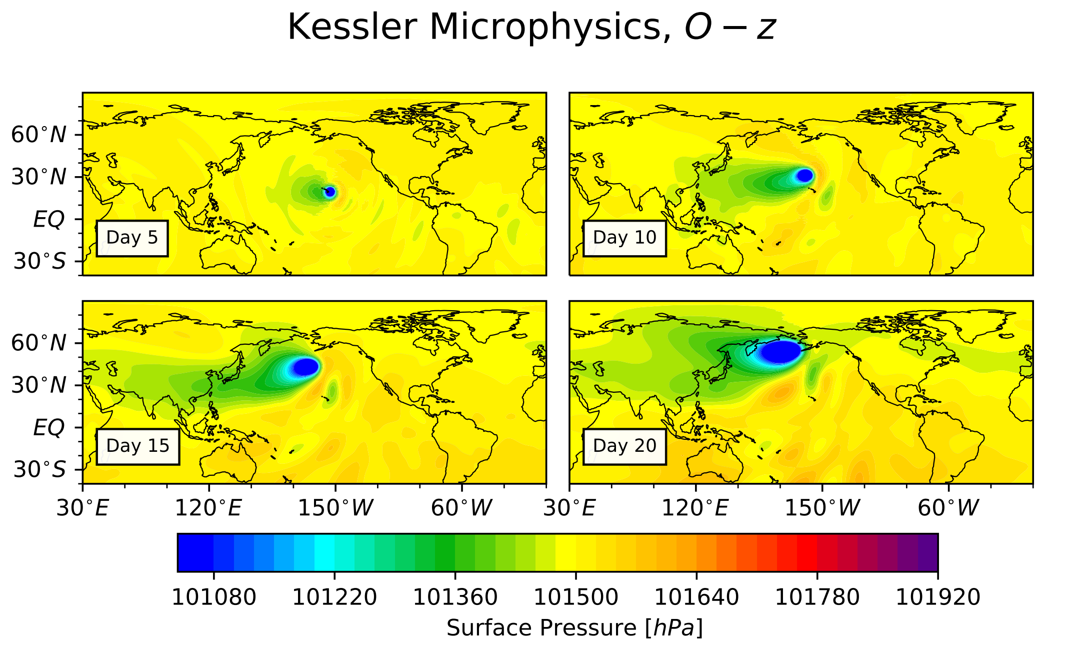


FIG. A.3. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the original physics of HOMME and a vertical height coordinate (combination $O-z$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

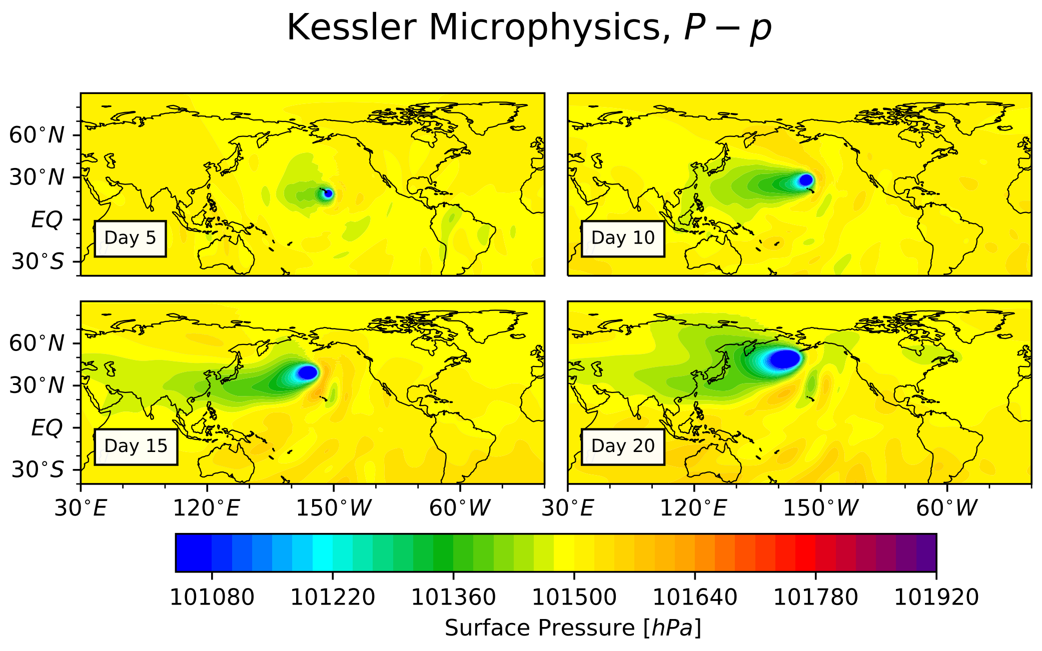


FIG. A.4. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isobaric physics of HOMME and a vertical pressure coordinate (combination $P-p$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

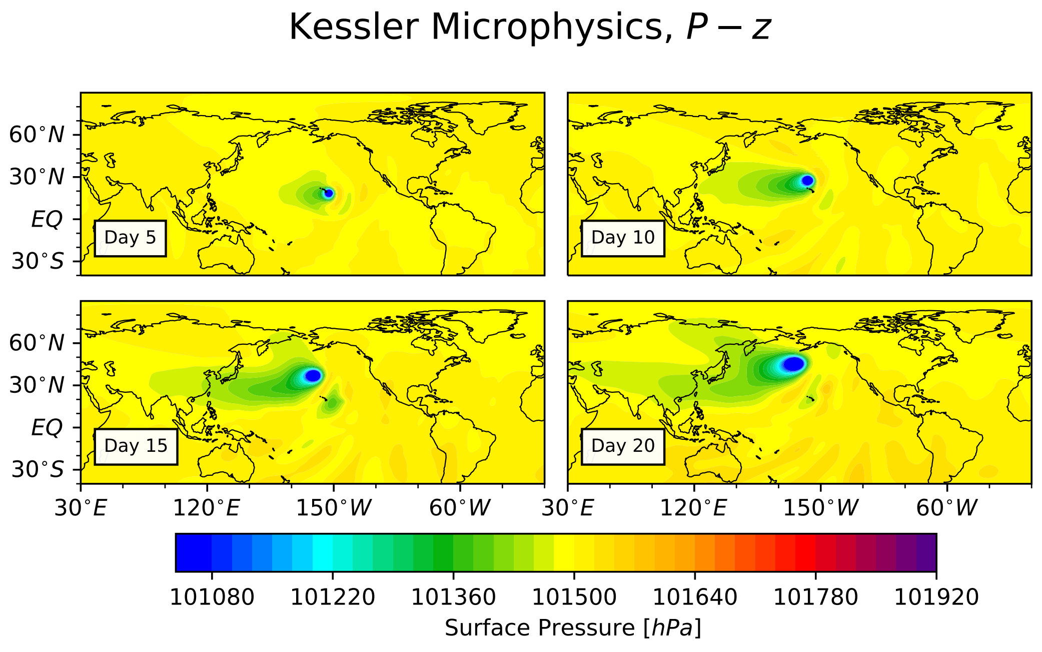


FIG. A.5. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isobaric physics of HOMME and a vertical height coordinate (combination $P-z$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

