

# CSRI SUMMER PROCEEDINGS 2023

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

**Editors:**

S.K. Seritan and B.W. Reuter  
Sandia National Laboratories

November 28, 2023



SAND2023-13916R

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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 report describes objective technical results and analysis.

Any subjective views or opinions that might be expressed in the report do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831

Telephone: (865) 576-8401  
Facsimile: (865) 576-5728  
E-Mail: reports@adonis.osti.gov  
Online ordering: <http://www.doe.gov/bridge>

Available to the public from

U.S. Department of Commerce  
National Technical Information Service  
5285 Port Royal Rd  
Springfield, VA 22161

Telephone: (800) 553-6847  
Facsimile: (703) 605-6900  
E-Mail: orders@ntis.fedworld.gov  
Online ordering: <http://www.ntis.gov/ordering.htm>



# 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 2023.** In 2023, the CSRI summer program was executed in a hybrid fashion for the second year in a row. Most students were able to at least visit the Albuquerque or Livermore campuses during the internship. The summer program included students from 1400 – the Center for Computing Research (CCR) and 8700 – the Center for Computation & Analysis for National Security (CANS). This year’s program included the traditional *Summer Seminar Series* and *Summer Proceedings* and continued the fourth annual *Lightning Talks* (previously *Virtual Poster Blitz*). Our participation in the Computer Science and Analysis Institute (CSA) continued again this year with our partners in the Engineering Sciences Summer Institute (ESSI). Together, we provided four sessions on professional development with Dr. Jacquilyn Weeks from Word Tree Consulting, supported in-person site visits for some virtual interns, and ran an escape room and various facilities tours at Sandia and in the surrounding areas. We also hosted speakers from the Center of Cyber Defenders (CCD) and ESSI programs for joint talks during our summer seminar series.

**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, as well as different career paths available in the national labs. We extend our deepest thanks to the staff who spoke at the 2023 Seminar Series. These speakers and their talk titles are listed in Table 2.1. In addition to the traditional Summer Seminar Series, we also hosted a panel on “Getting the most from your summer internship” on 06/12/2023. We would like to thank Eric Phipps (1465), Jennifer Loe (1465), Jacob Davis (8734), and Tiernan Casey (8738) for serving as panelists and giving our interns their perspectives on mentor/mentee responsibilities, how to network as an intern, effective communication, and technical/professional development.

Table 2.1: List of talks and speakers in the 2023 Seminar Series

Date	Name	Org	Title
6/12	Jen Gaudioso	1400	An Introduction to Sandia, its Missions, and Computational Research at the Lab
6/21	Siva Rajamanickam	1465	Algorithm-Hardware-Software Co-design for Science Simulations and Machine Learning
6/28	Teresa Portone	1463	A brief survey of uncertainty quantification
7/5	Jackie Chen	8300	High performance computing and data science towards zero carbon fuels for power generation
7/12	Mark Taylor	1446	An overview of global climate change modeling and DOE's Energy Exascale Earth System Model
7/19	Marissa Adams Kris Beckwith	1684 1443	Verification, Validation, and Uncertainty Quantification for Computational Plasma Physics
7/26	Philip Kegelmeyer (CCD)	8730	The Counter-Intuitive Properties of Ensembles for Machine Learning (or, Democracy Defeats Meritocracy)
8/2	Laura Swiler	1400	Use of data analysis, dimension reduction, and machine learning in climate attribution
8/9	Sarah Scott (ESSI)	8751	We Didn't Start the Fire... But We Have to Deal with the Decomposing Organic Materials
8/16	Moe Khalil	8734	Bayesian transfer learning and model selection with application to machine-learning and physics-based modeling
8/23	Elise Walker Troy Shilt	1442 1513	Physics-informed machine learning for feature discovery, cross-modal inference, and causal inference
8/30	Ashe Miller	8739	Estimating the performance of quantum circuits by propagating errors

**3. Lightning Talks.** In preparation for the proceedings, *lightning talks* were held on 7/20/2023 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 lightning talk event are published under SAND2023-09319PE.

**4. 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 2023 CSRI Proceedings have been organized into three categories: *Computational & Applied Mathematics*, *High Performance & Post-Moore Computing* and *Machine Learning*.

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: Jonas Actor ♦ Andrew Baczewski ♦ Uma Balakrishnan ♦ Kris Beckwith ♦ Sylvain Bernard ♦ Patrick Blonigan ♦ Ron Brightwell ♦ Lincoln Collins ♦ Mary Alice Cusentino ♦ Eric Cyr ♦ Megan Dahlhauser ♦ Benjamin Feinberg ♦ Jacob Fustos ♦ Christian Glusa ♦ James Goff ♦ Anthony Gruber ♦ Mamikon Gulian ♦ Joey Hart ♦ Eric Ho ♦ Jonathan Hu ♦ Paul Kuberry ♦ Kim Liegeois ♦ Jay Lofstead ♦ William Maxwell ♦ Megan McCarthy ♦ Justin Owen ♦ Ravi Patel ♦ Mauro Perego ♦ Kara Peterson ♦ Nathan Porter ♦ Teresa Portone ♦ Tim Proctor ♦ Siva Rajamanickam ♦ Jaideep Ray ♦ Denis Ridzal ♦ Khachik Sargsyan ♦ Chris Siefert ♦ Ember Sikorski ♦ Laura Swiler ♦ Mark Taylor ♦ Irina Tezuar ♦ Aidan Thompson ♦ Kevin Thompson ♦ Heidi Thornquist ♦ Anh Tran ♦ Craig Ulmer ♦ Rebekah White ♦ Mitchell Wood ♦ Tian Yu Yen ♦ Kevin Young.

**Deepest Thanks.** We would like to thank all students and mentors for their extraordinary patience and dedication, especially given the hybrid nature of the intern program. We would also like to thank the program managers for the CSRI Summer Intern Program, Michael Wolf (1465) and Jerry McNeish (8734); as always, 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 Lisa Mahkee, Sandra Portlock, Hailey Poole, Sabrina Ahumada, and many others. We would also like to extend a very special thank you to Erin Stelter of the CSA program, Gaby Bran Anleu of the ESSI program, Sasha Safonov and Steven Barker of the CCD program, and Jacob Davis for his role in setting up tours. Their assistance greatly enriched the experiences of our interns this year, and we are deeply appreciative of their help.

S.K. Seritan  
B.W. Reuter

November 28, 2023



## Table of Contents

### Preface

*S.K. Seritan and B.W. Reuter* . . . . . iii

### Articles . . . . . 1

#### I. Computational & Applied Mathematics

*S.K. Seritan and B.W. Reuter* . . . . . 1

##### Clustering for Extreme-Scale Finite Element Simulations

*E. Agyei-Kodie and G. Harper* . . . . . 2

##### Quantifying Uncertainties in Ablation Models for Hypersonic Flight

*R. Bandy, M. Sands, and T. Portone* . . . . . 16

##### Fluid Dynamics on a Rotating Sphere

*M. Chiwere, P.A. Bosler, and G.B. Wright* . . . . . 30

##### Enhancing Linear Solvers through Advanced Matrix Clustering

*N. Etienne, G. Harper, and C. Siebert* . . . . . 42

##### Optimization-Based Approaching for Coupling Projection-Based ROMs

*E. Hawkins, P. Bochev, and P. Kuberry* . . . . . 54

##### Random Householder-Cholesky QR Factorization

*A.J. Higgins, E.G. Boman, D.B. Szyld, and I. Yamazaki* . . . . . 70

##### Optimal Experimental Design for Bayesian Linear Inverse Problems

*N. Neuberger, B.G. van Bloemen Waanders, and A. Alexanderian* . . . . . 79

##### Sensitivity Analysis for Geological Disposal Safety Assessment

*H. Rosso, T. Portone, and R. White* . . . . . 94

##### Magnetohydrodynamics

*D. Sharp and B.G. van Bloemen Waanders* . . . . . 109

##### Randomized Eigensolver Implementation and Spectral Graph Partitioning

*H. Switzer, J. Loe, and E.G. Boman* . . . . . 121

##### Patch-Based Adaptive Relaxation Methods

*A. Voronin, R.S. Tuminaro, L.N. Olsen, and S. MacLachlan* . . . . . 135

#### II. High Performance & Post-Moore Computing

*S.K. Seritan and B.W. Reuter* . . . . . 147

##### ECMF Back End Pipeline and Environment Container Build Program

*S. Toribio, R. Milewicz, J. Teves, J. Willenbring, and J. Frye* . . . . . 148

##### Optimizing Sparse Matrix-Vector Multiplication on a RISC-V GPU

*L. Cooper and K. Pedretti* . . . . . 158

##### Evaluating RISC-V “V” Extensions with Chipyard and FireSim

*G. Dube and K. Pedretti* . . . . . 164

##### System-Level Provenance Tools for Practical Use

*S. Grayson and R. Milewicz* . . . . . 172

##### Space-Constrained Quantum Query Complexity of Boolean Functions

*B. Holman, J. Kallaughher, and O. Parekh* . . . . . 182

##### Performance Portable Rigid Body Dynamics

*A. Johansson and S. Moore* . . . . . 193

##### Fast Kokkos Code from Sparse Tensor Algebra in Python using MLIR

*A. Khan, B. Kelley, K. Liegeois, and S. Rajamanickam* . . . . . 202

##### Is RMA The Way?

*D. Mishler, J. Ciesko, S. Olivier, and G. Bosilca* . . . . . 210

##### Offloading Node-Local Filesystems in HPC Environments

*J. Shawger and M. Curry* . . . . . 218

Simple Statistical Model for Randomized Benchmarking Data <i>S. Surti and T. Proctor</i>	231
Benchmarking of Quantum Algorithms for Chemistry <i>A.Q. Wilber-Gauthier and S.K. Seritan</i>	244
Are Hard Quantum Problems Also Classically Hard? <i>J. Yirka, J. Kallaugh, and O. Parekh</i>	255
<b>III. Machine Learning</b>	
<i>S.K. Seritan and B.W. Reuter</i>	266
Unsupervised Learning Sensor Fusion of Turbulent Flow Data <i>B. Dalman and M. Barone</i>	267
Multi-Fidelity Modeling with Fourier Feature Networks <i>O. Davis, G. Geraci, and M. Motamed</i>	278
Machine-Learned Potentials for Nickel-Platinum Catalysts <i>I. Furrick, M. Wood, and A. Hensley</i>	294
Inverse Sobolev-Type Inequalities for Tanh Neural Networks <i>E. Huynh, T. Meissner, P. Bochev, and P. Kuberry</i>	303
Uncertainty Quantification of Model Form Error <i>E. Islas Quinones and K.A. Maupin</i>	323
Neural ODE Flux Surrogate for Coupled Transmission Problems <i>R. Pawar, P. Bochev, and J. Owen</i>	336
Machine Learned Potentials for Grain Boundary Segregation <i>K. Polifrone, H. Bayat, J.M. Goff, and W. Xu</i>	353
Deep Least-Squares Method for Stokes Equation <i>T. Meissner, E. Huynh, P. Kuberry, and P. Bochev</i>	365
Organizing Training Data for Machine Learned Potentials <i>C. Mullen and E.L. Sikorski</i>	376
Random Forest Model Form Error <i>J. Ngangmeni and K.A. Maupin</i>	382
Coupling of PINNs via Schwarz Alternating Method <i>W. Snyder, I. Tezaur, and C.R. Wentland</i>	389
Machine Learned Potentials for Hafnium Ceramics <i>V.A. Vera Cruz and E.L. Sikorski</i>	411
Coordinate Encoding in Multi-Fidelity Neural Emulators <i>C. Villatoro, G. Geraci, and D.E. Schiavazzi</i>	422

# 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 or analyze new algorithms, discretize and solve partial differential equations, couple multiphysics systems of equations, and analyze sensitivity & quantify uncertainty in complex systems.

1. *Agyei-Kodie* and *Harper* propose a method to cluster mesh elements to speed up finite element simulations.
2. *Bandy, Sands* and *Portone* quantify and reduce uncertainties in multi-fidelity ablation models for hypersonic flight.
3. *Chiwere, Bosler* and *Wright* develop a method for solving equations of motion for a 2D fluid on a rotating sphere.
4. *Etienne, Harper* and *Siefert* enhance the performance of linear solvers through advanced matrix clustering techniques.
5. *Hawkins, Bochev* and *Kuberry* develop a PDE-constrained optimization method for coupled advection-diffusion equations using reduced order modeling.
6. *Higgins, Boman, Szyld* and *Yamazaki* test the performance of the Randomized Householder-Cholesky QR decomposition with multiple sketch matrices.
7. *Neuberger, van Bloemen Waanders* and *Alexanderian* investigate the use of optimal experimental design for steady-state convection-diffusion.
8. *Rosso, Portone* and *White* use variance-based sensitivity analysis to perform uncertainty quantification of geological disposal safety assessment.
9. *Sharp and van Bloemen Waanders* discuss advances in inference, uncertainty quantification, and optimal control for magnetohydrodynamics.
10. *Switzer, Loe* and *Boman* implement a random eigensolver in Anasazi and demonstrate its' utility for spectral graph partitioning.
11. *Voronin, Tuminaro, Olson* and *MacLachlan* introduce a novel patch-based relaxation approach for multigrid solvers.

S.K. Seritan  
B.W. Reuter

November 28, 2023

## CLUSTERING AND STRUCTURE-DETECTING R-ADAPTIVITY FOR EXTREME-SCALE FINITE ELEMENT SIMULATIONS

EUGENE AGYEI-KODIE \* AND GRAHAM HARPER†

**Abstract.** In the world of computational sciences, there is a growing need for efficient and accurate numerical simulations. In order for one to facilitate the execution of extensive finite element applications within the constraints of finite computational resources, a variety of strategies have been employed. One prominent approach involves the utilization of techniques like sum factorization, which leverages structural characteristics to effectively compress data. This method accomplishes data compression by transforming  $d$ -dimensional datasets into  $d$  individual instances of 1-dimensional data, thereby optimizing the utilization of available resources. In this study, we introduce an analogous methodology that centers on the compression of basis values across mesh cells, with a primary focus on the similarity between said mesh cells. Our approach harnesses the principles of r-adaptive techniques to augment the compressibility of data residing on the mesh, thereby contributing to a more streamlined and efficient data representation. It combines clustering and structure-detection techniques while taking the reference to physical mapping into account. To achieve this objective, our methodology encompasses a two-step process. Initially, we employ mesh clustering, followed by the application of Sequential Quadratic Programming (SQP) optimization on the individual clusters by making use of the de Rham complex diagram. Our primary goal is to speed up finite element simulations by reusing finite element basis functions where we can reduce redundant calculations and significantly cut down computational time without compromising simulation accuracy. In later sections, we present numerical results that demonstrate the success of our approach in accelerating finite element simulations while maintaining accuracy. In conclusion, the integration of clustering, structure-detection, and optimization techniques enables us to create structured mesh cells that facilitate the reuse of finite element basis functions. This advancement has the potential to enhance the efficiency and performance of various computational applications, bringing us closer to faster and more accurate finite element simulations.

**1. Introduction.** In most linear finite element method (FEM) applications, the main assembly kernel involves computing many integrals of the form

$$\int_T k(\mathbf{x}) D^{\alpha_1} \psi_i(\mathbf{x}) D^{\alpha_2} \psi_j(\mathbf{x}) d\mathbf{x}, \quad (1.1)$$

where  $T$  is a domain that corresponds to a cell in a mesh  $\mathcal{T}_h$ ,  $k(\mathbf{x})$  is some (not necessarily scalar) problem coefficient,  $\alpha_1$  and  $\alpha_2$  are alpha-indices for the derivative operator  $D$ , and  $\psi_i(\mathbf{x})$  and  $\psi_j(\mathbf{x})$  are FEM basis functions [5]. There are many ways for computing such integrals, but we focus on one of the most common cases where the integral is transformed to some reference geometry and computed via quadrature, which is the case in high performance libraries such as Intrepid [2] and deal.II [1]. This requires constructing a reference to physical mapping  $\Phi_T : \hat{T} \rightarrow T$  as shown in Figure 1.1, where  $\hat{T}$  is some reference or “ideal” geometry, and  $T$  is the domain the integral is to be computed over. There are many ways to increase the speed of a FEM application, including the use of sum factorization techniques [13] for assembly, which reduces the amount of data throughput required while increasing the amount of computation required. This reduction in data storage also increases the ability to run larger problems with fixed computational resources. Some work related to linear solvers has also been performed, showing reuse of data for a class of solvers and preconditioners greatly speeds up computations [7]. However, another approach for speeding up FEM assembly is to reuse data such as transformed basis values, which also greatly decreases necessary data throughput, but comes with almost no additional computation. Figure 1.2 was presented at the SIAM MDS22 conference [8], showing the performance-relevant impacts of compressing transformed FEM basis values in an application called MrHyDE at Sandia National Laboratories. The main takeaway is that basis

---

\*Sandia National Laboratories, Michigan State University (agyeikod@msu.edu)

†Sandia National Laboratories, (gcharpe@sandia.gov)

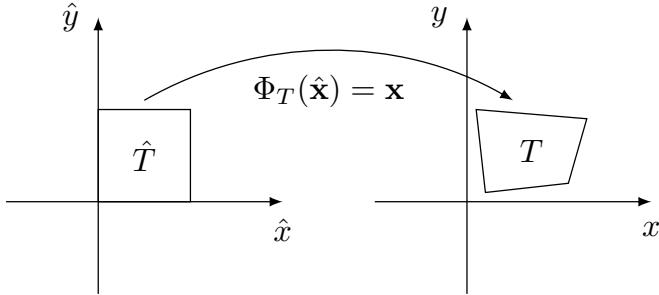


FIG. 1.1. Illustration of the reference to physical mapping  $\Phi_T$  for a mesh cell  $T$  and reference cell  $\hat{T}$ .

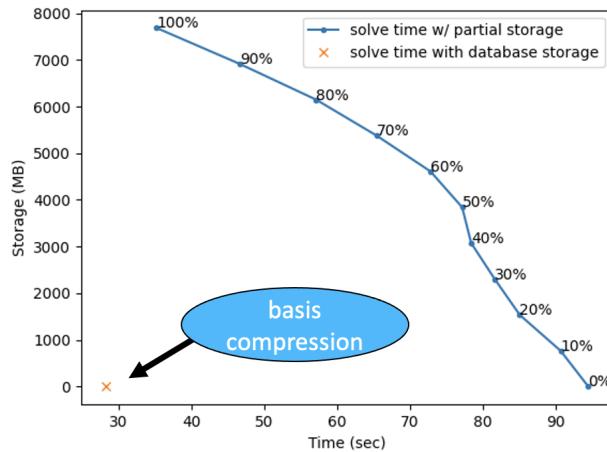


FIG. 1.2. Comparison of speed and memory costs for recomputing  $X\%$  of physical FEM basis values versus compressing the values based on similarities in the Jacobian in a simple application using MrHyDE

compression may run faster than storing 100% of the basis in memory because the amount of required memory throughput is greatly reduced. This is especially relevant in the context of modern simulations utilizing accelerators such as GPUs, as transferring large amounts of data to accelerators is much more expensive than transferring the same data to a CPU. The goal of this paper is to use mesh clustering techniques to group mesh cells by similarity and then perform r-adaptive mesh optimization to improve the mesh structure for the purpose of FEM data compression. This compression ultimately increases the speed of a FEM simulation by simply moving mesh nodes. Section 2 expands on the relevant background for FEMs, and Section 3 expands on the clustering and optimization aspects. Finally, we present results in Section 4 and conclude in Section 5.