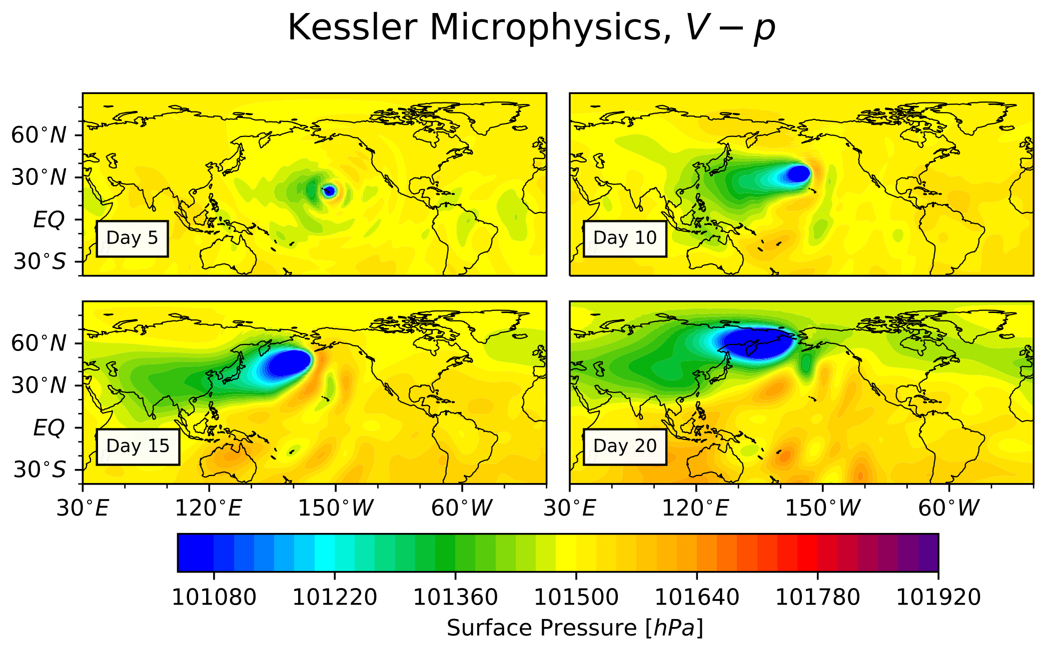


Fig. A.6. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isochoric physics of HOMME and a vertical pressure coordinate (combination $V-p$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

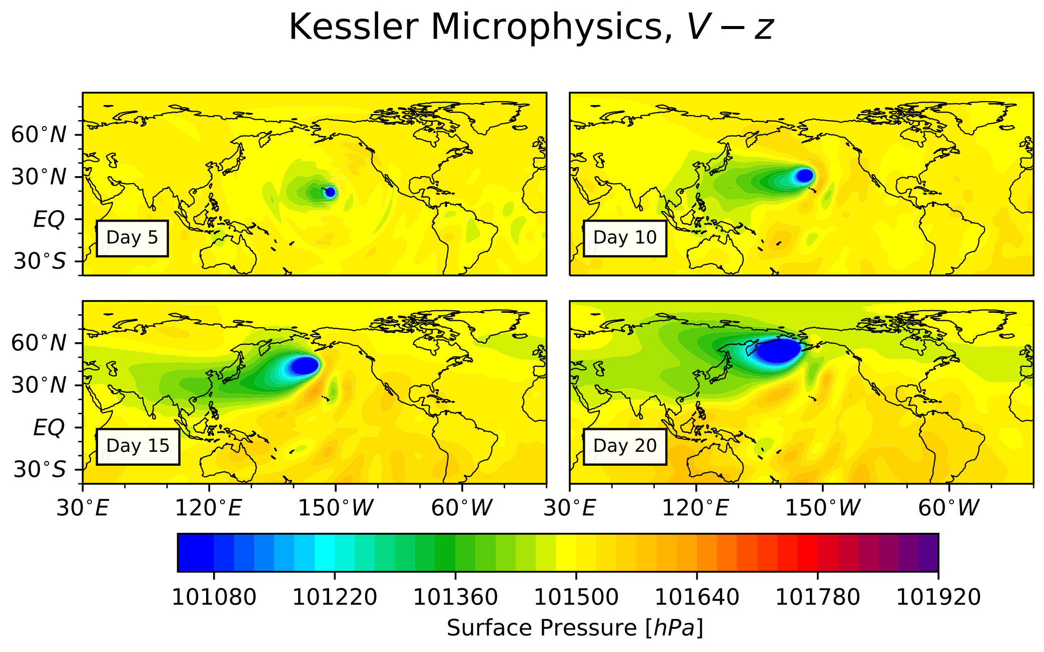


Fig. A.7. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isochoric physics of HOMME and a vertical height coordinate (combination $V-z$), with Kessler microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

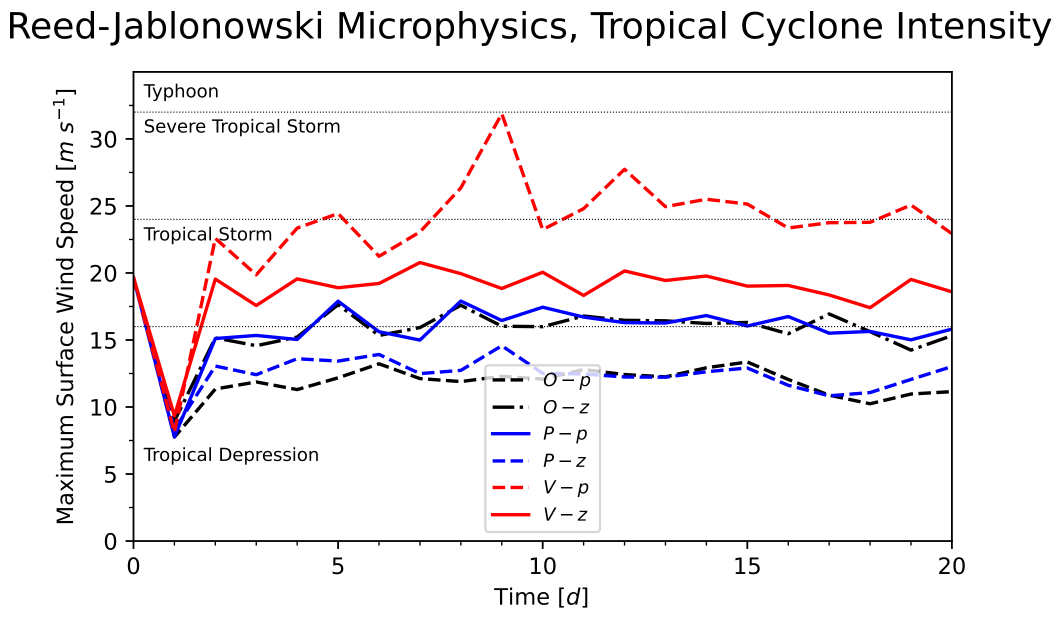


FIG. A.8. *Evolution of cyclone intensity (maximal surface wind speed) in the DCMIP 2016 tropical cyclone test case for each physics-vertical coordinate combination, using the Reed-Jablonowski microphysics parametrization. The solid lines represent consistent physics-vertical coordinate combinations. The dashed and dot-dashed lines represent physics-vertical coordinate combinations which are not consistent. The thin horizontal dotted lines represent the demarcations of cyclone intensity categories, which are labelled on the left.*

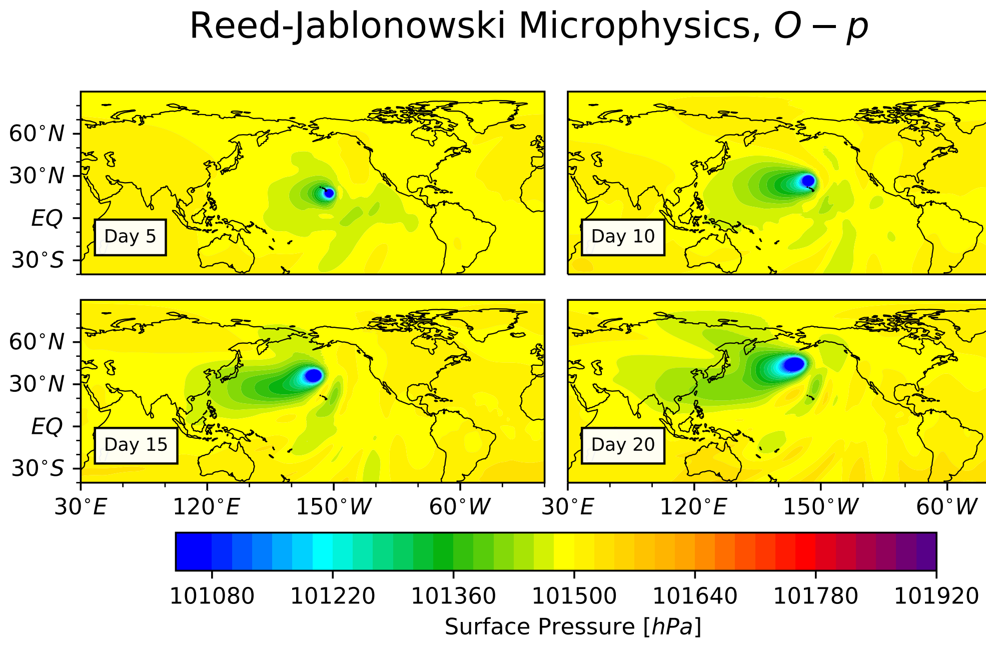


Fig. A.9. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the original physics of HOMME and a vertical pressure coordinate (combination $O-p$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale. Further, unlike all other combinations, each combination using the original physics of HOMME with the Reed-Jablonowski microphysics scheme required the true value of the bulk transfer coefficient for water vapor C_E and the bulk heat transfer coefficient C_H to sustain the tropical cyclone.*

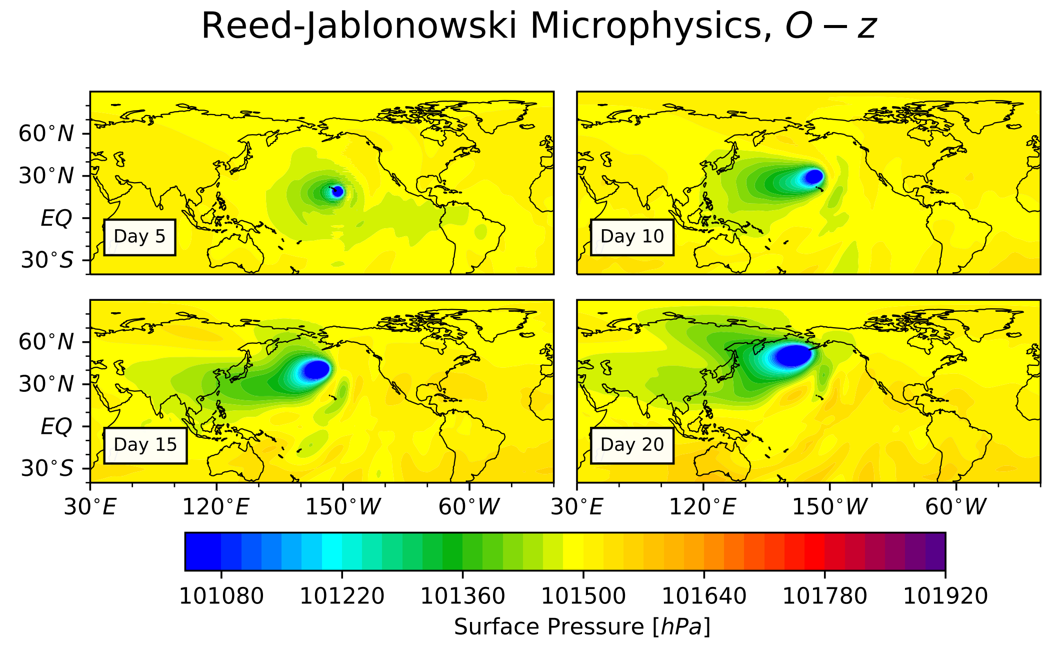


Fig. A.10. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the original physics of HOMME and a vertical height coordinate (combination $O-z$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale. Further, unlike all other combinations, each combination using the original physics of HOMME with the Reed-Jablonowski microphysics scheme required the true value of the bulk transfer coefficient for water vapor C_E and the bulk heat transfer coefficient C_H to sustain the tropical cyclone.*

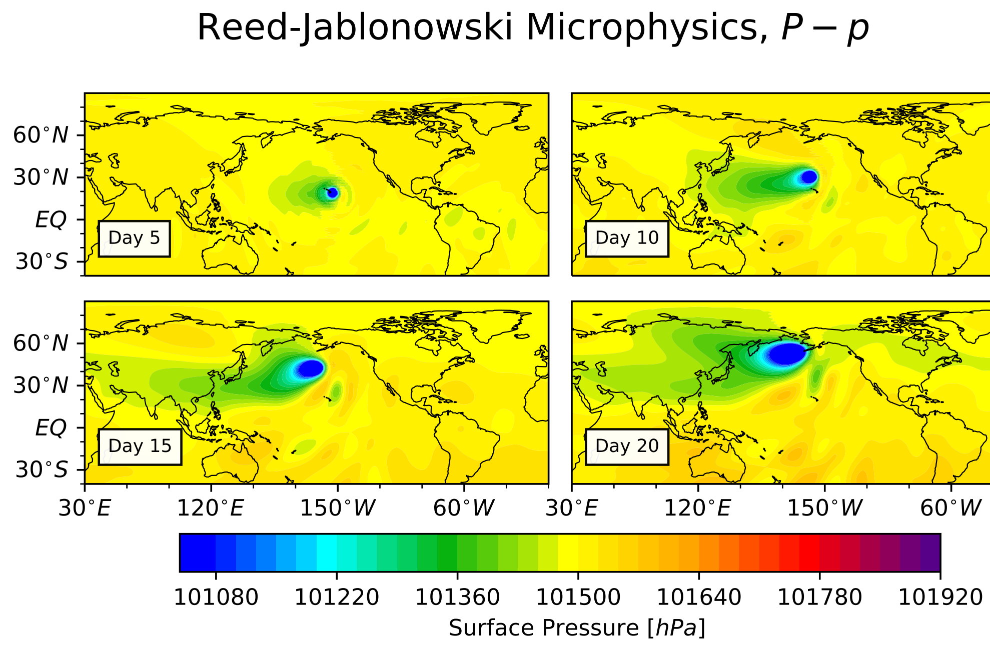


FIG. A.11. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isobaric physics of HOMME and a vertical pressure coordinate (combination $P-p$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

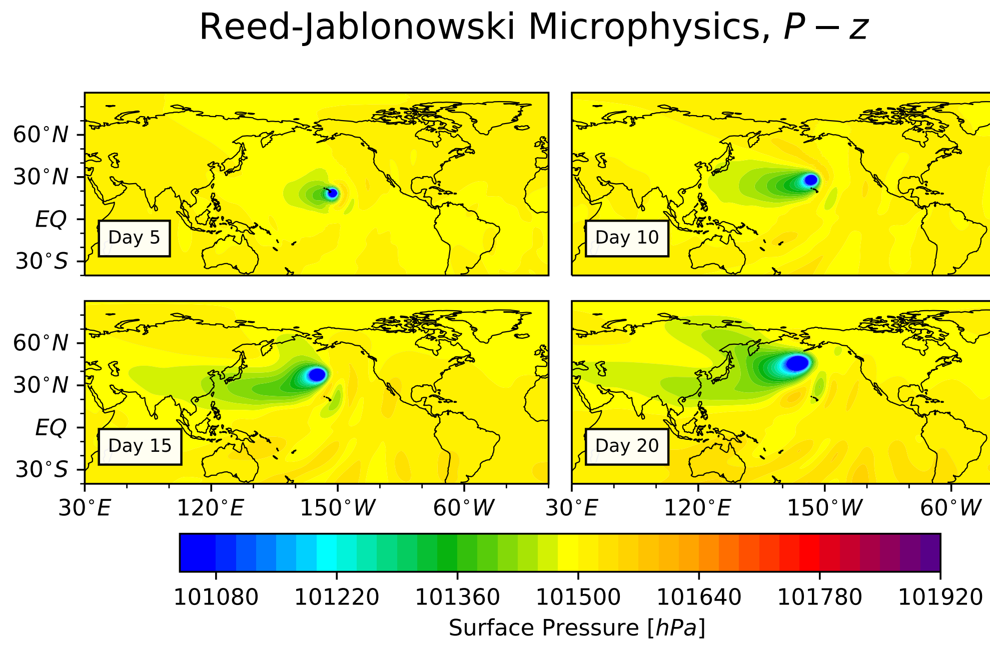


FIG. A.12. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isobaric physics of HOMME and a vertical height coordinate (combination $P-z$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