## 2. Background on Finite Element Methods.

**2.1. Reference to Physical Mapping.** Now, in order to properly transform the original integral presented in (1.1), we start with some background. The reference to physical mapping  $\Phi_T : \hat{T} \rightarrow T$  discussed in Section 1 is one of the most important tools in FEMs, as it maps reference coordinates on  $\hat{x}$  on the ideal domain  $\hat{T}$  to physical coordinates  $x$  on the physical domain  $T$ . Using such a mapping to evaluate integrals on the reference domain allows one to only define and integrate a single set of functions on the reference domain  $\hat{T}$  as opposed to constructing new functions for every domain in a mesh. When a domain is

understood, we relate  $\hat{\mathbf{x}} = [\hat{x}, \hat{y}]^T$  and  $\mathbf{x} = [x, y]^T$  by the mapping  $\Phi$  where  $\mathbf{x} = \Phi(\hat{\mathbf{x}})$  is the pushforward and  $\hat{\mathbf{x}} = \Phi^{-1}(\mathbf{x})$  is the pullback. Similarly, for a function  $f : T \rightarrow \mathbb{R}$ , we define  $\hat{f} : \hat{T} \rightarrow \mathbb{R}$  where  $\hat{f}(\hat{\mathbf{x}}) = f(\Phi_T(\hat{\mathbf{x}})) = f(\mathbf{x})$  using the previously defined relation. To better understand how integration works with these pullbacks, we briefly study where integrated quantities live with respect to the de Rham complex, which was first seriously utilized in FEMs by Bossavit [4] and has been developed extensively since then (c.f. §2.1.4, §2.5.6 of [3]). The de Rham complex connects function spaces by differential operators and defines pullback transformations by utilizing the derivative of the reference to physical mapping  $J_T := \nabla\Phi_T$ , which we may also denote by  $J$  when the domain is not relevant. A crude de Rham complex is presented below, where the function spaces are identified in the top row, the function spaces are connected by the differential operators on the arrows, and transformations underneath each space correspond to the operator to premultiply when transforming coordinates.

$$\begin{array}{ccccccc} \text{space:} & HGRAD & \xrightarrow{\nabla} & HCURL & \xrightarrow{\nabla \times} & HDIV & \xrightarrow{\nabla \cdot} & HVOL \\ \text{pullback:} & I & & J^{-T} & & J\det(J)^{-1} & & \det(J)^{-1} \end{array} \quad (2.1)$$

Note that  $I$  is the identity transformation, and  $J^{-T}$  is the Jacobian inverse-transpose, and we have used the short names HGRAD, HCURL, HDIV, and HVOL in place of the Sobolev spaces  $H^1(\Omega)$ ,  $H(\text{curl}, \Omega)$ ,  $H(\text{div}, \Omega)$ , and  $L^2(\Omega)$ , respectively.

As an example, let  $f \in H^1(T)$  be a scalar valued HGRAD function on  $T$ . Combined with the change of coordinates formula from calculus which produces  $\det(J(\hat{\mathbf{x}}))$ , we evaluate the integral on  $\hat{T}$  by pulling back the coordinates and premultiplying the appropriate pullback from (2.1), yielding

$$\int_T f(\mathbf{x}) d\mathbf{x} = \int_{\hat{T}} I \hat{f}(\hat{\mathbf{x}}) \det(J_T(\hat{\mathbf{x}})) d\hat{\mathbf{x}}. \quad (2.2)$$

However, to integrate  $\nabla f$  instead, which now lives in the HCURL space according to the de Rham complex, one must instead transform by premultiplying by the  $J^{-T}$  operator

$$\int_T \nabla f(\mathbf{x}) d\mathbf{x} = \int_{\hat{T}} J_T^{-T}(\hat{\mathbf{x}}) \nabla \hat{f}(\hat{\mathbf{x}}) \det(J_T(\hat{\mathbf{x}})) d\hat{\mathbf{x}}. \quad (2.3)$$

This multiplication of the reference function values and the mapping from the de Rham complex  $J_T^{-T} \nabla \hat{f}(\hat{\mathbf{x}})$  is referred to as physical function value, as the Jacobian and reference function values ultimately determine the integral.

**2.2. Accelerating Finite Element Computations by Jacobian Compression.** In light of the fact that the reference function values and Jacobian-based mapping from the de Rham complex determine the physical basis values, we now discuss the compression of function values based on analyzing the Jacobian. In (2.2) and (2.3), we observed the HGRAD function data stored by a FEM application on two mesh cells  $T_1$  and  $T_2$  is given by  $J_{T_1}^{-T} \nabla \hat{f}(\hat{\mathbf{x}})$  and  $J_{T_2}^{-T} \nabla \hat{f}(\hat{\mathbf{x}})$ . Therefore, if  $J_{T_1}$  and  $J_{T_2}$  are equal, the application need only store one set of function values and not both. In order to measure when these matrix-valued functions are equal, a scalar measure for similarity across the domain  $\hat{T}$  is defined by computing a norm and integral.

$$d(J_{T_1}, J_{T_2}) = \int_{\hat{T}} \|J_{T_1}(\hat{\mathbf{x}}) - J_{T_2}(\hat{\mathbf{x}})\|_F d\hat{\mathbf{x}}. \quad (2.4)$$

For a given  $\delta$ , we say  $J_{T_1}$  and  $J_{T_2}$  are approximately equal if  $d(J_{T_1}, J_{T_2}) < \delta$ . If they are approximately equal, we may then only store the finite element data for  $J_{T_1}^{-T} \nabla \hat{f}(\hat{\mathbf{x}})$  on  $T_1$

and reuse it for  $T_2$ . For reasonable choices of  $\delta$ , the amount of error introduced by such substitution is negligible while increasing the speed of the application; however, the error also depends on the conditioning of the Jacobian due to the inversion.

Therefore, by reusing Jacobians and reducing the amount of data throughput required by an application to compute necessary integrals, we are able to drastically increase the speed at which an application runs. With this in mind, given some  $\delta > 0$ , we now apply compression to the mesh as a preprocessing step. We start by constructing a subset of mesh cells  $\mathcal{T}_B \subseteq \mathcal{T}_h = \{T_1, T_2, \dots, T_n\}$  and a mapping  $\phi : \{1, \dots, n\} \rightarrow \{1, \dots, |\mathcal{T}_B|\}$  such that  $\phi(i) = j$  implies  $d(J_{T_i}, J_{T_j}) < \delta$ . This implies the function values for  $i$  may be replaced by the stored function values for  $j$ . We may also abbreviate  $\phi(i)$  as a vector  $\vec{\phi}$  with entries  $\phi_i$  less formally. We say  $\mathcal{T}_B$  is the compressed mesh database, and  $\phi$  is the lookup function, and define  $m = |\mathcal{T}_B|$ . Furthermore, we say the compression ratio of a mesh is  $\frac{|\mathcal{T}_h| - |\mathcal{T}_B|}{|\mathcal{T}_h|} = \frac{n-m}{n}$ . Our goal now is to increase the compressibility of a mesh, as it increases the speed at which a FEM simulation runs. However, before we attempt to increase the compression ratio of a mesh, we briefly provide some concrete examples for computing these Jacobian-based quantities on quadrilateral meshes.

**2.3. Computations on Quadrilaterals.** For the remainder of the paper, we will focus on quadrilaterals, as the tensor product geometry structure often allows similarly shaped mesh cells to be packed near each other as opposed to simplicial geometry. For quadrilaterals, we define the idea reference geometry  $\hat{T}$  by the points  $(0, 0), (1, 0), (1, 1), (0, 1)$ , where a counter-clockwise node ordering convention is assumed. For another quadrilateral given by the points  $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ , we look for a mapping of the form

$$\Phi_T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{bmatrix} a_1 + a_2\hat{x} + a_3\hat{y} + a_4\hat{x}\hat{y} \\ b_1 + b_2\hat{x} + b_3\hat{y} + b_4\hat{x}\hat{y} \end{bmatrix}$$

for unknown scalars  $a_i$  and  $b_i$ ,  $i = 1, 2, 3, 4$ . This equation may be solved by specifying the conditions  $\Phi_T([0, 0]^T) = [x_1, y_1]^T$ ,  $\Phi_T([1, 0]^T) = [x_2, y_2]^T$ ,  $\Phi_T([1, 1]^T) = [x_3, y_3]^T$ , and  $\Phi_T([0, 1]^T) = [x_4, y_4]^T$ . This yields

$$\Phi_T \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{bmatrix} x_1 + (x_2 - x_1)\hat{x} + (x_4 - x_1)\hat{y} + (x_1 - x_2 + x_3 - x_4)\hat{x}\hat{y} \\ y_1 + (y_2 - y_1)\hat{x} + (y_4 - y_1)\hat{y} + (y_1 - y_2 + y_3 - y_4)\hat{x}\hat{y} \end{bmatrix}.$$

In terms of the formula we provided for quadrilaterals above, we may compute the Jacobian by

$$J_T(\hat{\mathbf{x}}) = \begin{bmatrix} (x_2 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{y} & (x_4 - x_1) + (x_1 - x_2 + x_3 - x_4)\hat{x} \\ (y_2 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{y} & (y_4 - y_1) + (y_1 - y_2 + y_3 - y_4)\hat{x} \end{bmatrix}.$$

However, after more careful examination, we see we may split the Jacobian as a constant term plus a spatially varying outer product.

$$J_T(\hat{\mathbf{x}}) = \begin{bmatrix} x_2 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_4 - y_1 \end{bmatrix} + \begin{bmatrix} x_1 - x_2 + x_3 - x_4 \\ y_1 - y_2 + y_3 - y_4 \end{bmatrix} \begin{bmatrix} \hat{y} & \hat{x} \end{bmatrix}.$$

The terms for  $(x_1 - x_2 + x_3 - x_4)$  and  $(y_1 - y_2 + y_3 - y_4)$  measure the total distortion of a quadrilateral. For squares and rectangles, these terms are zero. Assuming distortion is not extreme – an assumption which often applied to many large-scale applications, and is similarly to the conditioning of the Jacobian mentioned in the previous section – we may disregard those terms and define

$$\tilde{J}_T = \begin{bmatrix} x_2 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_4 - y_1 \end{bmatrix}. \quad (2.5)$$

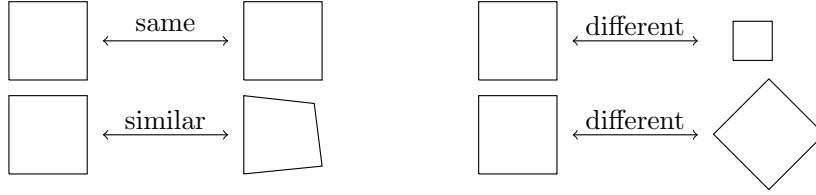


FIG. 3.1. *Visual depiction of similar and dissimilar quadrilaterals for clustering*

This significantly reduces the complexity of computations later at the cost of some error. It is not difficult to see this simplified quadrilateral Jacobian coincides with the Jacobian for the triangle defined by using vertices 1,2,4 of the quadrilateral.

### 3. Clustering-Based Mesh Optimization.

**3.1. Clustering for Mesh Cells.** Thanks to optimization techniques such as the ones presented in [6], it's possible to adapt and modify the locations of mesh nodes using optimization techniques without changing the underlying topology. We will use a modification of this approach in order to optimize the mesh for compressibility. To ease the strain on large-scale optimization problems, we would like to first group mesh cells into groups of similar shapes. The most difficult aspect is defining a distance metric between two mesh cells  $T_1$  and  $T_2$ . A general idea of similarity and dissimilarity metrics are shown in Figure 3.1. From a clustering standpoint, it is not sufficient to simply consider the nodes of the quadrilaterals, as the nodes themselves do not capture "shape" and therefore will not benefit FEM computations.

However, one metric that preserves information up to translation is the Jacobian of the reference to physical mapping,  $J_T(\hat{\mathbf{x}})$ . In fact, following the same calculations in Section 2.3, if  $T_1$  is a translation of  $T_2$ , it is easy to verify  $J_{T_1} \equiv J_{T_2}$ . Therefore, we will proceed by treating these Jacobians as a surrogate for shape information of the mesh cells. Now, there is one problem that has been glossed over. In FEMs, these Jacobians are often not symbolically defined as functions, but rather evaluated at the quadrature points on the reference cell  $\hat{T}$ . This greatly increases the amount of information required for a clustering algorithm, as there is now a  $2 \times 2$  matrix for every quadrature point. Therefore, we define the distance between two quadrilateral mesh cells by the Frobenius norm of the difference of the approximate Jacobians defined in (2.5),

$$d(T_1, T_2) = \|\tilde{J}_{T_1} - \tilde{J}_{T_2}\|_F. \quad (3.1)$$

We will utilize the  $k$ -means algorithm, which requires a fixed input  $n_C \in \mathbb{N}$  for the number of clusters. The  $k$ -means algorithm alternates computing distances to assign data points (Jacobians) to the nearest cluster means (Jacobian means), and then updating the cluster means using the new cluster assignments. Some of the first descriptions of this approach are presented in [12, 9], with modifications and modern approaches described in [11]. Ultimately, it returns a vector of cluster assignments  $\vec{\varphi}$  where  $\varphi_i = j$  implies mesh cell  $i$  is assigned to cluster  $j$ . Let  $C := \{C_1, \dots, C_{n_C}\}$  be a partitioning of the mesh cells according to the clustering results

$$C_k := \{T_i : \varphi_i = k\}.$$

Then we may compute the in-cluster variance by

$$\sigma_k^2 = \sum_{T_i \in C_k} d(T_i, \bar{T}_k)^2 \quad (3.2)$$

where  $\bar{T}_k$  is the mean for cluster  $k$ . We then define the total cluster variance by

$$\begin{aligned}\sigma^2 &= \sum_{k=1}^{n_C} \sigma_k^2 \\ &= \sum_{k=1}^{n_C} \sum_{T_i \in C_k} d(T_i, \bar{T}_k)^2 \\ &= \sum_{k=1}^{n_C} \sum_{T_i \in C_k} \|\tilde{J}_{T_i} - \overline{\tilde{J}_{T_k}}\|_F^2.\end{aligned}\tag{3.3}$$

The total cluster variance is an extremely useful metric for determining the quality of clustering approximations, and we plot the total cluster variance in Section 4 to search for the optimal number of clusters using the elbow method [11]. While this clustering approach allows for an initial decomposition of the mesh cells into sets of similar shape, we must now take the resulting clusters and enhance the compressible structure.

**3.2. Sequential Quadratic Programming Optimization.** Once some initial structure has been detected in a mesh, we now move to enhance the structure. We utilize a trust-region sequential quadratic programming (SQP) optimization [10] method to take clusters obtained from a clustering method and optimize mesh shape within a single cluster. In particular, we use the mesh correction framework developed in [6], which utilizes a volume-constrained optimization problem. This approach uses r-adaptivity, which moves mesh nodes without modifying topological connections, to obtain a new mesh with target properties. Then, for each cluster, we compute the cluster volume by summing the individual cell volumes  $V_k = \sum_{T_i \in C_k} |T_i|$ , which depends on the mesh node locations. We then define

a mean cell volume by dividing by the number of cells in the cluster  $\bar{V}_k = V_k/|C_k|$ , and we perform a volume-constrained optimization which seeks to find the most similar mesh which has equal volumes across a cluster. However, we also fix nodes on the boundaries of clusters so that all clusters may be optimized in parallel. We then optimize as follows: Given a mesh with coordinates located at  $\tilde{\mathbf{p}}$ , seek new mesh cell coordinates  $\mathbf{p}$  satisfying

$$\left\{ \begin{array}{l} \min_{\mathbf{p}} \|\tilde{\mathbf{p}} - \mathbf{p}\|^2 \\ \text{s.t. } |T_i(\mathbf{p})| - \bar{V}_{\phi_i}(\tilde{\mathbf{p}}) = 0, \quad i = 1, \dots, n \\ \text{and } \mathbf{p}|_{\partial C_k} = \tilde{\mathbf{p}}, \quad k = 1, \dots, n_C \end{array} \right. \tag{3.4}$$

We believe that by detecting cells with similar shape and then optimizing them to have equal volumes, we can greatly enhance the compressibility of a mesh.

**4. Results.** Using clustering analysis to find patterns and organize data is well-recognized in our field. Our study focuses on applying clustering analysis to three types of meshes: a circle mesh, a square mesh with poor topology, and an electromagnetics metamaterial mesh.

In this section, we present the results from our experiments, emphasizing the unique findings for each mesh type. We also explain how we used the elbow curve method to figure out the best number of clusters.

#### Experiment 1 A circle mesh.

We began our numerical experiments with the circle mesh, a basic geometric configuration. In Figure 4.1, you can see the problem we are discussing on the left side. Our aim was to find a configuration that best matched the mesh's underlying cluster structure. Additionally, we wanted to see how clustering analysis could reveal structural characteristics in

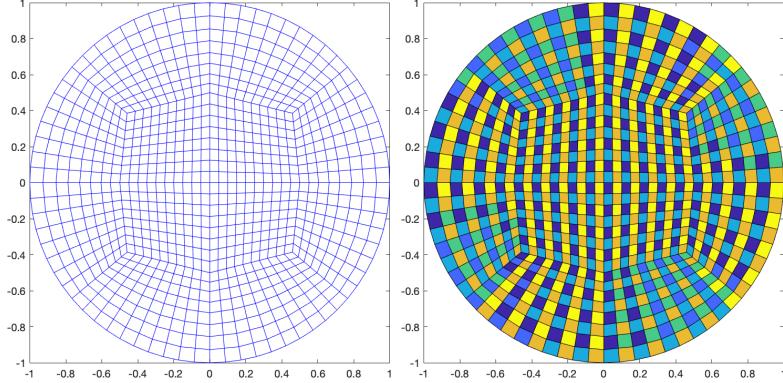


FIG. 4.1. *Experiment 1* (left) mesh and (right) clustering checkerboard produced by alternating node orientations with 6 clusters

this simple mesh. In Figure 4.1, you will find the output before we made any adjustments to the orientation.

In our experiments, we discovered insights about the structure and orientation of the basic mesh. One key insight was that the node orderings in the mesh alternated between neighboring cells, which you can see on the left side of Figure 4.2. This caused differences in Jacobians that produced the aforementioned checkerboard pattern. To fix this, we permuted the nodes clockwise until the first node is below and left of the mesh cell centroid, and the result is on the right side of Figure 4.2. There were minor unexplainable imperfections, but the orientations are now mostly consistent.

Next, we applied the algorithm to the circle mesh, which has 896 quadrilaterals, using different cluster counts: 10, 12, 20, 24, 40, and 80, as seen in Figure 4.4. We noticed a clear bend in the elbow curve in Figure 4.3, indicating an optimal cluster count. This approach allowed us to explore how different cluster configurations affected the results, going beyond the number of cluster examples in Experiment 1. We found that using 30 clusters provided the best fit for the circle mesh. Our choice was based on the distinctive curvature we observed in the elbow curve, strategically chosen to pinpoint the transition point in the graph. Additionally, we considered the spatial arrangement and coherence of the mesh, where clusters of similar mesh cells were close together. We also took into account the presence of rotational symmetry within the mesh structure, supporting our selection.