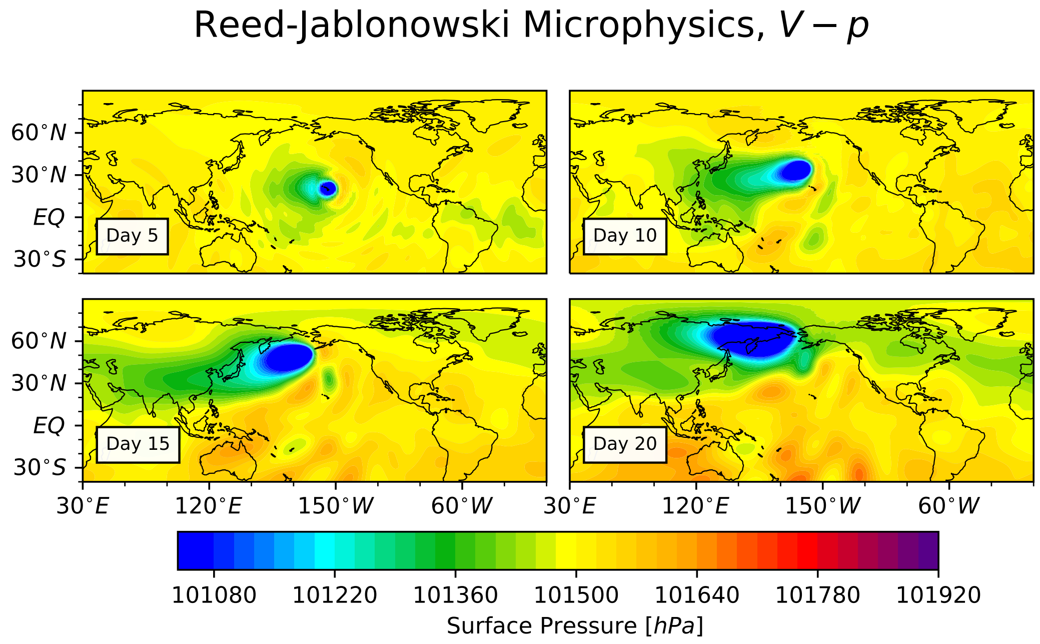


FIG. A.13. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isochoric physics of HOMME and a vertical pressure coordinate (combination $V-p$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

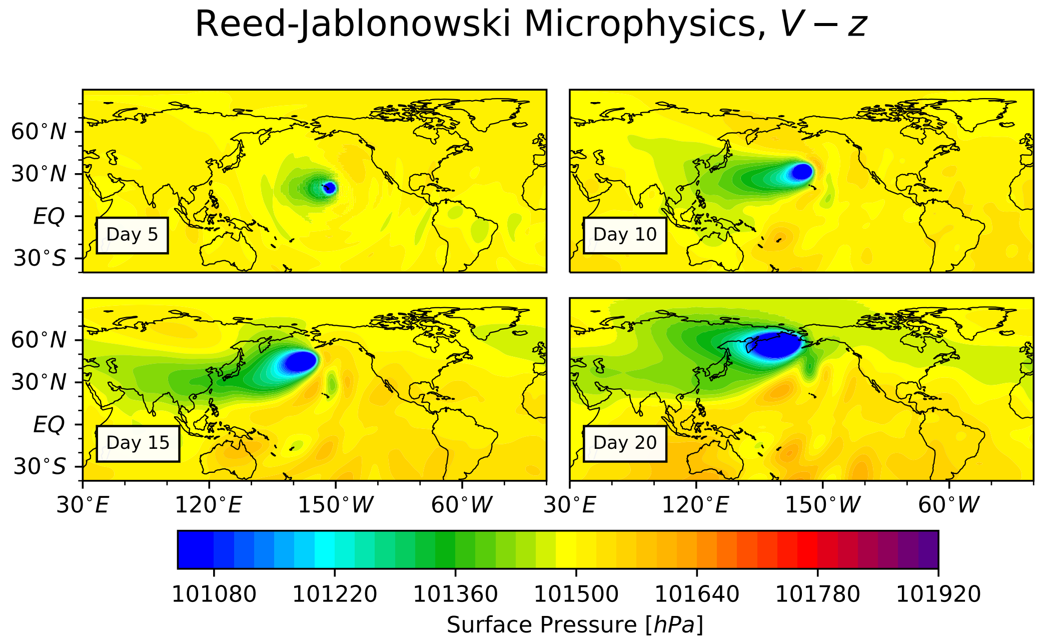


FIG. A.14. *Evolution of the surface pressure in the DCMIP 2016 tropical cyclone test case, utilizing the isochoric physics of HOMME and a vertical height coordinate (combination $V-z$), with Reed-Jablonowski microphysics. Note that the test case uses an Earth-size aqua planet. We include coastlines to provide an intuitive visual length-scale.*

PERIDYNAMIC CONTACT HANDLING

JACOB WILSON* AND STEWART SILLING†

Abstract. In the simulation of colliding objects, the peridynamic numerical method can become unstable due to contact handling failure. Under unfavorable conditions, material volumes of separate bodies pass through each other resulting in an unphysical tangling which we refer to as interpenetration. Existing contact algorithms require tuning parameters that must be carefully tuned to prevent such contact failure. Here, we explore the implementation of a viscous contact force as a robust solution to interpenetration. The proposed viscous contact force is tested under a wide range of impact conditions including with variable impact speed and material properties. The stability and convergence of the viscous contact force is also examined.

1. Introduction. Proper handling of contact forces is critical in the successful modeling of a wide range of phenomena such as particulate erosion, damage due to space debris, and automotive collisions. Contact algorithms used in conventional structural mechanic numerical methods such as the finite element method (FEM) have been heavily studied [1, 2] and shown to be successful in reproducing contact response. Contact treatment in peridynamics, a more recent, non-local continuum mechanics formulation, has received far less attention, a notable exception being [10]. As the contact methods that have been successfully applied to the FEM local method cannot be immediately extended to the non-local peridynamic method, contact handling methods in the context of peridynamic modeling must be more carefully explored.
The "short-range force method" may be regarded as the current standard in peridynamic contact treatment method employed by, for instance, Peridigm, a leading implementation of peridynamics [4, 5]. In this method, the contact is modeled as a series of massless "contact springs" placed between two impacting bodies. With the correct spring constant, the short-range force method can successfully prevent interpenetration, but as this constant is not strictly physical, it is difficult, if possible, to determine a priori. Instead, a guess-and-check procedure must be conducted which can be excessively time consuming. This issue is exacerbated by the fact that new contact spring constants must be identified whenever simulation parameters such as contact speeds and material properties are modified.
In this work, inspired by the strain rate hardening phenomenon, we investigate the use of a linear, viscous contact force as a robust contact model for use with the peridynamic formulation. Implementation of a viscous contact force in peridynamics has been studied previously [6] though not in the context of impact over a wide range of impact speed that we study here. Overcoming the main challenge of the short-range contact force model, we will show that the viscous contact force model successfully prevents the interpenetration phenomenon over a wide range of impact speeds without the need to carefully tune contact parameters. We also demonstrate that, unlike the short-range force model, the parameter required in the viscous contact model can be reasonably predicted a priori which reduces the need for a time-consuming guess-and-check parameter search process. As the viscous contact force has not been the subject of comprehensive study in the peridynamic simulation of colliding bodies, we also work to demonstrate the method's parameter stability and numerical behavior.

2. Peridynamic Model. Peridynamics is a reformulation of continuum mechanics developed by Silling in the early 2000s [7, 8] that relies on integro-differential equations rather than the fully differential governing equations of traditional continuum mechanics.

*Virginia Tech, jacobow@vt.edu
†Sandia National Laboratories, sasilli@sandia.gov

With this feature, peridynamics is naturally able to model spatial discontinuities that require special treatment in the traditional formulation. As opposed to the traditional continuum mechanics formulation, peridynamics is non-local in nature such that material points interact with other points in a finite rather than the traditionally infinitesimal neighborhood. Letting $\mathcal{H}_x$ denote the neighborhood of points within a finite distance δ of material point x in body's reference configuration, the peridynamic equation of motion of point x at time t is

$$\rho_x \frac{\partial^2 x}{\partial t^2} = \int\limits_{\mathcal{H}_x} \hat{f}_{x'x}\, dV_{x'} + \mathrm{b}_x \tag{2.1}$$

where ρ_x and b_x are the mass density and net external body force at material point x at time t. $\hat{f}_{x'x}$ is the peridynamic pairwise force density due to material point x' acting on x where x' is a material point in the neighborhood $\mathcal{H}_x$. Note that by this definition, $\hat{f}_{x'x}$ has units of force per squared volume and that the motion of point x automatically satisfies linear momentum conservation provided that $\hat{f}_{x'x}$ is antisymmetric.

Let $x(t)$ be the position of the material point x at time t. For notational convenience, let $\xi_{xx'}(t)$ be the relative position vector from material point x to x' at time t, $\xi_{xx'}(t) = x'(t) - x(t)$, and define a peridynamic strain, $\hat{\epsilon}_{xx'}$, characterizing the deformation between points x and x' as

$$\hat{\epsilon}_{xx'} = \frac{|\xi_{xx'}(t)| - |\xi_{xx'}(0)|}{|\xi_{xx'}(0)|} \tag{2.2}$$

It has been demonstrated that the constitutive model used in mapping deformation to $\hat{f}_{x'x}$ can be designed to reproduce any material model from the traditional continuum mechanics formulation [9]. However, in this work, we focus only on the simplest model, the prototype microelastic brittle (PMB) material, in which the force density of point x' in $\mathcal{H}_x$ on x at time t is given by

$$\hat{f}_{x'x} = \frac{\xi_{xx'}}{|\xi_{xx'}|}\, \hat{k}\hat{\epsilon}_{xx'}\, dV_{x'} \tag{2.3}$$

where $\hat{k}$ is the peridynamic spring constant, the force per squared unit volume per unit peridynamic strain.

As the pairwise force function $\hat{f}_{x'x}$ is only nonzero for material points x' in the neighborhood $\mathcal{H}_x$, a separate force term must be added to the peridynamic equation of motion to model contact forces between points that would not otherwise interact. This additional force term, call it $\tilde{f}_{x'x}$, is needed to model interactions forces between material points that are separated in the reference configuration but come within close proximity as the calculation progresses. The same term also applies to self-contact. With this additional term, the continuum peridynamic equation of motion becomes

$$\rho_x \frac{\partial^2 x}{\partial t^2} = \int\limits_{\mathcal{H}_x} \left[\hat{f}_{x'x} + \tilde{f}_{x'x}\right] dV_{x'} + \mathrm{b}_x \tag{2.4}$$

Though we shall see it results in practical difficulties, it is common to model the contact force by a method known as the short-range force method. This method models contact force as a compression-only spring that is activated when the distance between two material points satisfies a proximity criterion which we represent using a contact activation parameter λ_1 as follows

$$\lambda_1 = \begin{cases} 1 & \text{if } |\xi_{xx'}| \leq \tilde{\delta} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where $\tilde{\delta}$ is the separation distance at which contact between two material points is first activated. It is determined by

$$\tilde{\delta} = \min\left(\alpha|\xi_{xx'}(0)|, \Delta x_{\text{contact}}\right) \tag{2.6}$$

where $\Delta x_{\text{contact}}$ is a constant representing the nominal contact distance, which is typically greater than the initial grid spacing. The term involving α avoids contact force application to material points that are initially close to each other. Defining the peridynamic contact strain between points x and x' as

$$\tilde{\epsilon}_{xx'} = \frac{|\xi_{xx'}| - \tilde{\delta}}{\tilde{\delta}} \tag{2.7}$$

the short-range contact force density of point x' on x at time t is given by

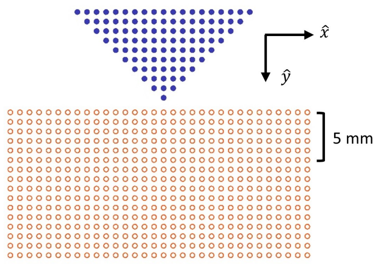
$$\tilde{f}_{x'x} = \lambda_1 \frac{\xi_{xx'}}{|\xi_{xx'}|} \tilde{k} \tilde{\epsilon}_{xx'} \, dV_{x'} \tag{2.8}$$

where $\tilde{k}$ is the contact spring constant with dimensions of force per squared unit volume per unit contact strain. In practice, $\tilde{k}$ is often chosen as the average of the peridynamic spring constants $\hat{k}$ of the materials in contact.

3. Numerical Implementation of Peridynamic Model. In the numerical implementation of peridynamic theory, space is discretized without geometric connectivity, and each node is assigned a fixed volume. At the n^{th} timestep, each individual node i is updated using the discretized peridynamic equation of motion

$$\rho_i \frac{\partial^2 x_i}{\partial t^2} = \sum_j \left[\hat{f}_{x_j x_i} + \tilde{f}_{x_j x_i}\right] V_j + \mathrm{b}_i \tag{3.1}$$

whereρ_i and b_i are the density and body force acting on node i, $\hat{f}_{x_j x_i}$ and $\tilde{f}_{x_j x_i}$ are the peridynamic and contact forces from node j acting on node i, and V_j is the volume of node j. To examine the success of contact force formulations, we chose to model a ballistic impact test involving an impactor in the shape of a triangular prism striking a planar target along one of its edges. Due to the relatively small contact area of this impact geometry and resulting large impact stresses, the test serves as a good indication of contact force robustness. Figure 3.1(a) shows the discretized geometry. Note that we simulate the impact in two-dimensions. The triangular prism has a 9 mm height and 18 mm base; the target has a height of 16 mm and an impact face 32 mm wide. Both geometries have been discretized using a node spacing of 1 mm. Each body responds via the PMB material model with properties summarized in Table 3.1(b), and $\tilde{k}$ was taken to be the average peridynamic spring constant of the two bodies.



(a)

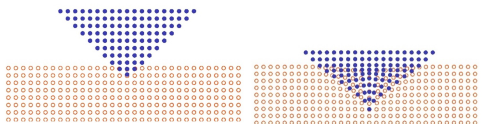
	Target	Impactor
ρ (kg/m^3)	8000	4000
K (GPa)	100	50
ν	1/3	1/3
$\hat{k}$(N/m^6)	1.36×10^{19}	6.8×10^{18}
$\tilde{k}$(N/m^6)	1.02×10^{19}	

(b)

FIG. 3.1. *(a) ballistic impact geometry used in evaluation of contact algorithms; (b) Summary of ballistic impact simulation material properties where K and ν are the bulk modulus and Poisson ratio, respectively*

4. Short Range Contact Force and Interpenetration. In the case of high-speed impact, the short-range contact force was seen to fail in that it allowed material nodes of the impactor and target to intersect in a phenomenon we refer to as interpenetration. The interpenetration numerical phenomenon is clearly not physical as it indicates that material volumes overlap. To demonstrate interpenetration, the case of the impactor striking the target at a speed of 5 km/s in the $\hat{y}$ direction is shown in Figure 4.1.

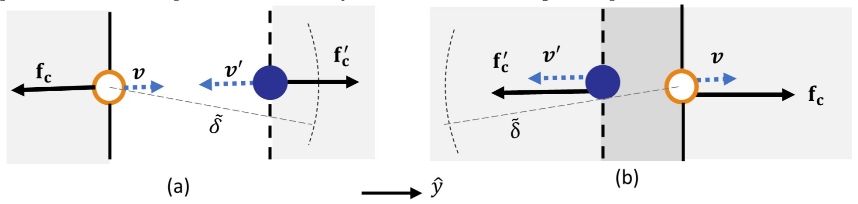
FIG. 4.1. *Interpenetration in the case of an impactor striking the stationary target at 5 km/s.*



Aside from the non-physicality of overlapping volumes, interpenetration is also harmful to the impact simulation fidelity as it leads to the development of unphysical contact forces when the short-range contact force model is used. Referring to figure 3.1(a), as we neglect adhesion, the contact force on the impactor nodes due to interaction with the target should always have a component in the $-\hat{y}$ direction, resisting target penetration. However, referring to the short-range contact force expression, with the onset of interpenetration, the contact force from the target interacting with nodes of the impactor may instead have a component directed in the $+\hat{y}$ direction which unphysically accelerates penetration into the target. The source of this issue is illustrated at the node level in figure 4.2. Note that the displayed contact force pair is only that associated with interaction between the two nodes shown.