However, our analysis did encounter some challenges. We noticed a problem with the local orientation of nodes during the evaluation. This led to some neighboring clusters having alternating colors, as you can see in Figure 4.4. This issue affected the overall visual representation of the clustering results.

In response, we conducted an optimization on the mesh, this time with 22 clusters. Surprisingly, we found that the mesh was not significantly affected. Even at a tolerance of  $10^{-7}$ , the mesh's compressibility remained at 48.55% before and after optimization.

It seems that this mesh may already be well-structured and cannot compress further. To illustrate the optimization, Figure 4.5 shows the effect using 3 clusters instead of 22. In the case of 3 clusters, the mesh structure did decrease. In summary, we have found that addressing the orientation issue is crucial for improving the clustering results. This observation highlights the need for further investigation and correction of the orientation. It is clear that refining and investigating the orientation algorithm is essential for enhancing the accuracy and reliability of the clustering outcomes.

**Experiment 2** A structured square domain with poor mesh topology.

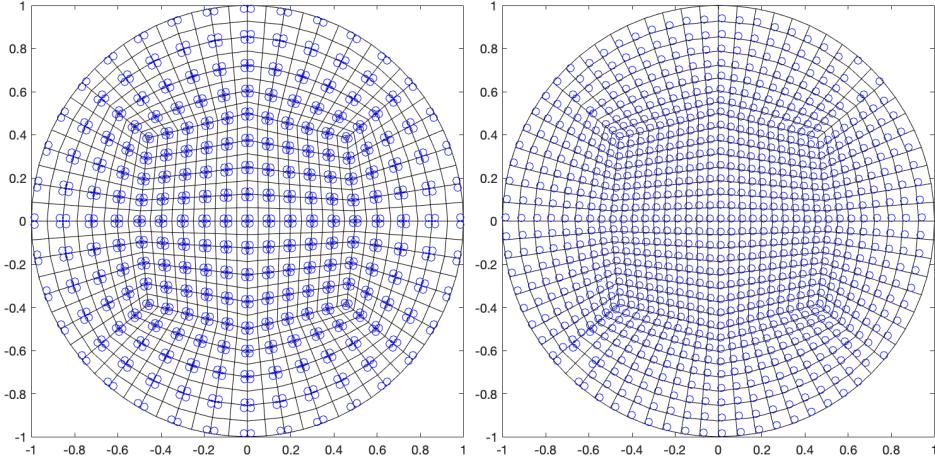


FIG. 4.2. Local mesh orientations (left) before and (right) after correction, with the first local node indicated by a blue circle

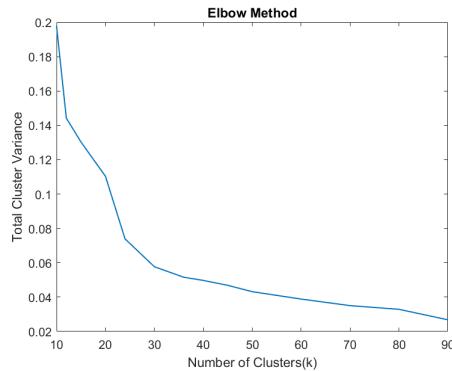


FIG. 4.3. Experiment 1 (left) mesh and (right) cluster variances versus number of clusters

This experiment focuses on a simple square mesh with poor topology, a situation sometimes encountered in practical applications. This mesh consists of 99 quadrilaterals. During our cluster analysis, we used 2, 3, 4, 5, 6, 7, and 8 clusters, which you can see in Figure 4.7.

This unique mesh configuration brought a new dimension to our study. Our findings reveal why these results occurred during the clustering experiments. Notably, we observed that the elbow curve for this mesh, as shown in Figure 4.6, differed from the one in Experiment 1. Interestingly, the elbow curve associated with this mesh exhibited a straight-line characteristic rather than the usual curve seen in Experiment 1. This unique feature of the elbow curve posed a challenge when determining the optimal number of clusters, as you can see in Figure 4.6.

Despite the unusual characteristics of the mesh, our study suggests that configuring it with 8 clusters could offer the most suitable representation. This choice was based on several factors. We considered the presence of rotational symmetry in the data and the level of locality within it, which you can see in the lower-left part of Figure 4.6. Although the elbow curve didn't have a clear bend, these geometric characteristics in the data guided us to select 8 clusters.

We also need to acknowledge a minor orientation error that we encountered in Experi-

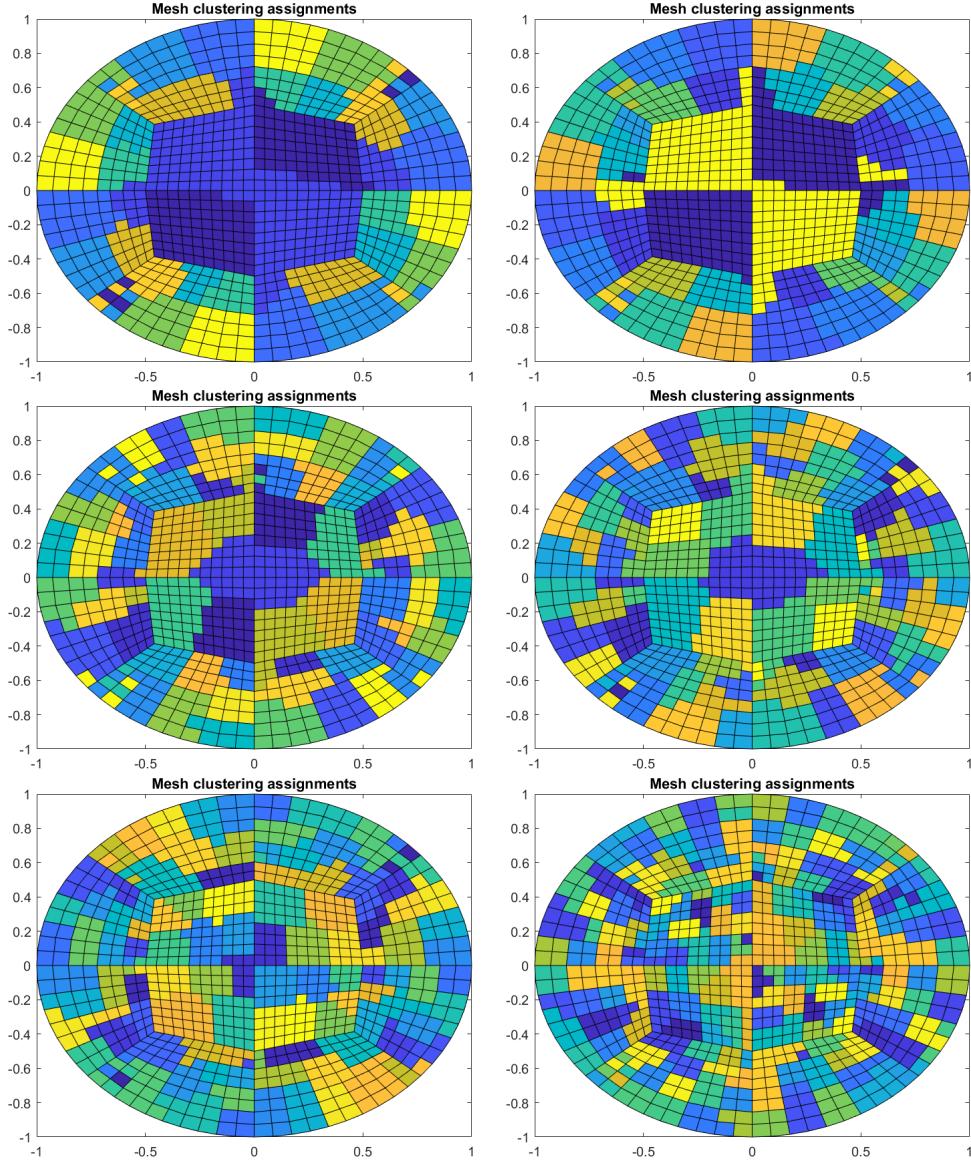


FIG. 4.4. *Experiment 1 clustering results using 10, 12, 20, 24, 40, and 80 clusters*

ment 1, which influenced our results in this experiment. However, despite these limitations, our findings emphasize the significance of these fundamental geometric attributes in the data. They underscore the potential for valuable insights once we rectify the orientation error, and our choice of 8 clusters is an initial step in capturing the mesh's inherent structural properties.

In conclusion, improving our algorithm, particularly by addressing the orientation error, holds promise for revealing more profound insights into the mesh's representation and structure.

### **Experiment 3** An electromagnetics metamaterial mesh.

In this mesh experiment, we used the clustering algorithm to cluster 6772 quadrilaterals.

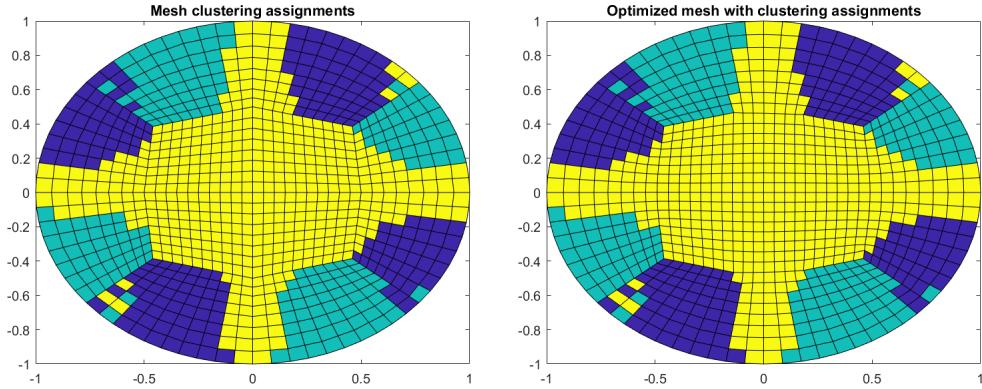


FIG. 4.5. *Experiment 1 with 3 clusters demonstrating (left) unoptimized and (right) optimized node positions*

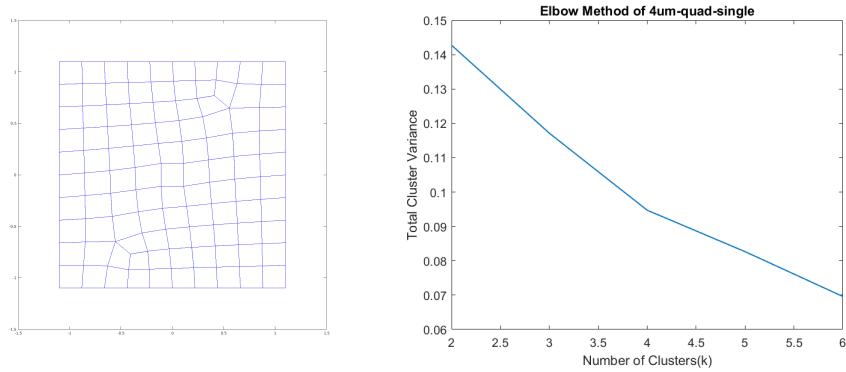


FIG. 4.6. *Experiment 2 (left) mesh and (right) cluster variances versus number of clusters*

This mesh has unique electromagnetic properties, making it an interesting case for clustering analysis.

However, due to its inherent irregularities and complexities, this mesh presents distinct challenges. Despite these challenges, the clustering results provided valuable insights into the mesh's structural limitations and suggested potential optimization strategies. We tested different cluster configurations, including 10, 12, 20, 24, 40, and 80 clusters, shown in Figure 4.9. The best fit for the mesh turned out to be 38 clusters, as indicated by the elbow curve in Figure 4.8. Similar to Experiment 2, determining the right number of clusters was crucial in capturing the key features of the electromagnetic response.

The insights gained from this study, especially regarding the nature of this mesh, provide essential knowledge about the reliability of our methodology in real-world, less idealized situations.

While the elbow curve for the electromagnetics mesh did not show a clear inflection point like the circle mesh, our analysis still revealed a significant insight: the presence of recurring patterns within the mesh. This reinforces the suitability of our chosen cluster count, as seen in Figure 4.8.

This observation indicates that our clustering method effectively recognizes and captures these repetitive patterns, treating them as consistent and coherent elements. This is evident in the various graphics in Figure 4.9.

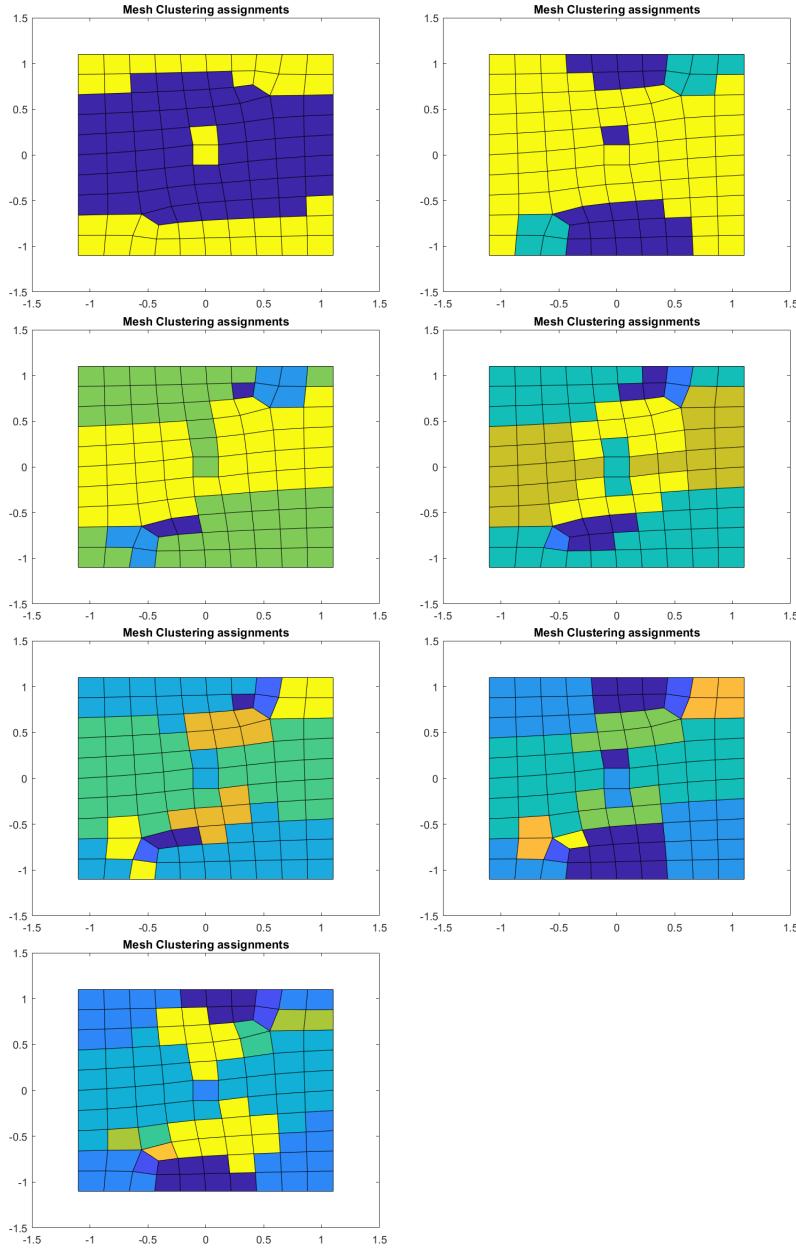


FIG. 4.7. *Experiment 2 clustering results using 2, 3, 4, 5, 6, 7, and 8 clusters*

In summary, even though the elbow curve was not as pronounced, the discovery of repeating patterns supports the effectiveness of our clustering approach in capturing essential details. This finding contributes to a deeper understanding of the mesh's underlying organization and validates the utility of our clustering methodology in handling challenging configurations.

**5. Conclusion.** Our study presents a new idea in the field of computational sciences by combining clustering and structure-detection methods within r-adaptive techniques for

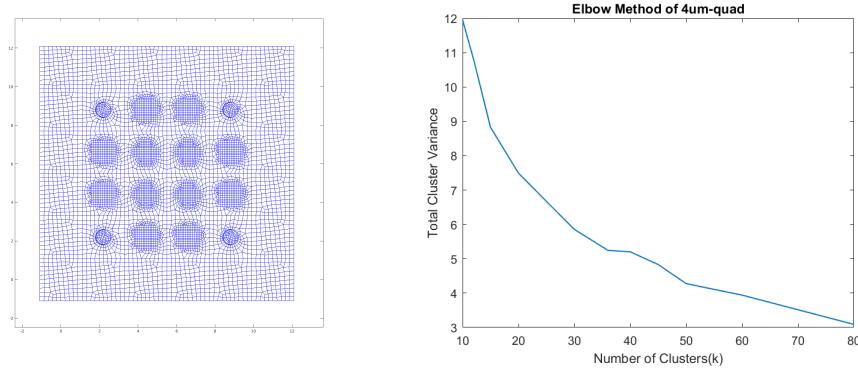


FIG. 4.8. *Experiment 3 (left) mesh and (right) cluster variances versus number of clusters*

finite element simulations. This approach aims to blend clustering and structure-detection techniques while considering physical mapping. Throughout our research, we examined how clustering algorithms can bring together mesh elements based on their geometric and physical characteristics. To enhance our understanding of mesh optimization, we employed the intricate de Rham complex diagram. By implementing the SQP optimization method, we organized mesh cells for uniformity and homogeneity, which ultimately accelerates finite element simulations while maintaining accuracy. This approach minimizes redundant calculations and optimizes the use of finite element basis functions, reducing computational overhead without compromising the integrity of the simulations.

Our study showed it is possible to optimize mesh shape based on clustering results. There were some difficulties related to local mesh node orientations, which we believe can be improved to benefit the optimization and clustering algorithms. We also believe some modifications to the SQP objective such as a cluster uniformity penalty may help improve results. Our study's significance reaches beyond the presented experiments, offering the potential to improve computational efficiency in various applications. It represents a substantial step towards achieving precise finite element simulations by combining clustering, structure-detection, and optimization techniques.

In conclusion, our research suggests that with minor adjustments, we can move closer to achieving highly accurate numerical simulations. Further studies can address these minor orientations and contribute to this goal.

The authors would like to thank Denis Ridzal for valuable discussions, guidance, and feedback. Eugene Agyei-Kodie was supported by the Sustainable Horizons Institute (SHI) Sustainable Research Pathways (SRP) program. Graham Harper was supported by the Sandia National Laboratories Laboratory Directed Research & Development Program. 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 grant DE-NA-0003525. 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.