To further examine the nature of interpenetration, we developed a simple algorithm for the identification and calculation of the extent of node separation, call it δ_{s}, in the case of ballistic impact involving two bodies. Since the exact direction of possible node interpenetration is not of interest for present purposes, we treated δ_{s} as a signed scalar rather than a vector with a negative sign indicating that interpenetration has occurred. A brief description of the δ_{s} calculation algorithm follows.First, each node i on the impactor boundary is paired with the node i' in the target predicted to be closest to the point at which node i would intersect the target surface in the absence of contact forces. Let $\vec{r}_{i'i}$ be the relative position vector from node i' on the target to i on the impactor at time t. The node separation magnitude, $|\delta_{\mathrm{s}}|$, is then defined as the magnitude of the projection of

Fig. 4.2. *Source of unphysical contact forces that can accelerate interpenetration illustration. (a) nodes of separate bodies approach with a repulsive force that would act against penetration and (b) nodes have interpenetrated and experience contact forces that would augment penetration.*



$\vec{r}_{i'i}$ onto the unit vector in the direction of the incident velocity of the impactor relative to the target, $\hat{v}_{0,\mathrm{rel}}$. Establishing the sign of this node separation requires the specification of a direction that is "away" from the target. This is accomplished by the vectors $\hat{n}_1$ and $\hat{n}_2$ defined in figure 4.3(a). The sign of the separation, $\operatorname{sgn}(\delta_\mathrm{s})$, between the impactor node i and target node i' is then given by

$$\operatorname{sgn}(\delta_\mathrm{s}) = \begin{cases} +1 & \text{if } [(\hat{n}_1 \cdot \vec{r}_{i'i} > 0) \text{ and } (\hat{n}_2 \cdot \vec{r}_{i'i} > 0)] \\ -1 & \text{otherwise} \end{cases} \tag{4.1}$$

Altogether, the extent of interpenetration is defined as

$$\delta_\mathrm{s} = \min\left[\operatorname{sgn}(\delta_\mathrm{s})\,|\vec{r}_{i'i} \cdot \hat{v}_{0,\mathrm{rel}}|\,\right] \tag{4.2}$$

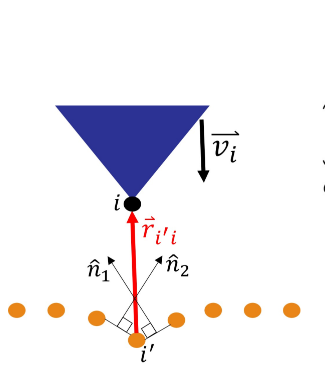
where the minimum is taken over all pairs of nodes, one in each material. Note that δ_s changes in time as the impactor approaches, strikes, and rebounds from the target. Figure 4.3(b) shows the node separation δ_s obtained for a sample of impact speeds in the ballistic impact simulation using the short-range contact force for the ballistic impact simulation described in section 3. We can see that interpenetration first occurs at around 2.5 km/s and increased in severity with increasing incident speed. The worsening of interpenetration as a function of impact speed is not a time step issue but rather an intrinsic issue of the contact force algorithm.

4.1. Viscous Contact Force Formulation. Demonstrated in the preceding example, the short-range contact force method results in excessive interpenetration if its associated contact stiffness parameter is not appropriately tuned. Attempting to solve this issue, we considered augmenting the contact force by a velocity-dependence term such that the contact force becomes

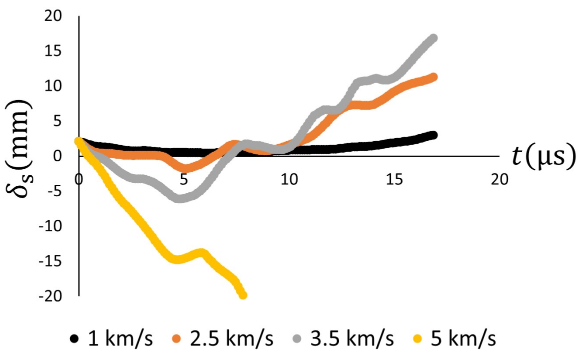
$$\tilde{f}_{x'x} = \left(\lambda_1 \frac{\xi_{xx'}}{|\xi_{xx'}|}\tilde{k}\tilde{\epsilon}_{xx'} - \lambda_2 \tilde{c} v_{xx'}\right) dV_{x'} \tag{4.3}$$

We refer to this two-term expression as the viscous contact force. $v_{xx'}$ is the velocity of material point x relative to point x', and λ_2 is the contact activation parameter associate with the velocity dependent term which restricts the term to act only during the compressive phase of impact as shown in the following definition

$$\lambda_2 = \begin{cases} 1 & \text{if } \left(|\xi_{xx'}| \le \tilde{\delta} \text{ and } \frac{d}{dt}|\xi_{xx'}| < 0\right) \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$



(a)



(b)

FIG. 4.3. *(a) Definition of the directions $\hat{n}_1$ and $\hat{n}_2$ used in determining the sign of δ_s for two arbitrary nodes i and i'; (b) variation of δ_s evolution with impact speed. Negative δ_s indicates interpenetration*

Using the viscous contact force, the discretized equation of motion of node i interacting with node j becomes

$$\rho_i \frac{d^2 x_i}{dt^2} = \sum_j \left[\frac{\xi_{ij}}{|\xi_{ij}|} \hat{k}\hat{\epsilon}_{ij} + \left(\lambda_1 \frac{\xi_{ij}}{|\xi_{ij}|} \tilde{k}\tilde{\epsilon}_{ij} - \lambda_2 \tilde{c} v_{ij} \right) \right] V_j + \mathrm{b}_i \tag{4.5}$$

We next establish an equation for a peridynamic-equivalent critical damping factor, $\tilde{c}_{\text{crit}}$, to serve as a basis for estimating the parameter $\tilde{c}$ in the viscous contact force. Consider a simple discretized peridynamic system consisting of just two nodes along an axis, call it the y-axis, like that illustrated in figure 4.4. Take these nodes to be sufficiently far apart in the reference configuration such that they interact only via contact forces and assume there are no external forces. Fix one of the nodes, call it node A, at the position $y = 0$, and assume that at time τ the other node, node B, is moving radially toward node A such that $\lambda_2 \neq 0$ (i.e., the full viscous contact force is activated). In this case, the equation of motion of node B simplifies to

$$\rho_{\text{B}} \frac{d^2 y}{dt^2} = \left[\left(\tilde{k}\tilde{\epsilon}_{\text{AB}} - \tilde{c} \frac{dy}{dt} \right) \right] V_{\text{A}} \tag{4.6}$$

If interpenetration does not occur, that is, the position of node A always satisfies $y > 0$, we can see that $|\xi_{\text{AB}}| = y$. It's then straightforward to show that

$$\frac{dy}{dt} = \tilde{\delta}\, \frac{d\tilde{\epsilon}_{\text{AB}}}{dt} \quad \text{and} \quad \frac{d^2 y}{dt^2} = \tilde{\delta}\, \frac{d^2 \tilde{\epsilon}_{\text{AB}}}{dt^2} \tag{4.7}$$

so that

$$\rho_{\text{B}} \tilde{\delta}\, \frac{d^2 \tilde{\epsilon}_{\text{AB}}}{dt^2} = \left[\left(\tilde{k}\tilde{\epsilon}_{\text{AB}} - \tilde{c}\tilde{\delta}\, \frac{d\tilde{\epsilon}_{\text{AB}}}{dt} \right) \right] V_{\text{A}} \tag{4.8}$$

or rearranging things slightly

$$\left(\rho_{\mathrm{B}}\tilde{\delta}\right)\frac{d^2\tilde{\epsilon}_{\mathrm{AB}}}{dt^2}+\left(V_{\mathrm{A}}\tilde{c}\tilde{\delta}\right)\frac{d\tilde{\epsilon}_{\mathrm{AB}}}{dt}-\left(V_{\mathrm{A}}\tilde{k}\right)\tilde{\epsilon}_{\mathrm{AB}}=0 \tag{4.9}$$