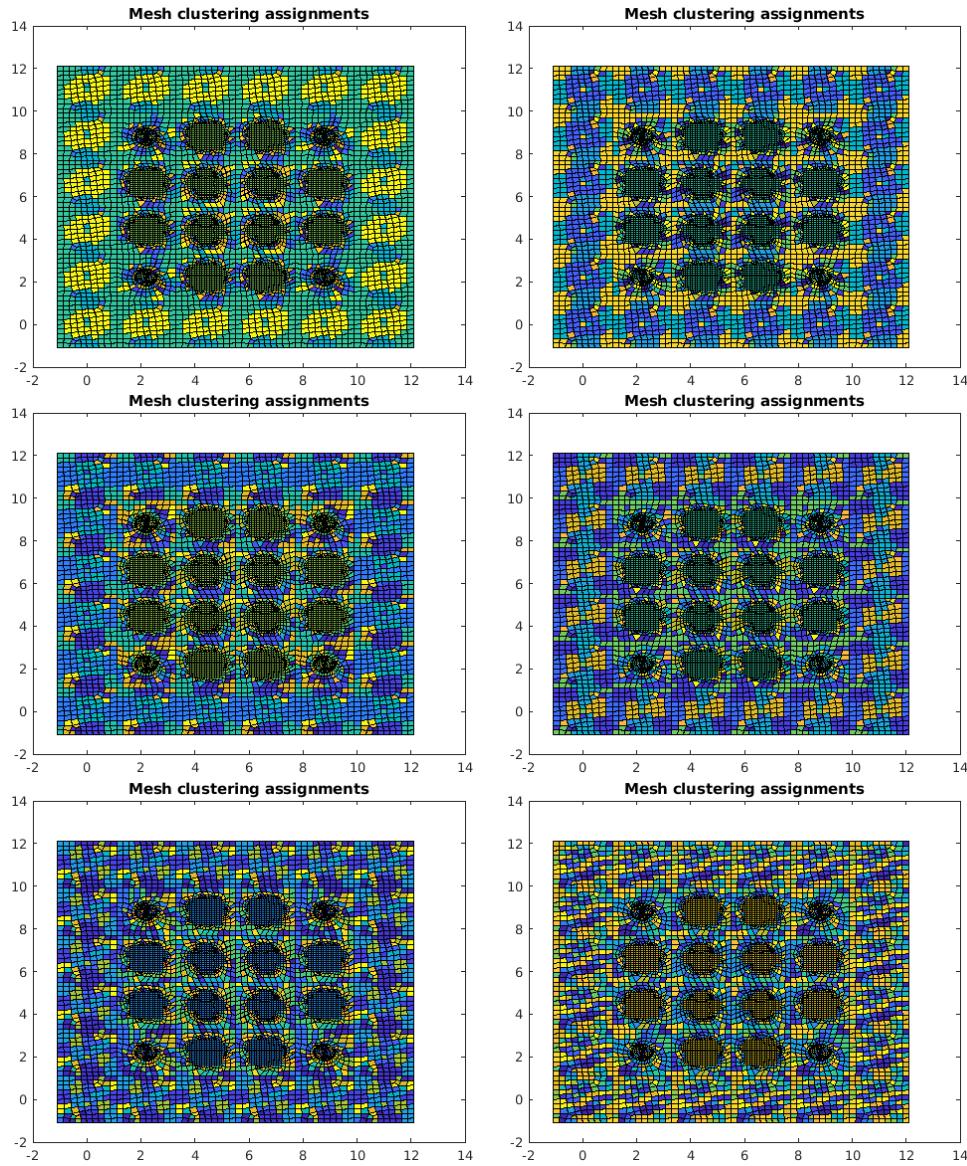


FIG. 4.9. *Experiment 3 clustering results using 10, 12, 20, 24, 40, and 80 clusters*

## REFERENCES

- [1] W. BANGERTH, R. HARTMANN, AND G. KANSCHAT, *deal.II—a general-purpose object-oriented finite element library*, ACM Transactions on Mathematical Software (TOMS), 33 (2007), pp. 24–es.
- [2] P. BOCHEV, H. C. EDWARDS, R. C. KIRBY, K. PETERSON, AND D. RIDZAL, *Solving pdes with intrepid*, Scientific Programming, 20 (2012), pp. 151–180.
- [3] D. BOFFI, F. BREZZI, M. FORTIN, ET AL., *Mixed finite element methods and applications*, vol. 44, Springer, 2013.
- [4] A. BOSSAVIT, *Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism*, IEE Proceedings A (Physical Science, Measurement and Instrumentation, Management and Education, Reviews), 135 (1988), pp. 493–500.
- [5] S. C. BRENNER AND L. R. SCOTT, *The mathematical theory of finite element methods*, Springer, 2008.

- [6] M. D'ELIA, D. RIDZAL, K. J. PETERSON, P. BOCHEV, AND M. SHASHKOV, *Optimization-based mesh correction with volume and convexity constraints*, Journal of Computational Physics, 313 (2016), pp. 455–477.
- [7] G. HARPER AND R. TUMINARO, *Compression and reduced representation techniques for patch-based relaxation*, 2023.
- [8] G. HARPER AND T. WILDEY, Data Compression Techniques for Large-Scale Memory-Bound Finite Element Applications. Poster presented at SIAM Mathematics of Data Science Conference 2022, San Diego, California, United States.
- [9] J. A. HARTIGAN AND M. A. WONG, *Algorithm as 136: A k-means clustering algorithm*, Journal of the royal statistical society. series c (applied statistics), 28 (1979), pp. 100–108.
- [10] M. HEINKENSCHLOSS AND D. RIDZAL, *A matrix-free trust-region sqp method for equality constrained optimization*, SIAM Journal on Optimization, 24 (2014), pp. 1507–1541.
- [11] A. K. JAIN, M. N. MURTY, AND P. J. FLYNN, *Data clustering: a review*, ACM computing surveys (CSUR), 31 (1999), pp. 264–323.
- [12] S. LLOYD, *Least squares quantization in pcm*, IEEE transactions on information theory, 28 (1982), pp. 129–137.
- [13] J. MORA AND L. DEMKOWICZ, *Fast integration of dpg matrices based on sum factorization for all the energy spaces*, Computational Methods in Applied Mathematics, 19 (2019), pp. 523–555.

## QUANTIFYING UNCERTAINTIES IN ABLATION MODELS FOR HYPersonic FLIGHT SIMULATION

RILEIGH BANDY\*, MICHAEL SANDS †, AND TERESA PORTONE‡

**Abstract.**

During hypersonic flight, air reacts with a planetary entry vehicle's thermal protection system (TPS), creating reaction products that result in the ablation of the TPS. Accurately modeling this phenomenon is critical to assessing TPS performance. Novel finite-rate gas-surface chemistry models are advancing state-of-the-art in TPS ablation modeling, but computational tractability will require important chemical species to be neglected in some cases. We leverage sensitivity analysis to analyze uncertainties in a high-fidelity carbon TPS ablation model and inform the formation of a low-fidelity model where species have been neglected. We then describe a plan for developing a model-form uncertainty representation to characterize uncertainties and reduce errors caused by omissions in the low-fidelity model.

### 1. Nomenclature.

$A_v$	=	Avogadro constant, $\text{mol}^{-1}$
$B$	=	total active site density, $\text{mol}\cdot\text{m}^{-2}$
$E$	=	Activation energy, J
$h$	=	Planck's constant, $\text{J}\cdot\text{s}$
$k_b$	=	Boltzmann constant, $\text{J}\cdot\text{K}^{-1}$
$m$	=	mass of an atom or molecule of a species, kg
$P$	=	Partial pressure
$R$	=	Universal gas constant, $\text{J}\cdot\text{K}^{-1}\cdot\text{mol}^{-1}$
$S$	=	Adsorption selectivity
$S_{\mathbf{u}}^g$	=	Sobol' main effect index for input parameter group $\mathbf{u}$
$T$	=	Temperature, K
$T_{\mathbf{u}}$	=	Sobol' total effect index for input parameter group $\mathbf{u}$
$x$	=	Longitudinal distance from the stagnation point, m
$\gamma$	=	Pre-exponential factor
<i>Subscript</i>		
$G$	=	Gas

**2. Introduction.** During hypersonic flight, a planetary entry vehicle can experience large heat fluxes for an extended period of time. Therefore, a thermal protection system (TPS) is required. Due to high flow speeds, the gas near the planetary entry vehicle's surface remains in chemical nonequilibrium, and reactive atomic species reach the surface and damage the TPS heat shield [14]. To protect the vehicle, ablative carbon-based shields are commonly used, where oxidation of carbon is central to the ablation [12]. At high enough temperatures, ionization reactions and carbon sublimation are possible. The carbon TPS ablators react with atomic species in the air (i.e., atomic oxygen (O) and atomic nitrogen (N) atoms) [8], or solid carbon turns into gas [19]. These gas-surface reactions need to be quantified to guarantee an adequately thick TPS has been applied to the hypersonic vehicle's surface. Making the shield excessively thick adds unnecessary weight and cost, while catastrophic outcomes could result by applying a shield that is too thin.

---

\*Department of Computer Science, University of Colorado Boulder, rjbandy@sandia.gov

†Department of Chemistry, University of Colorado Boulder, msands@sandia.gov

‡Sandia National Laboratories, tporton@sandia.gov

Experimentally determining the performance of a TPS is extremely costly. For example, full-scale wind tunnel tests cost roughly \$100,000 per day [6], and hypersonic flight experiments cost \$11 – 286 million in 2009 [2]. Predictive computational models of TPS ablation are thus critical for TPS design. New trajectories may require finite-rate chemistry models, but they are much more costly than the previous equilibrium chemistry models [14]. Embedding a high-fidelity chemistry model into a practical computational fluid dynamics (CFD) simulation can become computationally expensive and could motivate the derivation of a low-fidelity chemistry model by reducing the number of species and reactions tracked in the model. While a low-fidelity model would be more computationally efficient, the reductions could cause significant model-form error (MFE), resulting in inaccurate predictions. Here we investigate how such a low-fidelity chemistry model could be derived and how the resulting MFE could subsequently be addressed by introducing an embedded enrichment to the low-fidelity model to improve its predictive accuracy.

Finite-rate gas-surface chemistry models predict the mass production rate of carbon monoxide (CO), which is a quantity of interest (QoI) for determining the amount of ablated carbon from the surface. However, these finite-rate gas-surface models are new, quickly evolving, and there are several potential sources of uncertainty (i.e., neglected species, reaction forms, and reaction parameterization). In this work, we take a well-known finite-rate gas-surface chemistry model, the air-carbon-ablation (ACA) model [14], as our high-fidelity model.

The paper is organized as follows. In Section 3, we review the ACA model and introduce potential uncertainties within the model. In Section 4, inputs and the QoI are defined. In Section 5, we perform sensitivity analysis on the ACA model, and in Section 6 results from sensitivity analysis are used to form a low-fidelity model. In Section 7 we propose embedding a model-form uncertainty representation in the low-fidelity model to characterize uncertainty and reduce error caused by neglected species. Concluding remarks are given in section 8.

**3. High-Fidelity Model.** The ACA model, shown in Table 3.1, involves 20 reaction mechanisms that focus on the reactions between the carbon surface and gas species [14]. The model describes oxidation by O and molecular oxygen,  $O_2$ ; nitridation by N; and surface catalyzed recombination of both O and N. Prata et al. note “[t]he actual reactions between O or N atoms and carbon, especially highly defected carbon, are undoubtedly more complicated than those used in the model” [14]. Simplifying assumptions have been made in the ACA model, which introduce uncertainties and potential discrepancies between the ACA model and experimental observations.

The ACA model involves four types of reactions: adsorption, desorption, Eley-Rideal, and Langmuir-Hinshelwood as shown in Figure 3.1, and Arrhenius rates are determined by empirical function for each reaction type. The mechanisms associated with these reaction types are discussed below.

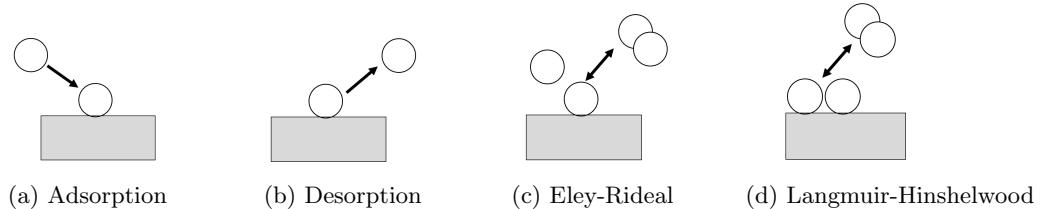


Fig. 3.1: Depictions of the four types of reactions in the ACA model.

Table 3.1: ACA gas-surface chemistry model reaction set.

Number	Reaction	$S$ or $\gamma$	$E/R$	Type
1	$O + (s) \rightarrow O(s)$	0.3	0	Adsorption
2	$O(s) \rightarrow O + (s)$	1.0	44277	Desorption
3	$O + O(s) + C(b) \rightarrow CO + O + (s)$	100.0	4000	Eley-Rideal
4	$O + O(s) + C(b) \rightarrow CO_2 + (s)$	1.0	500	Eley-Rideal
5	$O + (s) \rightarrow O^*(s)$	0.7	0	Adsorption
6	$O^*(s) \rightarrow O + (s)$	1.0	96500	Desorption
7	$O + O^*(s) + C(b) \rightarrow CO + O + (s)$	1000	4000	Eley-Rideal
8	$O^*(s) + O^*(s) \rightarrow O_2 + 2(s)$	$10^{-3}$	15000	Langmuir-Hinshelwood
9	$O(s) + O(s) \rightarrow O_2 + 2(s)$	$5.0 \times 10^{-5}$	15000	Langmuir-Hinshelwood
10	$N + (s) \rightarrow N(s)$	1.0	2500	Adsorption
11	$N(s) \rightarrow N + (s)$	1.0	73971	Desorption
12	$N + N(s) + C(b) \rightarrow CN + N + (s)$	1.5	7000	Eley-Rideal
13	$N + N(s) \rightarrow N_2 + (s)$	0.5	2000	Eley-Rideal
14	$N(s) + N(s) \rightarrow N_2 + 2(s)$	0.1	21000	Langmuir-Hinshelwood
15	$N(s) + C(b) \rightarrow CN + (s)$	$10^8$	20676	Eley-Rideal
16	$O_2 + 2(s) \rightarrow 2O(s)$	1.0	8000	Adsorption
17	$O_2 + O(s) + C(b) \rightarrow CO + O_2 + (s)$	100.0	4000	Eley-Rideal
18	$O_2 + O(s) + C(b) \rightarrow CO_2 + O + (s)$	1.0	500	Eley-Rideal
19	$O_2 + 2(s) \rightarrow 2O^*(s)$	1.0	8000	Adsorption
20	$O_2 + O^*(s) + C(b) \rightarrow CO + O_2 + (s)$	1000.0	4000	Eley-Rideal

A gas-phase reactant must first adsorb on the surface before forming a product through gas-surface interaction. Adsorption reactions involve a gas-phase reactant  $G$ , where  $G \in \{O, N, O_2\}$ , and an empty surface site ( $s$ ). Using the Arrhenius law, the adsorption rate coefficient is

$$k_{ad} = \frac{F_G}{B^k} S \exp\left(-\frac{E}{RT}\right), \quad (3.1)$$

where

$$F_G = \frac{1}{4} \sqrt{\frac{8k_b T}{\pi m_G}}$$

is one-fourth the mean thermal speed of the gas species (in mole per meter). The power of  $B$  depends on the gas species being adsorbed. For O and N,  $k = 1$  and for  $O_2$ ,  $k = 2$ . The flux of the gas species to the surface is  $F_G[G]$ , where  $[G]$  is the concentration of the gas-phase species (in mole per cubic meter) and  $m_G$  is the mass of the gas species. The surface sites that the gas species can adsorb to are modeled by  $B$ , the total active site density. The adsorption selectivity (or the sticking probability) is  $S$ , which is commonly set to one unless the gas species could bind to the surface in multiple ways. For example, the ACA model captures weakly-bonded and strongly-bonded O to the surface. There is a 30% probability that O will weakly bond resulting in  $O(s)$ , and a 70% chance that O will strongly bond resulting in  $O^*(s)$ . The activation energy is experimentally derived from molecular beam data [14]. Finally, the rate of adsorption is  $k_{ad}[G][(s)]$ , where  $[(s)]$  is the surface density of the empty reactive sites.

The adsorbed gas species desorbs from the surface with a rate coefficient of

$$k_{des} = \frac{2\pi m_G k_b^2 T^2}{A_v B h^3} \exp\left(-\frac{E}{RT}\right). \quad (3.2)$$

The rate of desorption is  $k_{des}[G(s)]$ , where  $[G(s)]$  is the surface density of the adsorbed species. The adsorbed species are  $G(s) \in \{O(s), O^*(s), N(s)\}$  or weakly-bonded O, strongly-bond O, and bonded N with no distinction of bonding strength. Note, oxygen can form  $O_2$  in the air, but adsorbing to the surface would require two empty sites. In this model, the molecule would have to split to attach to the surface. This is theoretically true for nitrogen as well, and the ACA model does allow for the production of molecular nitrogen,  $N_2$ . However, it does not account for the nitridation by  $N_2$ , which is a simplifying assumption.

Eley-Rideal and Langmuir-Hinshelwood reactions produce gas-phase products, and in the ACA model they more specifically remove a carbon atom from the surface. Eley-Rideal reactions involve one gas-phase reactant and one chemisorbed species, which is a gas species adsorbed to the surface. The rate of an Eley-Rideal reaction is given by  $k_{er}[G][G(s)]$ , where the rate coefficient is

$$k_{er} = \frac{F_G}{B} \gamma \exp\left(-\frac{E}{RT}\right). \quad (3.3)$$

Theoretically,  $\gamma \in [0, 1]$ , but the ACA model experimentally sets the  $\gamma$ -values to match molecular beam data. Therefore, that constraint is relaxed to  $\gamma \in [0, \infty)$ , and a  $\gamma$ -value larger than one signifies that the reaction is not an elementary reaction, which is a well-defined, single-step reaction resulting from a single collision between two or rarely three molecules or ions. Instead, there are intermediate reactions untracked by the model, which potentially increase uncertainty in the model.

Langmuir-Hinshelwood reactions involve two chemisorbed species. The rate of a Langmuir-Hinshelwood reaction is given by  $k_{lh}[G(s)][G(s)]$ , where the rate coefficient is

$$k_{lh} = \sqrt{\frac{A_v}{B}} F_{G,2D} \gamma \exp\left(-\frac{E}{RT}\right), \quad (3.4)$$

and

$$F_{G,2D} = \sqrt{\frac{\pi k_b T}{2m_G}}$$

is the mean thermal speed of the adsorbed species on the surface.

Of course, there are other possible reactions that the ACA model has omitted (e.g., formation of NO compounds in the air), but these 20 reactions provide us with a testbed for measuring the uncertainty in a finite-rate gas-surface chemistry model. The resulting

production rates of the ACA model for each species are

$$\begin{aligned}
\frac{d[O(s)]}{dt} &= k_1[O][(s)] - k_2[O(s)] - k_3[O][O(s)] - k_4[O][O(s)] \\
&\quad - 2k_9[O(s)]^2 + 2k_{16}[O_2][(s)]^2 - k_{17}[O_2][O(s)] - k_{18}[O_2][O(s)] \\
\frac{d[O^*(s)]}{dt} &= k_5[O][(s)] - k_6[O^*(s)] - k_7[O][O^*(s)] - 2k_8[O^*(s)]^2 \\
&\quad + 2k_{19}[O_2][(s)]^2 - k_{20}[O_2][O^*(s)] \\
\frac{d[CO]}{dt} &= k_3[O][O(s)] + k_7[O][O^*(s)] + k_{17}[O_2][O(s)] + k_{20}[O_2][O^*(s)] \\
\frac{d[CO_2]}{dt} &= k_4[O][O(s)] + k_{18}[O_2][O(s)] \\
\frac{d[O]}{dt} &= \frac{P_O}{A_v \sqrt{2\pi m_O k_b T}} - k_1[O][(s)] + k_2[O(s)] - k_4[O][O(s)] \\
&\quad - k_5[O][(s)] + k_6[O^*(s)] + k_{18}[O_2][O(s)] \tag{3.5} \\
\frac{d[O_2]}{dt} &= \frac{P_{O_2}}{A_v \sqrt{2\pi m_{O_2} k_b T}} + k_8[O^*(s)]^2 + k_9[O(s)]^2 \\
&\quad - k_{16}[O_2][(s)]^2 - k_{18}[O_2][O(s)] - k_{19}[O_2][(s)]^2 \\
\frac{d[N(s)]}{dt} &= k_{10}[N][(s)] - k_{11}[N(s)] - k_{12}[N][N(s)] - k_{13}[N][N(s)] \\
&\quad - 2k_{14}[N(s)]^2 - k_{15}[N(s)] \\
\frac{d[CN]}{dt} &= k_{12}[N][N(s)] + k_{15}[N(s)] \\
\frac{d[N]}{dt} &= \frac{P_N}{A_v \sqrt{2\pi m_N k_b T}} - k_{10}[N][(s)] + k_{11}[N(s)] - k_{13}[N][N(s)] \\
\frac{d[N_2]}{dt} &= k_{13}[N][N(s)] + k_{14}[N(s)]^2.
\end{aligned}$$

The total site density of the surface  $B$  is held constant resulting in

$$B = [O(s)] + [O^*(s)] + [N(s)] + [(s)],$$

and  $C(b)$  refers to a carbon atom in the bulk media of the surface.

A steady state solution to the ACA model is found by setting the rate of change in surface coverage equal to zero (i.e.,  $\frac{d[O(s)]}{dt} = \frac{d[O^*(s)]}{dt} = \frac{d[N(s)]}{dt} = 0$ ). In a large-scale CFD simulation of a hypersonic flow environment, the surface coverage is calculated at each flow solver timestep. Assuming the ACA model is at a steady state does not imply chemical equilibrium; it implies that the local surface coverage adapts instantly [14]. However, the surface chemistry is still changing over the course of the CFD simulation.

Let

$$\begin{aligned}
A_1 &= 2k_{16}[\text{O}_2] \\
B_1 &= k_1[\text{O}] \\
C_1 &= 2k_9 \\
D_1 &= k_2 + (k_3 + k_4)[\text{O}] + (k_{17} + k_{18})[\text{O}_2] \\
A_2 &= 2k_{19}[\text{O}_2] \\
B_2 &= k_5[\text{O}] \\
C_2 &= 2k_8 \\
D_2 &= k_6 + k_7[\text{O}] + k_{20}[\text{O}_2] \\
A_3 &= 0 \\
B_3 &= k_{10}[\text{N}] \\
C_3 &= 2k_{14} \\
D_3 &= k_{11} + k_{15} + (k_{12} + k_{13})[\text{N}].
\end{aligned}$$

Then,

$$[\mathcal{O}(s)] = \frac{2(A_1[(s)]^2 + B_1[(s)])}{D_1 + \sqrt{D_1^2 + 4C_1(A_1[(s)]^2 + B_1[(s)])}}, \quad (3.6)$$

$$[\mathcal{O}^*(s)] = \frac{2(A_2[(s)]^2 + B_2[(s)])}{D_2 + \sqrt{D_2^2 + 4C_2(A_2[(s)]^2 + B_2[(s)])}}, \quad (3.7)$$

$$[N(s)] = \frac{2B_3[(s)]}{D_3 + \sqrt{D_3^2 + 4C_3B_3[(s)]}}, \quad (3.8)$$

and

$$\begin{aligned}
[(s)] &= B - \frac{2(A_1[(s)]^2 + B_1[(s)])}{D_1 + \sqrt{D_1^2 + 4C_1(A_1[(s)]^2 + B_1[(s)])}} \\
&\quad - \frac{2(A_2[(s)]^2 + B_2[(s)])}{D_2 + \sqrt{D_2^2 + 4C_2(A_2[(s)]^2 + B_2[(s)])}} \\
&\quad - \frac{2B_3[(s)]}{D_3 + \sqrt{D_3^2 + 4C_3B_3[(s)]}}.
\end{aligned} \quad (3.9)$$

We solve for  $[(s)]$  using the bisection root-finding algorithm [18] with bounds  $[(s)] \in [0, B]$ . Then, that root is substituted into Equation (3.6), Equation (3.7), and Equation (3.8) to obtain the surface coverage of each bonding group. Finally, the steady-state concentrations can be solved by substituting the surface coverages and the molar concentration of the corresponding gas at the surface into Equation (3.5).

**4. Inputs and the Quantity of Interest.** To capture the uncertainties introduced by the high-fidelity model when used to make predictions about a planetary entry vehicle's flight, we use a simulation test-case from Mussoni et al. [10] that was previously outlined in [3, 4]. Specifically, the planetary entry vehicle's geometry is chosen as a 10 centimeter radius sphere -8° cone as shown in Figure 4.1, where  $x$  is the longitudinal distance from the

stagnation point (e.g., the tip of the nose). Freestream conditions correspond to an altitude of 40 kilometers with a freestream speed of 7 kilometers per second. A chemically reacting laminar boundary layer is applied, and the boundary conditions are in non-ablating, non-catalytic, radiative equilibrium. Therefore, the gas temperature at surface is equal to the surface temperature. The US3D data supplies the partial pressures of O, N, and O<sub>2</sub> and the temperatures for the high-fidelity model at discretized longitudinal distances from the stagnation point. In the test case, Candler shifts the resulting temperature to give realistic values of an ablating surface [3]. However, this was an arbitrary choice that we plan to investigate further. In this work, we do not shift the resulting temperature.

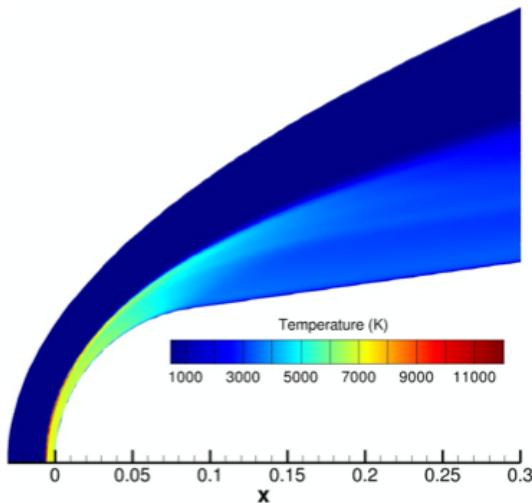


Fig. 4.1: Temperature contour of the flow field around the sphere-cone geometry for freestream velocity of 7 km/s at an altitude of 40 km.

In this study, we are interested in quantifying the recession of the carbon surface, which we estimate by the amount of carbon monoxide, CO, being produced in the Eley-Rideal and Langmuir-Hinshelwood reactions. Therefore, our QoI is the probability of CO production:

$$\text{CO probability} = \frac{[\text{CO}]}{O_{flux}}, \quad (4.1)$$

where [CO] is the equilibrium concentration of CO. The flux of oxygen is

$$O_{flux} = \frac{P_O}{A_v \sqrt{2\pi m_O k_b T}}, \quad (4.2)$$

where  $P_O$  is the partial pressure of O and  $m_O$  is O's molecular mass.

**5. Sensitivity Analysis.** We perform variance-based sensitivity analysis (VBSA) on the high-fidelity model to inform which species and reactions to omit when forming the low-fidelity model. VBSA measures the fraction of variance in a model prediction that can be attributed to each input, so it is commonly used to determine the impact parameter uncertainty has on predictions [15, 16, 17]. Specifically, we use grouped Sobol' indices, which measure the aggregate importance of a group of parameters (both their individual importance as well as the importance of their interactions). We group the model inputs by

type of surface interaction. In the ACA model, O can bond to the surface weakly or strongly (i.e., O(s) and O\*(s)) and N can bond to the surface (i.e., N(s)). The ACA reaction set and associated model parameters can be divided into three corresponding groups.

**5.1. Sobol' Indices.** We summarize Sobol' [17] to review standard Sobol' indices. Let  $\mathbf{X} = (X_1, X_2, \dots, X_n)$  be a vector of independent, uncertain input parameters. Then the model output can be decomposed as

$$f(\mathbf{X}) = f_\emptyset + \sum_{i=1}^n f_i(\mathbf{X}_i) + \sum_{i=1}^n \sum_{j=i+1}^n f_{i,j}(\mathbf{X}_i, \mathbf{X}_j) + \dots + f_{1,2,\dots,n}(\mathbf{X}), \quad (5.1)$$

where

$$\begin{aligned} f_\emptyset &= \mathbb{E}[f(\mathbf{X})] \\ f_i(X_i) &= \mathbb{E}_{\mathbf{X}_{\sim i}}[f(\mathbf{X})|X_i] - f_\emptyset \\ f_{i,j}(X_i, X_j) &= \mathbb{E}_{\mathbf{X}_{\sim i,j}}[f(\mathbf{X})|X_i, X_j] - f_i(X_i) - f_j(X_j) - f_\emptyset \\ &\vdots \\ f_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}) &= \mathbb{E}_{\mathbf{X}_{\sim \mathbf{u}}}[f(\mathbf{X})|\mathbf{X}_{\mathbf{u}}] - \sum_{\mathbf{v} \in \mathcal{P}(\mathbf{u})} f_{\mathbf{v}}(\mathbf{X}_{\mathbf{v}}). \end{aligned}$$

The  $\mathbf{X}_{\sim i}$  notation denotes the set of all uncertain input parameters except  $X_i$ . The effect of varying  $X_i$  alone is  $f_i$ , and the effect of varying  $X_i$  and  $X_j$  together is  $f_{i,j}$ . Higher-order terms have analogous definitions. Let  $\mathbf{u} = \{i_1, i_2, \dots, i_k\} \subset \{1, 2, \dots, n\}$  and  $\mathbf{X}_{\mathbf{u}} = (X_{i_1}, X_{i_2}, \dots, X_{i_k})$ . Then the effect of varying the set of parameters  $\mathbf{X}_{\mathbf{u}}$  is  $f_{\mathbf{u}}$ , where  $\mathcal{P}(\mathbf{u})$  is the set of all subsets of  $\mathbf{u}$ .

Since the inputs are independent, elements of the decomposition are orthogonal:

$$\mathbb{E}[f_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}) f_{\mathbf{v}}(\mathbf{X}_{\mathbf{v}})] = 0, \quad \text{for all } \mathbf{u} \neq \mathbf{v}. \quad (5.2)$$

The variance of the output can be decomposed with respect to sets of input parameters, which leads to the decomposition of variance expression:

$$\begin{aligned} \text{Var}(f(\mathbf{X})) &= \sum_{i=1}^n \text{Var}_{X_i}(f_i(X_i)) + \sum_{i=1}^n \sum_{j=i+1}^n \text{Var}_{\mathbf{X}_{i,j}}(f_{i,j}(X_i, X_j)) \\ &\quad + \dots + \text{Var}_{\mathbf{X}}(f_{1,2,\dots,n}(\mathbf{X})). \end{aligned} \quad (5.3)$$

The *main effect index* of  $X_i \in \mathbf{X}$  is the  $i$ th input's contribution to the output variance averaged over variations in other input parameters:

$$S_i = \frac{\text{Var}_{X_i} f_i(X_i)}{\text{Var} f(\mathbf{X})}. \quad (5.4)$$

By definition  $\sum_{i=1}^n S_i \leq 1$ . The sum of all of the main effects is equal to one if there are no interactions between parameters, and it is less than one if there are interactions. Alternatively, the *total effect index* of  $X_i$  measures the contribution to the output variance of  $X_i$ , including all variance caused by its interactions:

$$T_i = \frac{\mathbb{E}_{\mathbf{X}_{\sim i}}[\text{Var}_{X_i}(f(\mathbf{X})|\mathbf{X}_{\sim i})]}{\text{Var}(f(\mathbf{X}))}. \quad (5.5)$$

Since the effect of interactions between two parameters is double counted in their total effect indices,  $\sum_{i=1}^n T_i \geq 1$ . The sum of all of the total effects is equal to one if there are no interactions between parameters, and it is greater than one if there are interactions. Therefore,  $S_i \leq T_i$ , where  $S_i = T_i$  indicates parameter  $i$  does not interact with any other parameter.

**5.2. Grouped Sobol' Indices.** The notion of main and total effect indices can also be extended to groups of parameters. The benefit of using grouped indices in this application are twofold. First, parameters can be grouped phenomenologically by the bonding types, making their importance easily ranked and interpreted. Second, grouping the parameters by bonding type is a lower cardinality than the original set of parameters, which results in a decrease in computational cost to compute the indices. We follow the notation of Prieur and Tarantola [15].

The *grouped main effect index* is defined as

$$S_{\mathbf{u}}^g = \frac{\text{Var}_{\mathbf{X}_{\sim \mathbf{u}}} (\mathbb{E}_{\mathbf{X} \sim \mathbf{u}} [f(\mathbf{X}) | \mathbf{X}_{\mathbf{u}}])}{\text{Var}(f(\mathbf{X}))}, \quad \mathbf{u} \subset \{1, 2, \dots, n\}. \quad (5.6)$$

The *grouped total effect index* is defined as:

$$T_{\mathbf{u}} = \frac{\mathbb{E}_{\mathbf{X}_{\sim \mathbf{u}}} [\text{Var}_{\mathbf{X}_{\mathbf{u}}} (f(\mathbf{X}) | \mathbf{X}_{\sim \mathbf{u}})]}{\text{Var}(f(\mathbf{X}))}. \quad (5.7)$$

Instead of analytically computing the main and total effect indices, we estimated them with the pick-and-freeze method [15]. A pick-and-freeze estimate is computed numerically using  $N$  Monte Carlo samples, where  $N$  should be large enough to achieve stable estimates [15].

**5.3. Groupings and Uncertainty Characterizations.** Reactions from Table 3.1 are split into groups involving weakly-bonded O, strongly-bonded O and bonded N as:

$$\begin{aligned} R_{O(s)} &= \{1, 2, 3, 4, 9, 16, 17, 18\} \\ R_{O^*(s)} &= \{5, 6, 7, 8, 19, 20\} \\ R_{N(s)} &= \{10, 11, 12, 13, 14, 15\}. \end{aligned} \quad (5.8)$$

These reactions all contain uncertain parameters that Prata et al. defined with experimental estimates [14].

**Uncertainty Characterizations** We represent the uncertain parameters in the equations as random variables and assign probability distributions to reflect what we know about plausible behavior for the parameters. In total, we define 34 uncertain parameters from the high-fidelity model.

The adsorption selectivity, being a probability, is assumed to be

$$S_j \sim \mathcal{U}[0, 1], \quad j \in \{1, 10, 16, 19\}, \quad (5.9)$$

where  $j$  is the reaction number from the Table 3.1, and  $S_5 = 1 - S_1$  since the total sticking probability of O should sum to one. If Prata et al. [14] defined the pre-exponential factor  $\gamma \leq 1$ , we use theory to constrain it:

$$\gamma_j \sim \mathcal{U}[0, 1], \quad j \in \{4, 8, 9, 13, 14, 18\}. \quad (5.10)$$

When Prata et al., relax the constraint and let the  $\gamma$ -value exceed one, we constrain it to  $\pm 20\%$  of Prata's nominal value:

$$\begin{aligned} \gamma_j &\sim \mathcal{U}[80, 120], \quad j \in \{3, 17\} \\ \gamma_7 &\sim \mathcal{U}[1.2, 1.8] \\ \gamma_{15} &\sim \mathcal{U}[8 \times 10^8, 12 \times 10^8] \\ \gamma_{17} &\sim \mathcal{U}[80, 120] \\ \gamma_{20} &\sim \mathcal{U}[800, 1200]. \end{aligned} \quad (5.11)$$

The activation energies could theoretically by any nonnegative real value, but we constrain them to an order of magnitude above and below the Prata nominal values:

$$\begin{aligned}\log_{10} E_j &\sim \mathcal{U}[3, 5], \quad j \in \{2, 6, 8, 9, 11, 14, 15\} \\ \log_{10} E_j &\sim \mathcal{U}[2, 4], \quad j \in \{3, 7, 10, 12, 13, 16, 17, 19, 20\} \\ \log_{10} E_j &\sim \mathcal{U}[1, 3], \quad j \in \{4, 18\}.\end{aligned}\quad (5.12)$$

Following Prata et al. [14], we do not define the activation energies in reactions one and five as uncertain parameters, therefore,  $E_1 = E_5 = 0$ . This assumes that there is no activation energy required for the reaction to take place, and only the adsorption selectivity influences the probability of the reaction.

**5.4. Results.** The main effect and total effect indices for the three bond groups over the range of longitudinal distances are shown in Figure 5.1. To ensure stable estimates,  $N = 50,000$  Monte Carlo samples were used. Three replicate sets of sensitivity indices were computed, which all yielded similar results. The main effect and total effect indices for the nitrogen-related reactions are all approximately zero, indicating that reactions 10 – 15 in Table 3.1 have no effect on the CO probability. The strongly-bonded O reactions have greatest impact on the variance in the QoI, which is shown by main effect indices near 0.5 and total effect indices near 0.6. Variance in the weakly-bonded O parameters has a moderate impact on the variance in the QoI, where the main effect indices are near 0.4 and total effect indices near 0.5. The increased total indices for the two O reaction sets indicate their interaction is important to the QoI.

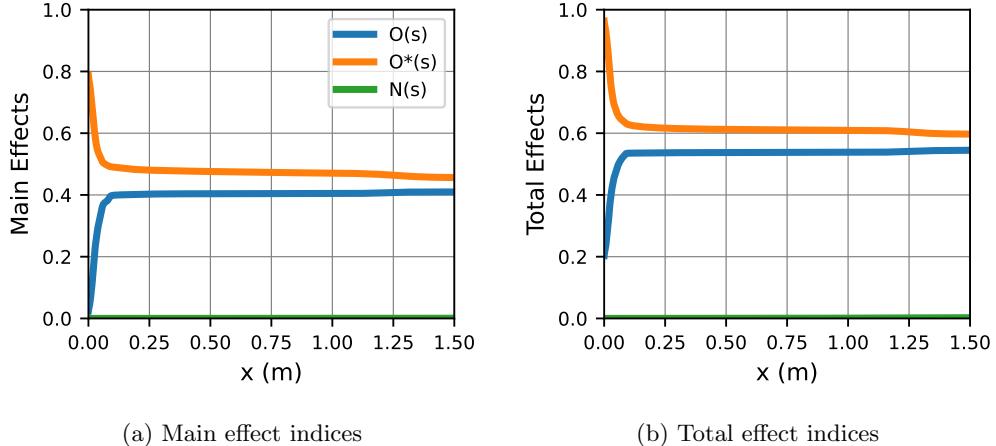


Fig. 5.1: Grouped Sobol' indices (a) main effects and (b) total effects for a range of longitudinal distances from the stagnation point of the ACA model grouped by the bond type to the surface (weakly-bonded O:  $O(s)$ , strongly-bonded O:  $O^*(s)$ , and bonded N:  $N(s)$ ).

**6. Low-Fidelity Model.** Leveraging the results from the sensitivity analysis, a low-fidelity model is formed by omitting reactions from the high-fidelity model. Nitrogen-related reactions had no effect on the QoI, so we remove them in the low-fidelity model. We also remove the reactions involving weakly-bonded O to emulate the intractability of modeling every interaction in a complete system in simulations.