Comparing this with the equation of motion for a simple damped oscillator constrained to the $y-$axis

$$m\frac{d^2y}{dt^2}+c\frac{dy}{dt}+ky=0 \tag{4.10}$$

and recalling that for this classic motion, the critical damping factor of c is given by $c_{\mathrm{crit}}=\sqrt{km}$, equating coefficients indicates that an expression for a peridynamic-equivalent critical damping factor is

$$\tilde{c}_{\mathrm{crit}}=2\sqrt{\tilde{k}\rho_{\mathrm{B}}/\tilde{\delta}\,V_{\mathrm{A}}} \tag{4.11}$$

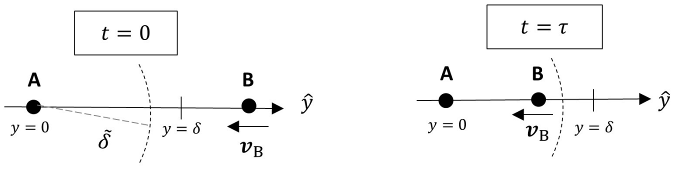


FIG. 4.4. *Simplified two-node impact used in the derivation of the peridynamic critical damping constant*

4.2. Viscous Contact Force Evaluation. Since in this study, we were only interested in the success of contact forces in preventing interpenetration between separate bodies, the viscous contact force was taken to be active only between nodes of different bodies. That is, if $\mathcal{B}_i(\mathcal{B}_j)$ is the body that node $i(j)$ belongs to, the contact force was taken to depend on the nature of the interacting node pair as follows:

$$\tilde{f}_{ij}=\begin{cases}V_j\left(\lambda_1\frac{\xi_{ij}}{|\xi_{ij}|}\tilde{k}\tilde{\epsilon}_{ij}-\lambda_2\tilde{c}v_{ij}\right) & \text{if } \mathcal{B}_i\neq\mathcal{B}_j\\ V_j\left(\lambda_1\frac{\xi_{ij}}{|\xi_{ij}|}\tilde{k}\tilde{\epsilon}_{ij}\right) & \text{otherwise}\end{cases} \tag{4.12}$$

Using the damping constant $\tilde{c}=\tilde{c}_{\mathrm{crit}}$, the interpenetration time history curves of figure 4.5 demonstrate the ability of the viscous contact force to prevent interpenetration over a wide range of incident speeds. The minimum target-impactor separation remains positive in all cases indicating that interpenetration does not occur, and the curves corresponding to lower impact speeds behave as expected with δ_{s} reaching a minimum value before rebounding. The evolution of δ_{s} in the case of the 10 km/s impact speed is a bit unexpected in that it exhibits an oscillation. However, this may simply be attributed to the oscillatory displacement field of a reflected elastic wave interacting with the impactor tip.

To facilitate comparison, the simulation parameters used in obtaining the curves were taken to be those defined in section 3, and as before, $\tilde{k}$ was taken to be the average peridynamic spring constant of the two bodies. For all impact speeds considered, the impactor started the same distance from the target surface. However, since the contact was not detected and the contact force was not engaged until after the first time step, the impactor moved freely from the first to the second data point such that δ_s varies with impact speed in the second reported data point. We note that at the 10 km/s impact speed, two data points at roughly at 1.5 and 5.8 μs with negative δ_s were removed as they are assumed to be a failure of the interpenetration detection algorithm rather than actual interpenetration. Further, due to limitations of the simple penetration distance algorithm, the penetration distance history curves are not intelligible at impact speeds much higher than 10 km/s. However, we can qualitatively evaluate the ability of the viscous contact force to prevent interpenetration at higher impact speeds by visual examination of the impacts. As shown in figure 4.6, the viscous force is qualitatively successful in preventing interpenetration at speeds as high as 50 km/s.

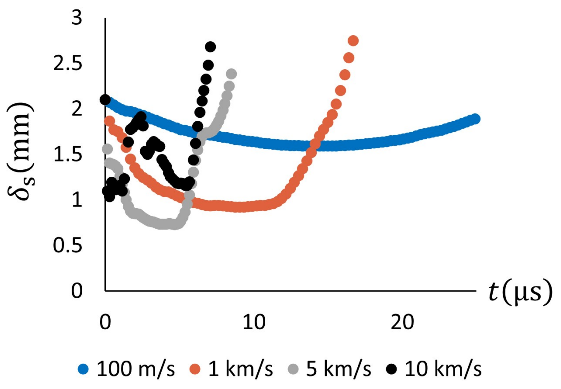


FIG. 4.5. *Variation of δ_s evolution with impact speed with use of the viscous contact force.*

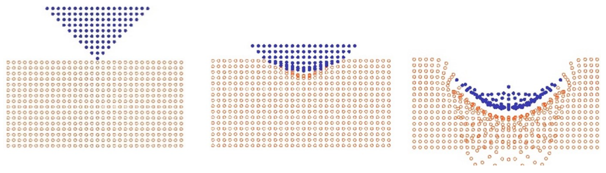


FIG. 4.6. *Interpenetration prevention in the case of 50 km/s impact speed*

4.3. Viscous Contact Force Numerical Behavior and Stability. As the velocity-dependent contact force term may contribute considerably to total contact stiffness, we evaluate the extent to which the term influences the numerical stability of the impact simulation. To accomplish this, we analyzed the total system energy variation with timestep when the full viscous contact force was activated ($\lambda_1 = \lambda_2 = 1$) and compare this to the same simulation but with the velocity-dependent term deactivated ($\lambda_1 =; \lambda_2 = 0$). Figure 4.7 shows the result of such an analysis conducted using the simulation defined in section 3 with the impactor striking the target at an incident speed of 1 km/s and the damping constant set to the critical value $\tilde{c} = \tilde{c}_{\text{ref}}$. The reported timesteps in figure 4.7 are normalized by an estimate of the time it takes information to travel between nodes in the reference configuration given by $\text{dt} = dx/\sqrt{K/\rho}$ where dx is the reference node separation, K is the bulk modulus, and ρ is the mass density. In the current simulation, this gives $\text{dt} \approx 0.28\mu\text{s}$.

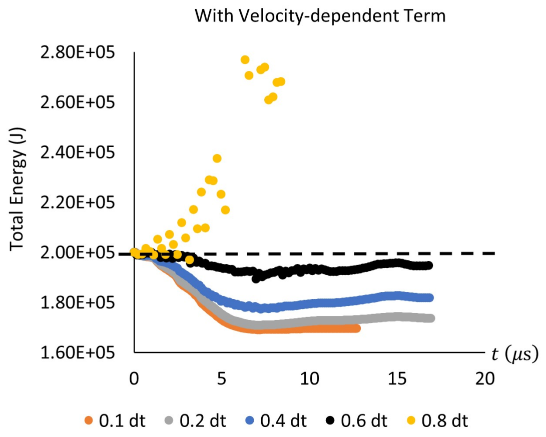
We take the onset of instability to correspond to the timestep at which the total system energy history grows significantly compared to its evolution using a converged, stable timestep. By this criterion, the velocity-dependent term seems to reduce the maximum stable timestep from around 1.4-1.6dt to around 0.2-0.4 dt. Note that with the velocity-dependent term necessarily leads to energy dissipation such that, as illustrated in figure 4.7, a stable simulation is expected to experience some level of total energy loss.

In addition to the effect of contact force stiffening, the viscous contact force has a discontinuity in time which is thought to affect the numerical behavior of impact simulations. Unlike the short-range contact force which is a linear function of the contact strain and thus gradually increases in magnitude with increasing strain, the viscous contact was seen to jump from zero to its largest magnitude at first contact detection. The discontinuity results from the lack of contact strain dependence in the velocity-dependent term, and the viscous contact force likely reaches its maximum value at the first detection of contact due to the high impact speeds simulated.