The low-fidelity model is the reduced ACA model shown in Table 6.1, which involves six reaction mechanisms. The low-fidelity model describes only oxidation by O and O<sub>2</sub> and surface catalyzed recombination of O.

Table 6.1: Reduced ACA gas-surface chemistry model reaction set.

Number	Reaction	<i>S</i> or $\gamma$	<i>E</i>	Type
5	O + (s) $\rightarrow$ O*(s)	0.7	0	Adsorption
6	O*(s) $\rightarrow$ O + (s)	1.0	96500	Desorption
7	O + O*(s) + C(b) $\rightarrow$ CO + O + (s)	1000	4000	Eley-Rideal
8	O*(s) + O*(s) $\rightarrow$ O <sub>2</sub> + 2(s)	10 <sup>-3</sup>	15000	Langmuir-Hinshelwood
19	O <sub>2</sub> + 2(s) $\rightarrow$ 2O*(s)	1.0	8000	Adsorption
20	O <sub>2</sub> + O*(s) + C(b) $\rightarrow$ CO + O <sub>2</sub> + (s)	1000.0	4000	Eley-Rideal

The resulting production rates for each species are:

$$\begin{aligned}
 \frac{d[O^*(s)]}{dt} &= k_5[O][(s)] - k_6[O^*(s)] - k_7[O][O^*(s)] - 2k_8[O^*(s)]^2 \\
 &\quad + 2k_{19}[O_2][(s)]^2 - k_{20}[O_2][O^*(s)] \\
 \frac{d[CO]}{dt} &= k_7[O][O^*(s)] + k_{20}[O_2][O^*(s)] \\
 \frac{d[O]}{dt} &= \frac{P_O}{A_v\sqrt{2\pi m_O k_b T}} - k_5[O][(s)] + k_6[O^*(s)] \\
 \frac{d[O_2]}{dt} &= \frac{P_{O_2}}{A_v\sqrt{2\pi m_{O_2} k_b T}} + k_8[O^*(s)]^2 - k_{19}[O_2][(s)]^2.
 \end{aligned} \tag{6.1}$$

The total site density of the surface is

$$B = [O^*(s)] + [(s)]. \tag{6.2}$$

A steady state solution to the reduced ACA model is found by setting the rate of change in surface coverage equal to zero (i.e.,  $\frac{d[O^*(s)]}{dt} = 0$ ). Following the steady-state solution of the ACA model, let

$$\begin{aligned}
 A_2 &= 2k_{19}[O_2] \\
 B_2 &= k_5[O] \\
 C_2 &= 2k_8 \\
 D_2 &= k_6 + k_7[O] + k_{20}[O_2].
 \end{aligned}$$

Then

$$[O^*(s)] = \frac{2(A_2[(s)]^2 + B_2[(s)])}{D_2 + \sqrt{D_2^2 + 4C_2(A_2[(s)]^2 + B_2[(s)])}} \tag{6.3}$$

and

$$[(s)] = B - \frac{2(A_2[(s)]^2 + B_2[(s)])}{D_2 + \sqrt{D_2^2 + 4C_2(A_2[(s)]^2 + B_2[(s)])}}. \tag{6.4}$$

**6.1. Model-Form Error Analysis.** We compare predictions of CO probability from the high and low-fidelity models in Figure 6.1. Removing the nitrogen-related reactions did not affect the CO probability, so the high-fidelity model predictions were analogous to predictions from the ACA model omitting the nitrogen-related reactions. Since the low-fidelity model is a direct reduction from the high-fidelity model, the discrepancies can be attributed to MFE. Specifically, removing the weakly-bonding O reactions from the ACA model causes an underestimate of the CO probability for distances greater than 0.2 meters from the stagnation point. The discrepancy is inversely related to the temperature. At temperatures above 1500 kelvins, the high and low-fidelity models are in good agreement, but as the temperature decreases the discrepancy grows.

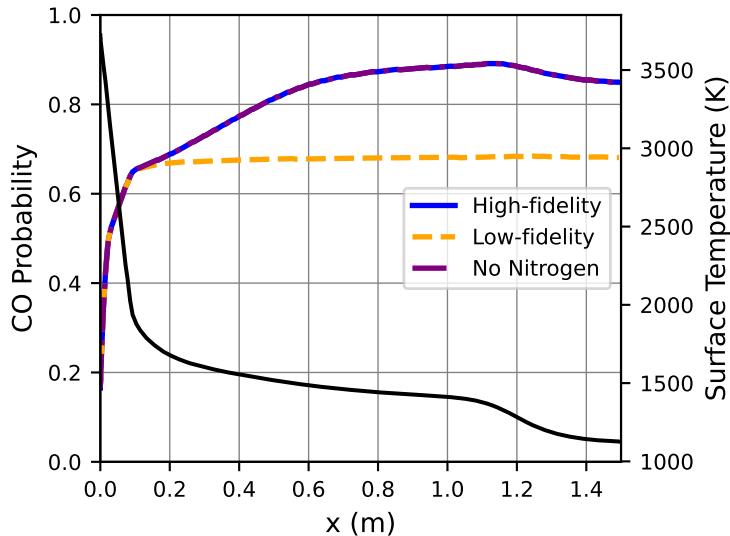


Fig. 6.1: Comparison of CO probability predictions from the high-fidelity model (blue curve), the high-fidelity model with nitrogen-related reactions removed (purple dot-dashed curve), and low-fidelity model (orange dashed curve) over a range of distances on the surface for a 40 kilometer altitude test-case. Every point on the surface has a distinct temperature (black curve).

In this application, underprediction of CO production is undesirable. If a low-fidelity model like the one defined here were used to design a TPS, it could result in a TPS that is too thin, with disastrous outcomes. We thus propose to develop a model enrichment to account for the effect of the missing weakly-bonding O reactions.

**7. Model Correction.** To reduce MFE and prevent the low-fidelity model from underpredicting, we propose to introduce a model enrichment to account for the neglected weakly-bonded O species. There are many approaches to address MFE. One common approach is to add a nonintrusive, stochastic term to the model output [7, 5]. The stochastic term is calibrated to minimize misfit with high-fidelity observations, allowing interpolation within the calibration regime. However, it cannot be applied to unobservable model outputs, e.g., predictions of the future. Another approach is to embed a correction into the low-fidelity model to form an enriched model [11, 9, 13]. The correction is theory-informed and model specific potentially allowing for extrapolations. These embedded corrections can

be more laborious to construct, but the trade-off of build time for predictive power may be worthwhile [1].

We outline a potential model correction that can be embedded into the reduced ACA model to form an enriched model. In the spirit of the catch-all reactions developed by Morrison *et al.* [9], we consider adding reactions to the low-fidelity model to capture the salient behavior of the high-fidelity model. To address the issue of the reduced ACA model underpredicting the CO probability, we consider adding a pseudo-reaction producing more CO:



Constraints for the pseudo-reaction are that it still adheres to conservation of atoms, and it does not add new species to the low-fidelity model that would increase the number of state variables. The rate coefficient for the pseudo-reaction  $k_p$  could take numerous forms, such as  $k_p = \gamma \exp(-\frac{E}{RT})$  defined by the Arrhenius equation. However the pre-exponential factor  $\gamma$  and the activation energy  $E$  could be challenging to calibrate or require non-physical values since the reaction itself is fictitious. Instead, we propose simply defining  $k_p$  as a function of temperature, like  $k_p = f(T) = mT + b$  where  $m$  and  $b$  are model parameters constrained so that  $k_p > 0$ .

Because the enriched model is still an approximation of the high-fidelity model, each model parameter will be defined by a distribution with hyperparameters, and the hyperparameters will be calibrated by observations from the high-fidelity model. Implementation and analysis of this model correction are ongoing.

**8. Conclusion.** Finite-rate gas-chemistry models provide state-of-the-art predictions for TPS ablation. However, there are numerous uncertainties and simplifying assumptions in these models. We investigate the ACA model to understand its uncertainties and the ramifications of reducing the model further. Sensitivity analysis indicates that the nitrogen-related reactions in the ACA model do not impact predictions for our test case. Furthermore, we exhibit how sensitivity analysis can inform the formation of a low-fidelity model—we omit weakly-bonded oxygen and nitrogen-related reactions to create a reduced ACA model. As the temperature is decreased, the MFE in the reduced ACA model causes increased discrepancy between predictions from the ACA and reduced ACA models.

To decrease the discrepancy, we will embed a pseudo-reaction into the reduced ACA model to form an enriched model. The enriched model will involve the same number of state variables as the reduced ACA model, and it will still conserve atoms.

Grouping the uncertain parameter by reactions involving weakly-bonded O, strongly-bonded O and bonded N was one of many possibilities. In the future, we will investigate additional options, such as grouping uncertain parameters by reaction number. This could inform which reactions in the ACA model are the most important to the production of CO and could potentially inform the model correction.

**9. 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 & 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 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. The authors would like to thank Erin Mussoni and Eric Smoll for many helpful discussions on ablation modeling.

## REFERENCES

- [1] E. ACQUESTA, T. PORTONE, R. DANDEKAR, C. RACKAUCKAS, R. BANDY, AND J. HUERTA, *Model-form epistemic uncertainty quantification for modeling with differential equations: Application to epidemiology*, Sandia Report, (2022), p. 44.
- [2] S. BERRY, R. KIMMEL, AND E. RESHOTKO, *Recommendations for hypersonic boundary layer transition flight testing*, in 41st AIAA Fluid Dynamics Conference and Exhibit, 2011, p. 3415.
- [3] G. CANDLER, *Nonequilibrium processes in hypervelocity flows: an analysis of carbon ablation models*, in 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, 2012, p. 724.
- [4] G. V. CANDLER, C. R. ALBA, AND R. B. GREENDYKE, *Characterization of carbon ablation models including effects of gas-phase chemical kinetics*, Journal of Thermophysics and Heat Transfer, 31 (2017), pp. 512–526.
- [5] D. HIGDON, J. GATTIKER, B. WILLIAMS, AND M. RIGHTLEY, *Computer model calibration using high-dimensional output*, 103 (2008), pp. 570–583.
- [6] M. HOWARD, D. W. KUNTZ, R. E. HOGAN, AND B. F. BLACKWELL, *Ablation modeling and simulation at Sandia National Laboratories.*, (2012).
- [7] M. C. KENNEDY AND A. O'HAGAN, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63 (2001), pp. 425–464.
- [8] H. L. KLINE, C.-L. CHANG, AND F. LI, *Boundary layer stability and transition in a chemically reacting martian atmosphere using lastrac*, in 22nd AIAA international space planes and hypersonics systems and technologies conference, 2018, p. 5206.
- [9] 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.
- [10] E. MUSSONI, R. M. WAGNILD, J. WINOKUR, AND J.-P. R. DELPLANQUE, *Sensitivity analysis of air/carbon finite-rate surface ablation models*, in AIAA Aviation 2022 Forum, 2022, p. 3944.
- [11] T. A. OLIVER, G. TEREJANU, C. S. SIMMONS, AND R. D. MOSER, *Validating predictions of unobserved quantities*, Computer Methods in Applied Mechanics and Engineering, 283 (2015), pp. 1310–1335.
- [12] S. POOVATHINGAL, T. E. SCHWARTZENTRUBER, V. J. MURRAY, T. K. MINTON, AND G. V. CANDLER, *Finite-rate oxidation model for carbon surfaces from molecular beam experiments*, AIAA journal, 55 (2017), pp. 1644–1658.
- [13] T. PORTONE AND R. D. MOSER, *Bayesian inference of an uncertain generalized diffusion operator*, SIAM/ASA Journal on Uncertainty Quantification, 10 (2022), pp. 151–178.
- [14] K. S. PRATA, T. E. SCHWARTZENTRUBER, AND T. K. MINTON, *Air–carbon ablation model for hypersonic flight from molecular-beam data*, AIAA journal, 60 (2022), pp. 627–640.
- [15] C. PRIEUR AND S. TARANTOLA, *Variance-Based Sensitivity Analysis: Theory and Estimation Algorithms*, in Handbook of Uncertainty Quantification, R. Ghanem, D. Higdon, and H. Owhadi, eds., Springer International Publishing, Cham, 2017, pp. 1217–1239.
- [16] A. SALTELLI, P. ANNONI, I. AZZINI, F. CAMPOLONGO, M. RATTO, AND S. TARANTOLA, *Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index*, Computer Physics Communications, 181 (2010), pp. 259–270. Number: 2.
- [17] I. SOBOL', *Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates*, The Second IMACS Seminar on Monte Carlo Methods, 55 (2001), pp. 271–280. Number: 1.
- [18] P. VIRTANEN, R. GOMMERS, T. E. OLIPHANT, M. HABERLAND, T. REDDY, D. COURNAPEAU, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, AND SciPy 1.0 CONTRIBUTORS, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.
- [19] S. V. ZHLUKTOV AND T. ABE, *Viscous shock-layer simulation of airflow past ablating blunt body with carbon surface*, Journal of thermophysics and heat transfer, 13 (1999), pp. 50–59.

## FLUID DYNAMICS ON A ROTATING SPHERE WITH DOUBLE FOURIER SERIES AND LAGRANGIAN ADVECTION

MICHAEL CHIWERE\*, PETER A. BOSLER†, AND GRADY B. WRIGHT‡

**Abstract.** A method for solving the equations of motion for a two-dimensional fluid on a rotating sphere is presented. The method combines Lagrangian advection of vorticity with the double Fourier series Poisson solver. Results are presented for the case of incompressible flow, i.e., the barotropic vorticity equation.

**1. Introduction.** Atmospheric global circulation can be modeled mathematically using the full 3D Euler equations, and inviscid approximations of the Navier-Stokes equations [27]. These equations are complicated to solve, so we usually simplify them depending on the part of the global circulation to be studied. Suppose one wants to develop new numerical methods for geophysical flow. In that case, it is imperative to work with simpler forms of these equations, which can capture essential features in the geophysical flow but are still simple enough. For global climate, the typical choice is the two-dimensional fluid layer on the rotating sphere. In this case, the primary assumption is that the atmosphere is a single layer with uniform density [27]. Shallow water equations on a rotating sphere represent 2D fluid flow and also include gravity waves. An alternative is when the 2D flow is also assumed to have constant depth, which eliminates the gravity equations, simplifying them further to barotropic vorticity equations (BVE). It is worth pointing out that the first ever successful climate model implemented BVE [3].

Bosler [3] proposes a method for solving 2D fluid on a rotating sphere using a vortex-based method. In vortex methods, velocity is computed from vorticity using Biot-Savart law [10, 18, 4]. The discretization of the Biot-Savart results in an  $N$ -body problem that scales as  $\mathcal{O}(N^2)$  where  $N$  is a number of particles, which is not optimal. Nevertheless, applying the vortex methods to 2D rotating fluid showed the scheme's accuracy and ability to resolve small-scale features. This success implies that the method has more potential if the computation cost for evaluating the  $N$ -body problem is reduced or eliminated.

Accelerated summation algorithms such as fast multipole methods [20] and tree-traversal methods [29] reduce the computational cost to  $O(N)$  (typically with a large constant) and  $O(N \log N)$  (typically with a smaller constant), respectively, and retain the Lagrangian particle-based traits of the discretization. Yet another alternative, which we investigate here, are hybrid particle/grid methods, where the advection of vorticity is handled by Lagrangian particles while a grid is used for efficient solves, typically based on fast Fourier transforms or multigrid methods.

In the current work, we are proposing solving the BVE [27, 3],

$$\frac{\partial}{\partial t}(\zeta + f) + \vec{u} \cdot \nabla(\zeta + f) = 0, \quad (1.1)$$

where  $\zeta = \nabla \times \vec{u}$  is the relative vorticity and  $f = 2\Omega z$  is the Coriolis parameter, by combining the vortex method for 2D fluid on a rotating sphere in [3] with a double Fourier sphere (DFS) method based spectral discretization described in [28, 25]. In this formulation, the advection equations are computed on the particles like [4]. In contrast, the velocity is computed on a uniform co-latitude-longitude grid using the DFS-based Poisson solver in [28, 25]. At each step, we interpolate vorticity to the grid, which is then used to compute the stream

---

\*Boise State University, michaelchiwere@u.boisestate.edu

†Sandia National Laboratories, pabosle@sandia.gov

‡Boise State University, gradywright@u.boisestate.edu

function by solving a Poisson equation relating the vorticity to the stream function. Then, the rotational derivative of the stream function on the sphere gives the fluid velocity. The velocity is then interpolated to each particle position. Grid point indices  $(i, j)$  correspond to co-latitude  $\theta_i = i\Delta\theta \in [0, \pi]$  and longitude  $\phi_j = j\Delta\phi \in [0, 2\pi]$ , where  $\Delta\phi = \Delta\theta = 2\pi/n_{lon}$  and  $n_{lon}$  is the number of points used to discretize longitude. We assume that  $n_{lon}$  is even, so that  $n_{lat} = n_{lon}/2 + 1$ , where  $n_{lat}$  is the number of grid points used for co-latitude. The total number of grid points is therefore  $n_{lon}(n_{lon}/2 + 1) = n_{lon}^2/2 + n_{lon} \approx n_{lon}^2$ , where the approximation is used in the context of grid refinement as  $n_{lon} \rightarrow \infty$  for asymptotic estimates.

The challenge with developing high-order methods on a tensor-product latitude-longitude grid is that transforming a function defined on the sphere to a tensor-product grid introduces artificial boundaries at the poles and that the function's bi-periodicity is lost. The DFS method removes the singularity at the poles and recovers the  $2\pi$  bi-periodicity. The  $2\pi$  bi-periodicity makes it possible to approximate functions on the sphere using bivariate Fourier Series, or double Fourier series, which is where the name, DFS method originates [21, 6, 30, 12, 25]. The DFS method has been successfully utilized to define new methods for solving geophysical problems, examples include [8, 16, 32]. Wilber [28] uses the DFS method and fast Fourier transform [9] to define an optimal method for solving a Poisson equation that is also exponentially accurate. The method scales as  $\mathcal{O}(n_{lon}^2 \log n_{lon}^2)$ . In addition, they also define an optimal method for computing the rotational derivative of a scalar field on the sphere using the DFS method and FFT that is also exponentially accurate and efficient. In the current work, we propose replacing the Biot-Savart integral in the method in [3] by the DFS-based Poisson solver and the DFS-based computation of the rotation of a scalar field to compute the velocity from vorticity.

A challenge here is that the fluid velocity computed using a DFS-based scheme is on the grid, but the rest of the algorithm will need the fluid velocity to be defined on the particles. Here, we will interpolate the fluid velocity on the grid to the particle by expanding each component of the fluid velocity using a bivariate Fourier series. These expansions are made possible by the DFS method, which allows us to relate each velocity component to a bi-periodic function. To make the evaluation of the DFS expansion at the particles efficient, we use the nonuniform fast forward transform [11, 1, 13, 23, 2, 24]. Another challenge is that the vorticity is given at particles. However, we need it on the grid to be able to use it to compute the velocity on the grid. Here, we interpolate vorticity to the grid from the particles using the generalized least squares algorithm implementation in the Compadre Toolkit [15].

The rest of the report is organized as follows. Section 2 describes the numerical method utilized in this study in detail, beginning with the DFS-based techniques in subsection 2.1, followed by the discussion of the Lagrangian vorticity in subsection 2.2, the interpolation method in subsection 2.3. We then present some preliminary numerical results on the method in 3 and follow with some concluding remarks in section 4.

## 2. Numerical method.

**2.1. Double Fourier series (DFS) methods.** The transformation of functions on the sphere to a tensor product co-latitude longitude grid introduces artificial boundaries at poles. Also, it results in a loss of periodicity in the co-latitude direction. The loss of periodicity implies that algorithms for periodic functions such as the FFT [9] can not be directly applied. The artificial boundary at the poles makes developing high-order methods for the sphere challenging. These challenges are the motivations for the DFS method [21, 6, 30, 12, 25]. The DFS method relates a function  $f$  defined on the sphere to a “doubled up” function  $\tilde{f}$ . This process recovers periodicity in the co-latitude direction and eliminates the

artificial boundaries at the poles. The function  $\tilde{f}$  is said to be *block-mirror centrosymmetric* type I function (BMC-I) (see [25]). The properties of the “double up” function  $\tilde{f}$  allow for developing optimal and high-order methods for the sphere. Some of such methods, which are utilized in this study include a method for solving a Poisson equation [25], a method for computing a rotation of the scalar field on the sphere [28], and exponentially accurate interpolation on the sphere which utilizes the nonuniform fast Fourier transform (NUFFT) [11, 13, 23, 2].

**2.1.1. Poisson solver based on the DFS method.** The stream function,  $\psi(x, y, z)$  and relative vorticity,  $\zeta(x, y, z)$  satisfies the equation [27, 3]

$$\Delta\psi(x, y, z) = \zeta(x, y, z), \quad (2.1)$$

where  $\Delta$  is the spherical Laplacian or Laplace–Beltrami operator. Several methods, including finite-difference and Galerkin methods, can be used to solve (2.1). However, in this work we follow a method discussed in [25] which utilizes the DFS method [21, 6, 30, 31, 12, 25] and Fourier spectral discretization [7] to compute the numerical solution for (2.1). The scheme’s appeal is that it results in tridiagonal matrices, which can be easily solved in  $\mathcal{O}(N)$  operations for  $N$  number of points on the grid.

To discretize (2.1) we first transform it to spherical coordinates by letting [25, 28],

$$x = \cos\phi \sin\theta, \quad y = \sin\phi \sin\theta, \quad z = \cos\theta,$$

where  $\theta$  is the co-latitude and  $\phi$  is the longitude. Then, we can rewrite the Poisson equation in the spherical coordinates as,

$$\sin^2\theta \frac{\partial^2\psi(\phi, \theta)}{\partial\theta^2} + \cos\theta \sin\theta \frac{\partial\psi(\phi, \theta)}{\partial\theta} + \frac{\partial^2\psi}{\partial\phi^2} = \sin^2\theta \zeta(\phi, \theta), \quad (\phi, \theta) \in [0, 2\pi] \times [0, \pi], \quad (2.2)$$

where  $\int_0^\pi \int_0^{2\pi} \psi(\phi, \theta) \sin\theta d\phi d\theta = 0$  insures a unique solution. Here, the stream function  $\psi$  is not  $2\pi$  periodic in the azimuthal direction and is not smooth at the poles. These issues are fixed by applying the DFS method to (2.2), and solve for the solution that satisfies,

$$\sin^2\theta \tilde{\psi}_{\theta\theta} + \cos\theta \sin\theta \tilde{\psi}_\theta + \tilde{\psi}_{\phi\phi} = \sin^2\theta \tilde{\psi}, \quad (\phi, \theta) \in [0, 2\pi] \times [-\pi, \pi]. \quad (2.3)$$

The solution  $\tilde{\psi}(\phi, \theta)$  must also satisfy  $\int_0^\pi \int_0^{2\pi} \tilde{\psi}(\phi, \theta) \sin\theta d\phi d\theta = 0$  as it coincides with  $\psi(\phi, \theta)$  [25]. The functions in (2.3) are  $2\pi$  bi-periodic, allowing for discretization using the Fourier spectral method [7, 25].

Given  $n_{lat}$  grid points in co-latitude  $\theta$ , and  $n_{lon}$  grid points in longitude  $\phi$ , let  $m_{lat} = 2(n_{lat} - 1)$ , [25] shows that the Fourier spectral discretization of (2.3) can be written as

$$(M_{\sin}^2 D_{m_{lat}}^2 + M_{\cos} M_{\sin} D_{m_{lat}}) X + X D_{n_{lon}}^2 = R, \quad (2.4)$$

where

$$M_{\sin} = \frac{1}{2} \begin{bmatrix} 0 & i & & \\ -i & 0 & \ddots & \\ & \ddots & \ddots & i \\ & & -i & 0 \end{bmatrix} \quad M_{\cos} = \frac{1}{2} \begin{bmatrix} 0 & 1 & & \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 0 \end{bmatrix}, \quad (2.5)$$

$$D_l = \text{diag} \left( \left[ -\frac{l}{2}i, -i, 0, \dots, \frac{l-2}{2}i \right] \right), \quad (2.6)$$

and  $R \in \mathcal{C}^{m_{lat} \times n_{lon}}$  are the Fourier coefficients of  $\sin^2 \theta \tilde{\zeta}(\phi, \theta)$ . The matrix  $D_l$  represents the structure of matrices  $D_{m_{lat}}$  and  $D_{n_{lon}}$ , which have the same structure but may have different sizes. That is  $m_{lat} \times m_{lat}$  for  $D_{m_{lat}}$ , and  $n_{lon} \times n_{lon}$  for  $D_{n_{lon}}$ . For our specific application  $m_{lat}$  and  $n_{lon}$  are the same, so  $D_{m_{lat}}$  and  $D_{n_{lon}}$  are the same matrix.

To speed up the computation of the solution in (2.4) by a factor of two, we split the system into two smaller linear systems. One is for the odd modes, while the other is for the even modes. Then, we solve the resulting systems in parallel for a fixed longitude in the Kokkos framework. Let  $I$  be the identity matrix of the size  $\frac{m_{lat}}{2} \times \frac{m_{lat}}{2}$ ,  $\mathcal{L} = M_{\sin}^2 D_{m_{lat}}^2 + M_{\cos} M_{\sin} D_{m_{lat}}$ , and subscripts  $E$ , and  $O$  respectively represent the even, and odd Fourier modes of matrices in (2.4). Then for each fixed  $k \in [-\frac{n_{lon}}{2}, \dots, \frac{n_{lon}-2}{2}]$ , we represent this by the use of superscript  $k$ , we solve linear systems of the form,

$$(\mathcal{L}_E + k^2 I) X_E^k = R_E^k, \quad (2.7)$$

$$(\mathcal{L}_O + k^2 I) X_O^k = R_O^k. \quad (2.8)$$

In our application,  $\frac{m_{lat}}{2}$  is even, which results in the linear system (2.7) as periodic tridiagonal for all  $k$  except for  $k = 0$  where the resulting linear system is banded, with a dense row  $\frac{n_{lon}}{2} + 1$ . Furthermore, the resulting linear system (2.8) is tridiagonal for all modes  $k$ . We solve this resulting tridiagonal system using the Thomas algorithm. The periodic tridiagonal and banded tridiagonal systems are solved using modified Gaussian elimination to minimize the amount of fill. These algorithms are of the computational complexity  $\mathcal{O}(m_{lat} n_{lon})$ , same as the original Thomas algorithm.

In our algorithm, we only solve the systems (2.7) and (2.8) for the nonpositive modes  $k < 1$ . The solutions for the positive modes are computed easily from the solutions of the systems for the negative modes using symmetry. The result of this computation is matrix  $\Psi$  of Fourier coefficients of the BMC-I function  $\tilde{\psi}$  that coincides with the stream function  $\psi$  by the DFS method. We do not transform this back to values because our target is to use these coefficients to compute velocity  $\vec{u}$ , which can also be done optimally in Fourier coefficient space.

### 2.1.2. The rotation of the gradient of the stream function using DFS method.

The computation of the fluid velocity  $\vec{u}(\vec{x})$  of the incompressible flow from the stream function has a spherical coordinates representation [3, 28]

$$\vec{u} = \nabla \psi \times \vec{x} = \begin{pmatrix} -\frac{\cos \phi \cos \theta}{\sin \theta} \frac{\partial \psi}{\partial \phi} - \sin \phi \frac{\partial \psi}{\partial \theta} \\ -\frac{\sin \phi \cos \theta}{\sin \theta} \frac{\partial \psi}{\partial \phi} + \cos \phi \frac{\partial \phi}{\partial \theta} \\ \frac{\partial \psi}{\partial \phi} \end{pmatrix}, \quad (2.9)$$

at a position  $\vec{x}$  on the sphere. In our computation, the stream function at each time step comes from the solution of the Poisson equation relating the stream function [3] to the relative vorticity computed using the method discussed in section 2.1.1. As pointed out earlier, the results in section 2.1.1 are the Fourier coefficients of the “doubled up” stream function, which is a BMC-I function [25]. So we use the grid of size  $m_{lat}$  by  $n_{lon}$  as in section 2.1.1. It is convenient for us because it makes evaluation of (2.9) fast when exploiting the sparsity of the Fourier representation of the trigonometric functions, sine, and cosine.

If we let  $\vec{U}$  be Fourier coefficients of  $\vec{u}$ , and matrices  $M_{\sin}$ ,  $M_{\cos}$ ,  $N_{\sin}$ ,  $N_{\cos}$ , represent matrices for multiplying any matrix of Fourier coefficients of a function by  $\sin \theta$ ,  $\cos \theta$ ,  $\cos \phi$ , and  $\sin \phi$  respectively. Matrices  $M_{\sin}$  and  $M_{\cos}$  are the same matrices given in (2.5), whereas

matrices  $N_{\sin}$  and  $N_{\cos}$  have the same structure as matrices in (2.5) but have dimension  $n_{lon} \times n_{lon}$  representing multiplication on the right-hand side by  $\sin \phi$  and  $\cos \phi$  respectively. We also take matrices  $D_{m_{lat}}$  and  $D_{n_{lat}}$  to be as defined in section 2.1.1. Multiplying Fourier coefficients of a function on the left by matrix  $D_{m_{lat}}$  represents taking the partial derivative  $\frac{\partial}{\partial \theta}$ , and multiplying matrix  $D_{n_{lon}}$  on the right represents the partial derivative  $\frac{\partial}{\partial \phi}$  in Fourier coefficient space. So we can rewrite (2.9) in Fourier spectral discretization as [25, 7, 26]

$$\vec{U} = \begin{pmatrix} -M_{\cos} M_{\sin}^{-1} \Psi D_{n_{lon}} N_{\cos} - D_{m_{lat}} \Psi N_{\sin} \\ -M_{\cos} M_{\sin}^{-1} \Psi D_{n_{lon}} N_{\sin} - D_{m_{lat}} \Psi N_{\cos} \\ \Psi D_{n_{lon}} \end{pmatrix}. \quad (2.10)$$

In implementing (2.10), the only stored matrices are  $\vec{U}$  and  $\Psi$ . Here the  $(i, j)^{\text{th}}$  entry of the product  $D_{m_{lat}} \Psi$  is given as  $(D_{m_{lat}} \Psi)_{ij} = (D_{m_{lat}})_{ii} (\Psi)_{ij}$ , where as  $(i, j)^{\text{th}}$  entry of the product  $\Psi D_{n_{lon}}$  is given as  $(\Psi D_{n_{lon}})_{ij} = (\Psi_{ij}) (D_{n_{lon}})_{jj}$ . Thus, we only store the diagonal elements of matrices  $D_{m_{lat}}$  and  $D_{n_{lon}}$ . These computations have computational complexity  $\mathcal{O}(m_{lat} n_{lon})$ . Also, when we multiply any matrix  $X$  by matrices  $M_{\sin}$  or  $N_{\sin}$ , we have the  $(i, j)^{\text{th}}$  entry of  $M_{\sin} X$  given as  $(M_{\sin} X)_{ij} = \frac{i}{2} (-X_{i-1,j} + X_{i+1,j})$ , for  $i = 2, \dots, n_{lon} - 1$ ,  $(M_{\sin} X)_{1j} = \frac{i}{2} X_{2,j}$  and  $(M_{\sin} X)_{m_{lat}j} = -\frac{i}{2} X_{m_{lat}-1,j}$ . Whereas,  $(X N_{\sin})_{i,1} = -\frac{i}{2} X_{i,2}$ ,  $(X N_{\sin})_{i,j} = \frac{i}{2} (-X_{i,j-1} + X_{i,j+1})$ , for  $i = 1, \dots, m_{lat}$ ,  $j = 2, \dots, n_{lon} - 1$ , and  $(X N_{\sin})_{i,n_{lon}} = \frac{i}{2} X_{i,n_{lon}}$ . These computations also have computational complexity  $\mathcal{O}(m_{lat} n_{lon})$ . We implemented multiplication by matrices  $M_{\cos}$  and  $N_{\cos}$  in the same way.

For the product  $M_{\sin}^{-1} X$  for any matrix  $X$ , we do not compute the inverse of matrix  $M_{\sin}$ . Rather, we solve for the equation  $M_{\sin} Y = X$  by exploiting the structure of matrix  $M_{\sin}$ . First, we solve for the rows  $i = 1, 3, \dots, m_{lat} - 1$  by forward substitution, and we use backward substitution to solve for the rows  $= m_{lat}, m_{lat} - 2, \dots, 2$ . Making the multiplication  $M_{\sin}^{-1} X$  also have computational complexity  $\mathcal{O}(m_{lat} n_{lon})$ . Therefore, we compute the Fourier coefficients of fluid velocity from the stream function using the formula in (2.10) with computational complexity  $\mathcal{O}(m_{lat} n_{lon})$ .

**2.1.3. Interpolation on the sphere using DFS method.** Given the  $m_{lat} \times n_{lon}$  matrix  $X$  of Fourier coefficients of BMC-I function  $\tilde{f}(\phi, \theta)$ . We can approximate the function  $f(\phi, \theta)$  on the sphere using a 2D nonuniform discrete Fourier transform as [25, 23, 11, 13],

$$f(\phi, \theta) = \sum_{j=-\frac{m_{lat}}{2}}^{\frac{m_{lat}}{2}-1} \sum_{k=-\frac{n_{lon}}{2}}^{\frac{n_{lon}}{2}-1} X_{jk} e^{ijk\theta} e^{ik\phi}. \quad (2.11)$$

Here, the frequencies  $j$  and  $k$  are integers, and for the case where the points  $\phi \in [0, 2\pi]$ ,  $\theta \in [-\pi, \pi]$  are equally spaced on the grid, we have an entirely uniform transform, and so we can evaluate it using the fast Fourier transform (FFT) in  $\mathcal{O}(m_{lat} n_{lon} \log m_{lat} n_{lon})$  [9]. When the frequencies are noninteger, or the samples are nonuniform, the sum (2.11) is a nonuniform transform. A class of quasi-optimal algorithms exist for evaluating these nonuniform transforms known as nonuniform fast Fourier transforms (NUFFTs) [23, 2, 13, 11] which are categorized into three types depending on whether the frequencies are noninteger, or the samples are nonuniform, or both are true. These types are NUFFT-I, NUFFT-II, and NUFFT-III, representing nonuniform fast Fourier transforms of the first, second, and third kind, respectively. In this work, we are interested in interpolating particles unevenly distributed on the sphere. That is, we have nonuniform samples and integer frequencies, and in such a case, we evaluate the sum in (2.11) using NUFFT-II [23, 2]. It has

computational complexity is  $\mathcal{O}(K(m_{lat}n_{lon} \log m_{lat}n_{lon}))$  for a constant  $K$  that depends on the accuracy parameter  $\epsilon > 0$  [25]. There are many implementations available for NUFFTs, but in this study, we use the FINUFFT [2], a library of efficient parallel algorithms for all types of NUFFTs up to three dimensions implemented in C++. It was chosen because of its high speed and efficient memory usage.

This study uses this approximation to interpolate the velocity from the grid to the particles. The Fourier coefficients of the velocity are computed using the algorithm discussed in section 2.1.2. Each component of  $\vec{U}$  in equation (2.10) are Fourier coefficients of BMC-I function, so we approximate the velocity on any point on the sphere (the particles) using the formula given in equation (2.11) via NUFFT-II implementation in the FINUFFT library.

**2.2. Lagrangian vorticity dynamics.** A set of Lagrangian particles and panels (see [3, 5] for details) are defined on the sphere. Each Lagrangian particle is labeled by a Lagrangian parameter, or material coordinate,  $\vec{a}$ , so that its physical position is  $\vec{x}(\vec{a}, t)$ . We solve the vorticity dynamics on the particles, to take advantage of the Lagrangian reference frame; each particle's vorticity is  $\zeta(\vec{x}(\vec{a}, t), t)$ . Particle vorticity is interpolated to the DFS grid and we use the DFS velocity (2.9) to advect the particles. Interpolation from scattered particle data is discussed in the next section. This combination of Lagrangian particle with the above DFS velocity solver constructs a vortex-particle-mesh method that solves the barotropic vorticity equation (1.1). For each particle we have the coupled ODEs [10, 3]

$$\frac{D\vec{x}}{Dt}(\vec{a}, t) = \vec{u}(\vec{x}(\vec{a}, t), t), \quad \vec{x}(\vec{a}, 0) = \vec{a}, \quad (2.12)$$

$$\frac{D\zeta}{Dt}(\vec{a}, t) = -2\Omega \frac{Dz}{Dt}, \quad \zeta(\vec{x}, 0) = \zeta_0(\vec{x}). \quad (2.13)$$

Absolute vorticity is initialized as  $\omega(\vec{x}, t) = \zeta_0(\vec{x}) + f(\vec{x})$ . Since absolute vorticity is materially conserved, i.e.,  $D(\zeta + f)/Dt = 0$ , the vorticity of each particle at later times is known exactly:

$$\zeta(\vec{x}, t) = \zeta_0(\vec{a}) - f(\vec{a}) - f(\vec{x}(\vec{a}, t)). \quad (2.14)$$

The second-order Runge-Kutta (RK2) time stepping scheme with time step  $\Delta t$  is

$$\zeta_1^*(\vec{x}, t_n) = \Delta t (-2\Omega w(\vec{x}, t_n)), \quad (2.15)$$

$$\vec{x}_1^*(\vec{x}, t_n) = \Delta t u(\vec{x}, t_n), \quad (2.16)$$

$$\zeta_2^*(\vec{x}, t_{n+1/2}) = \Delta t (-2\Omega w(\vec{x} + \vec{x}_1^*, t_{n+1/2})), \quad (2.17)$$

$$\vec{x}_2^*(\vec{x}, t_{n+1/2}) = \Delta t \vec{u}(\vec{x}, t_{n+1/2}), \quad (2.18)$$

$$\zeta(\vec{x}, t_{n+1}) = \zeta(\vec{x}, t_n) + \frac{1}{2}(\zeta_1^*(\vec{x}, t_n) + \zeta_2^*(\vec{x}, t_{n+1/2})), \quad (2.19)$$

$$\vec{x}(\vec{a}, t_{n+1}) = \vec{x}(\vec{a}, t_n) + \frac{1}{2}(\vec{x}_1^*(\vec{x}, t_n) + \vec{x}_2^*(\vec{x}, t_n)). \quad (2.20)$$

The above equations yield the following algorithm. For each time step, at each particle  $\vec{x}_j(t_n) \approx \vec{x}(\vec{a}_j, t_n)$ :

1. On input, at  $t = t_n$ , vorticity  $\zeta_j(t) = \zeta(\vec{x}_j(t_n), t_n)$  and velocity  $\vec{u}_j(t_n) = \vec{u}(\vec{x}_j(t_n), t_n)$  are known.
2. The first stage of the RK2 algorithm,  $\zeta_1^*$  and  $\vec{x}_1^*$  is computed at each particle using the input data.
3. The intermediate vorticity,  $\zeta_j(t_{n+1/2}) = \zeta_j(t_n) + \zeta_{1,j}^*(t_n)$  is computed at each particle.