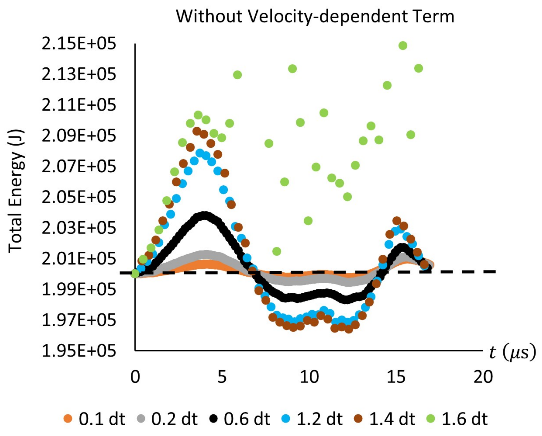
To investigate the discontinuity's effect on the impact simulation, we consider the two-node impact of figure 4.4 using the material properties outlined in section 3, the critical damping constant $\tilde{c}_{\text{crit}}$, and an incident relative speed of 1 km/s. The viscous contact force discontinuity in time for this two-node interaction is shown in figure 4.8 along with a comparison to the smoother contact force that results from removal of the velocity-dependent force term.

Because the viscous contact force is maximized at the first point of contact detection, we found that the simulation contact energy, the potential energy associated with the contact forces, is sensitive to the node separation distance at which contact between two nodes is first detected. Call this separation distance $\xi_{\text{c,AB}}$. To demonstrate the sensitivity, in figure 4.9(a) we vary the initial node separation $\xi_{\text{AB}}(0)$ of the previously defined two-node impact so that, due to the finite timestep, contact detection will occur at different values of $\xi_{\text{c,AB}}$. As the figure shows, variation in $\xi_{\text{AB}}(0)$ and thus $\xi_{\text{c,AB}}$ does indeed result in significant contact energy variation. The simulation timestep used in figure 4.9(a) is 0.2dt, and as shown in figure 4.10, the simulation does seem to be stable at this timestep. With smaller timesteps, nodes outside the contact boundary at one timestep cannot travel far past a contact boundary before being detected in the next timestep, so $\xi_{\text{c,AB}}$ is always very nearly $\tilde{\delta}$. Figure 4.9(b) shows that with the reduced timestep of 0.05dt and corresponding reduced variability in $\xi_{\text{c,AB}}$, the variation in contact energy is reduced. Interestingly, counter to what was seen in the many-node impact, the two-node impact seems to have a maximum stable simulation timestep that is more comparable to that of the many-node non-viscous impact which from figure 4.7 is about 1.5dt.

5. Conclusion. For many years, peridynamic modelers have struggled with the parameter-tuning balancing act required in the modeling of high-speed impact via the short-range contact force method. In this work, inspired by the strain-rate hardening phenomenon, we have demonstrated that the addition of a velocity-dependent contact force term has the potential to alleviate this problem. Using a fixed damping parameter with a clear physical interpretation, we showed that a viscous contact force can prevent interpenetration over a vast range of impact speeds- from 100 m/s to 10 km/s (and perhaps even as high as 50 km/s). However, it should be noted that this work has reported entirely on simulations with highly simplified material models (namely, the bond-based PMB model) and simple, 2D geometries. Further work must be done to evaluate the nature of the viscous contact force behavior in more realistic settings.



(a)



(b)

FIG. 4.7. *Use of total system energy variation with timestep as metric to evaluate stability of the impact simulation (a) with the velocity-dependent term and (b) without velocity dependence; $dt \approx 0.28\mu s$*

The numerical behavior of the proposed viscous contact force was also investigated. With the understanding that the increased stiffness of the viscous contact force may adversely affect stability, we compared qualitative estimates of the maximum stable timesteps with and without the addition of the velocity-dependent force and found the stability criteria to be influenced by the velocity-dependence- though not drastically. We also found that due to its discontinuous nature in time, the viscous force method may require a smaller timestep than that required for stability to converge. This suggests that a more appropriate contact model may involve the product of the contact strain and relative node velocity which would remove the discontinuity. This method has been explored theoretically [3] and even applied to peridynamic simulations [6], but does not yet seem to have been studied for use in high-speed impact.

REFERENCES

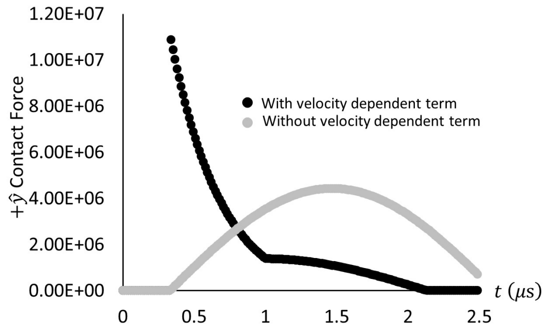
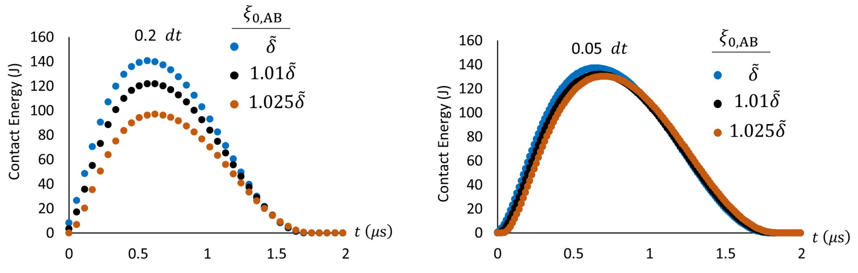


FIG. 4.8. *Illustration of the discontinuity in contact force history due to the velocity-dependent term. The smoother contact force evolution corresponding to removal of the velocity-dependent term included for comparison*



(a) (b)

FIG. 4.9. *Contact energy sensitivity to $\xi_{0,AB}$ to evaluate $\xi_{c,AB}$ sensitivity for (a) 0.2dt timestep and (b) 0.05dt timestep*

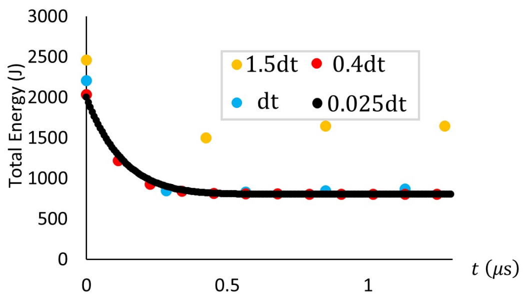


FIG. 4.10. *Stability of the two-node simulation*

[1] S. CHAN AND I. TUBA, *A finite element method for contact problems of solid bodies-part i. theory and validation*, Int J Mech Sci, 13 (1971), pp. 615–625.

[2] T. HUGHES, R. TAYLOR, J. SACKMAN, A. CURNIER, AND W. KANOKNUKULCHAI, *A finite element method for a class of contact-impact problems*, Comput Methods Appl Mech Eng, 8 (1976), pp. 249–276.

[3] K. HUNT AND F. CROSSLEY, *Coefficient of restitution interpreted as damping in vibroimpact*, J Appl Mech, 42 (1975), pp. 440–445.

[4] D. LITTLEWOOD, *Progress and challenges in computational peridynamics*, Sandia Labs Presentation, (2015).

[5] M. Parks, D. Littlewood, J. Mitchell, and S. Silling, *Peridigm users' guide v1.0.0*, Sandia Labs Technical Report, SAND2012-7800 (2012).

[6] T. Rabczuk and H. Ren, *A peridynamics formulation for quasi-static fracture and contact in rock*, Eng Geol, 225 (2017), pp. 42–48.

[7] S. Silling, *Reformulation of elasticity theory for discontinuities and long-range forces*, J Mech Phys Solids, 48 (2005), pp. 175–209.

[8] S. Silling and E. Askari, *A meshfree method based on the peridynamic model of solid mechanics*, Comput Struct, 83 (2000), pp. 1526–1535.

[9] S. Silling, M. Epton, O. Weckner, J. Xu, and E. Askari, *Peridynamic states and constitutive modeling*, J of Elasticity, 88 (2007), pp. 151–184.

[10] M. Tupek and R. Radovitsky, *An extended constitutive correspondence formulation of peridynamics based on nonlinear bond-strain measures*, J Mech and Phys of Solids, 65 (2014), pp. 82–92.