4. The intermediate particle positions are computed,  $\vec{x}_j(t_{n+1/2}) = \vec{x}_j(t_n) + \vec{x}_{1,j}^*(t_n)$ .
5. The intermediate vorticity is interpolated to the DFS grid.
6. The Poisson solver finds the intermediate velocity  $\vec{u}_j(\vec{x}_j(t_{n+1/2}), t_{n+1/2})$  at each particle's intermediate position.
7. The second stage of RK2,  $\zeta_2^*$  and  $\vec{x}_2^*$ , is computed at each particle.
8. The final update is computed,

$$\zeta_j(t_{n+1}) = \zeta_j(t_n) + \frac{1}{2} (\zeta_{1,j}^* + \zeta_{2,j}^*) , \quad (2.21)$$

$$\vec{x}_j(t_{n+1}) = \vec{x}_j(t_n) + \frac{1}{2} (\vec{x}_{1,j}^* + \vec{x}_{2,j}^*) . \quad (2.22)$$

9. Scatter the solution data back to the full particle or panel data structures.
10. Update the nearest neighbor lists for the mesh-to-grid interpolation.
11. Compute error and write output, if applicable.

**2.3. Interpolation.** In this study, in the algorithm discussed above, we interpolate the vorticity to the DFS grid using the Generalized Moving Least Squares (GMLS) [19, 22, 17, 14] method as implemented in the Compadre toolkit [15]. The GMLS is a nonparametric regression technique for approximating functions from scattered data [22]. After the interpolated vorticity is used to compute velocity using algorithms in section 2.1.1 and 2.1.2. We interpolate the velocity to the particles using the DFS interpolation algorithm discussed in section 2.1.3.

**3. Results.** To test our method, we use a standard problem: the Rossby-Haurwitz wave. Rossby-Haurwitz waves have vorticity distributions defined by linear combinations of spherical harmonics. We choose the commonly used  $Y_4^5$  harmonic and define the initial vorticity as

$$\zeta_0(\vec{x}) = z(1 - z^2)^2 \cos(4 \tan^{-1}(y, x)) . \quad (3.1)$$

These initial conditions are plotted in Figure 3.1, which shows both the DFS grid and the Lagrangian particle/panel representation of the sphere. The DFS grid used  $n_{lon} = 40$  points and the Lagrangian particles were initialized from a recursively-refined equiangular cubed sphere with  $N = 6144$  panels. The time step was  $\Delta t = 0.01$ . Solutions at  $t = 1$  and  $t = 2$  are shown in Figures 3.2 and 3.3, respectively.

The figures show that the particles have moved under the action of the flow field, and that relative vorticity error is very small. Figures 3.4 and 3.5 show different DFS grid resolutions and time step sizes. We find similar results; relative vorticity error remains at the level of double-precision roundoff for this test.

**4. Concluding discussion.** Our goal was to develop a method that retains the advantages of Lagrangian particle methods for advection-dominated problems with reduced sensitivity to particle distortion and faster time to solution, relative to previous work [3, 5]. The double Fourier series method for solving the Poisson equation is able to leverage the fast Fourier transform algorithm in both zonal and meridional directions for excellent accuracy and efficiency. Our results suggest that we have made good progress toward this goal; however, we have not had sufficient time during this summer project to perform the thorough analysis required to claim that we have achieved it. Such work is ongoing. Here, we report the current state of our investigation based on the preliminary results reported above.

The proposed vortex-particle-grid method has several sources of error. The RK2 scheme introduces time discretization error, which we expect to converge toward zero as  $O(\Delta t^2)$  as

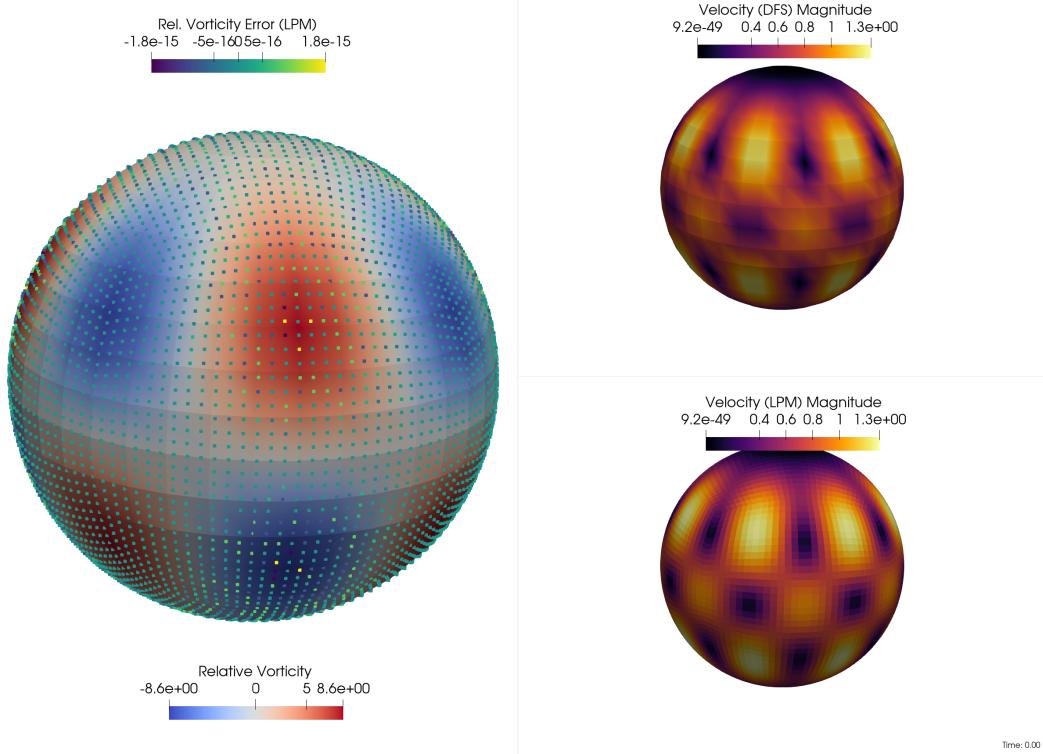


FIG. 3.1. *Initial conditions for the Rossby-Haurwitz wave. Left:* Relative vorticity is plotted on the sphere using the DFS grid; relative vorticity error is plotted at the particles. *Upper right:* Velocity magnitude on the DFS grid. *Lower right:* Velocity magnitude on the panels.

$\Delta t \rightarrow 0$ . Mapping relative vorticity from particles to the grid introduces interpolation error from GMLS. We have used a fourth-order basis in the results reported here, so we expect this error to decay at a rate of  $O(\Delta\lambda_p^5)$ , where  $\Delta\lambda_p$  is the approximate mesh size of the Lagrangian particles. For the results presented in Figures 3.1 through 3.3,  $N = 6144$  corresponds to  $\Delta\lambda_p \approx 2.6^\circ$ . The DFS grid used  $n_{lon} = 40$ , for  $\Delta\lambda_g = 9^\circ$ .

The spherical harmonics are analytic, which means the Rossby-Haurwitz is smooth. Fourier spectral representation is exponentially convergent for smooth functions [7, 26]. Thus, we expect the computation on the grid using the DFS methods to converge with the order  $\mathcal{O}(e^{-cm_{lat}n_{lon}})$ , for some constant  $c$ . The results in Figure 3.4 show that the solution converged for a grid of size  $8 \times 5$ . Also, increasing the grid size did not introduce instabilities in the solution, as the error remained at machine precision. The high accuracy in the solution made the error in time discretizations negligible, as shown in Figure 3.5. The solution also remained stable when we tried to reduce the time step after the solution had already converged.

We do not see these expected asymptotic converge rates in our Rossby-Haurwitz wave test. Instead, we observe that relative vorticity error on the particles remains near machine precision throughout the computation, even when the particles are quite distorted (e.g., Fig. 3.3) relative to their initial positions. We suspect that the Rossby-Haurwitz wave is not challenging enough to expose the true character of the numerical method.

Rossby-Haurwitz waves are defined by spherical harmonics, which are eigenfunctions of the Laplace-Beltrami operator. Hence, the stream function and the vorticity are simply

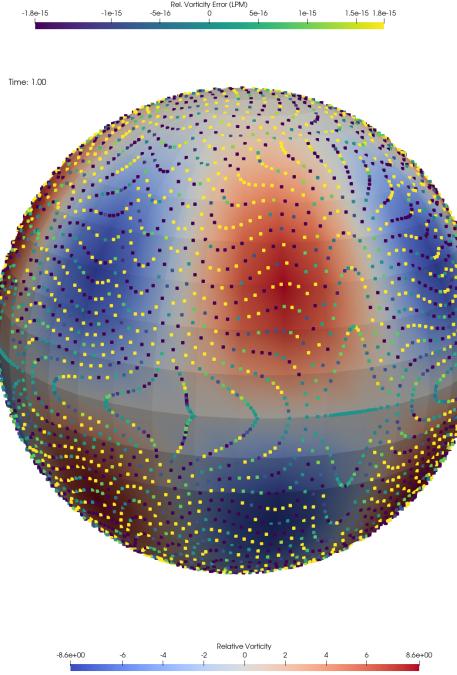


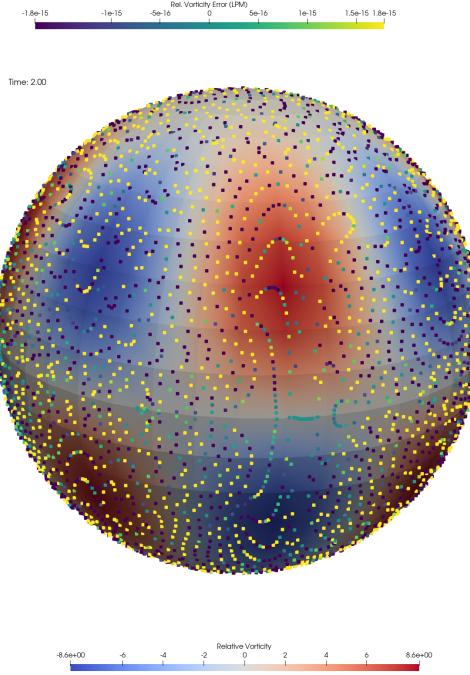
FIG. 3.2. *Solution of the Rossby-Haurwitz wave at  $t = 1$ .*

multiples of each other in both the physical domain and the spectral domain. We suspect that the numerical method is capable of exactly reproducing such relationships, though we have not had time to prove this conjecture. It is nonetheless encouraging that the GMLS interpolation error of the vorticity from the particles to the grid is low enough to retain these properties of the DFS spectral method.

Our future work plans additional tests to investigate convergence as both  $\Delta\lambda_g$  and  $\Delta\lambda_p$  decrease, and how they depend on each other in more challenging problems. We expect that, like previous work, time discretization error will be negligible, although we will also verify this expectation. Finally, we plan to expand the method to compressible flow regimes, to solve the shallow water equations on the rotating sphere.

## REFERENCES

- [1] C. ANDERSON AND M. D. DALEH, *Rapid computation of the discrete Fourier transform*, SIAM Journal on Scientific Computing, 17 (1996), pp. 913–919.
- [2] A. H. BARNETT, J. MAGLAND, AND L. AF KLINTEBERG, *A parallel nonuniform fast Fourier transform library based on an “exponential of semicircle” kernel*, SIAM Journal on Scientific Computing, 41 (2019), pp. C479–C504.
- [3] P. BOSLER, L. WANG, C. JABLONOWSKI, AND R. KRASNY, *A Lagrangian particle/panel method for the barotropic vorticity equations on a rotating sphere*, Fluid Dynamics Research, 46 (2014), p. 031406.
- [4] P. A. BOSLER, *Particle methods for geophysical flow on the sphere*, PhD thesis, University of Michigan, 2013.
- [5] P. A. BOSLER, J. KENT, R. KRASNY, AND C. JABLONOWSKI, *A Lagrangian particle method with remeshing for tracer transport on the sphere*, Journal of Computational Physics, 340, pp. 639–654.

FIG. 3.3. Solution of the Rossby-Haurwitz wave at  $t = 1$ .

- [6] J. P. BOYD, *The choice of spectral functions on a sphere for boundary and eigenvalue problems: A comparison of Chebyshev, Fourier and associated Legendre expansions*, Monthly Weather Review, 106 (1978), pp. 1184–1191.
- [7] ———, *Chebyshev and Fourier spectral methods*, Courier Corporation, 2001.
- [8] H.-B. CHEONG, *Application of double Fourier series to the shallow-water equations on a sphere*, Journal of Computational Physics, 165 (2000), pp. 261–287.
- [9] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of computation, 19 (1965), pp. 297–301.
- [10] G.-H. COTTET, P. D. KOUOMTSAKOS, ET AL., *Vortex methods: theory and practice*, vol. 8, Cambridge university press Cambridge, 2000.
- [11] A. DUTT AND V. ROKHLIN, *Fast Fourier transforms for nonequispaced data*, SIAM Journal on Scientific computing, 14 (1993), pp. 1368–1393.
- [12] B. FORNBERG, *A pseudospectral approach for polar and spherical geometries*, SIAM Journal on Scientific Computing, 16 (1995), pp. 1071–1081.
- [13] L. GREENGARD AND J.-Y. LEE, *Accelerating the nonuniform fast Fourier transform*, SIAM review, 46 (2004), pp. 443–454.
- [14] B. J. GROSS, N. TRASK, P. KUBERRY, AND P. J. ATZBERGER, *Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A generalized moving least-squares (GMLS) approach*, Journal of Computational Physics, 409 (2020), p. 109340.
- [15] P. KUBERRY, P. BOSLER, AND N. TRASK, *Compadre Toolkit*, Jan. 2019.
- [16] A. T. LAYTON AND W. F. SPOTZ, *A semi-lagrangian double Fourier method for the shallow water equations on the sphere*, Journal of Computational Physics, 189 (2003), pp. 180–196.
- [17] D. LEVIN, *The approximation power of moving least-squares*, Mathematics of computation, 67 (1998), pp. 1517–1531.
- [18] A. J. MAJDA, A. L. BERTOZZI, AND A. OGAWA, *Vorticity and incompressible flow. cambridge texts in applied mathematics*, Appl. Mech. Rev., 55 (2002), pp. B77–B78.
- [19] V. MARTIN, J. JAFFRÉ, AND J. E. ROBERTS, *Modeling fractures and barriers as interfaces for flow in porous media*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1667–1691.
- [20] P.-G. MARTINSSON, *Fast direct solvers for elliptic PDEs*, SIAM, 2019.

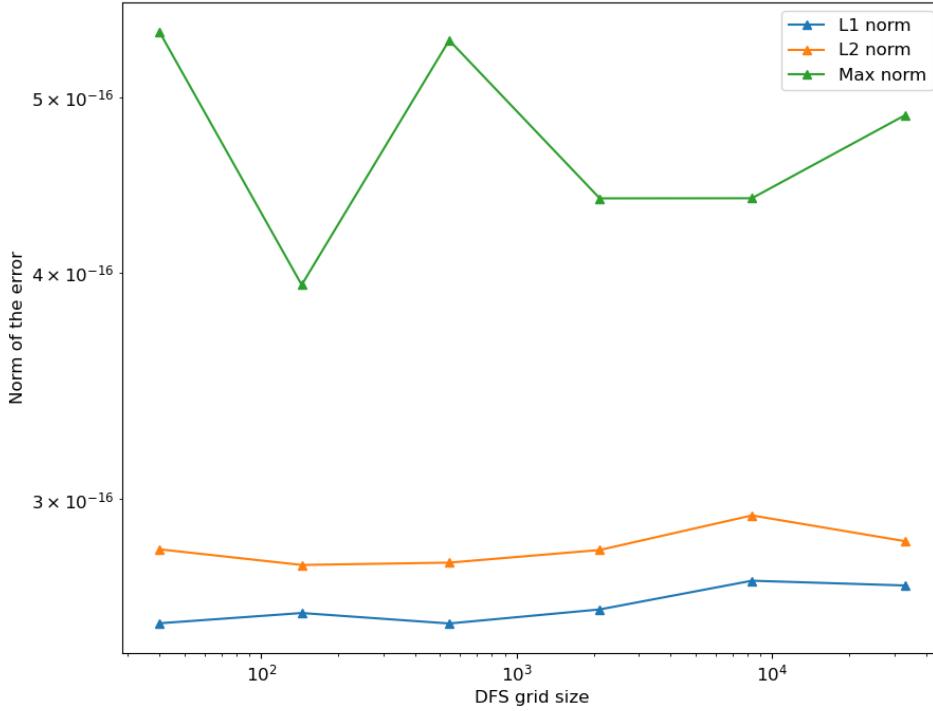


FIG. 3.4. Error in the solution of the Rossby-Haurwitz wave at  $t = 2$  for different DFS grid sizes.

- [21] P. E. MERILEES, *The pseudospectral approximation applied to the shallow water equations on a sphere*, *Atmosphere*, 11 (1973), pp. 13–20.
- [22] D. MIRZAEI, R. SCHABACK, AND M. DEHGHAN, *On generalized moving least squares and diffuse derivatives*, *IMA Journal of Numerical Analysis*, 32 (2012), pp. 983–1000.
- [23] D. RUIZ-ANTOLIN AND A. TOWNSEND, *A nonuniform fast Fourier transform based on low rank approximation*, *SIAM Journal on Scientific Computing*, 40 (2018), pp. A529–A547.
- [24] Y.-H. SHIH, G. WRIGHT, J. ANDÉN, J. BLASCHKE, AND A. H. BARNETT, *cuFINUFFT: a load-balanced GPU library for general-purpose nonuniform FFTs*, in 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2021, pp. 688–697.
- [25] A. TOWNSEND, H. WILBER, AND G. B. WRIGHT, *Computing with functions in spherical and polar geometries I. the sphere*, *SIAM Journal on Scientific Computing*, 38 (2016), pp. C403–C425.
- [26] L. N. TREFETHEN, *Approximation Theory and Approximation Practice, Extended Edition*, SIAM, 2019.
- [27] G. K. VALLIS, *Atmospheric and oceanic fluid dynamics*, Cambridge University Press, 2017.
- [28] H. D. WILBER, *Numer computing with functions on the sphere and disk*, (2016).
- [29] L. WILSON, N. VAUGHN, AND R. KRASNY, *A GPU-accelerated fast multipole method based on barycentric Lagrange interpolation and dual tree traversal*, *Computer Physics Communications*, 265 (2021), p. 108017.
- [30] S. Y. YEE, *Studies on Fourier series on spheres*, *Monthly Weather Review*, 108 (1980), pp. 676–678.
- [31] ———, *Solution of Poisson's equation on a sphere by truncated double fourier series*, *Monthly Weather Review*, 109 (1981), pp. 501–505.
- [32] H. YOSHIMURA, *Improved double Fourier series on a sphere and its application to a semi-implicit semi-lagrangian shallow-water model*, *Geoscientific Model Development*, 15 (2022), pp. 2561–2597.

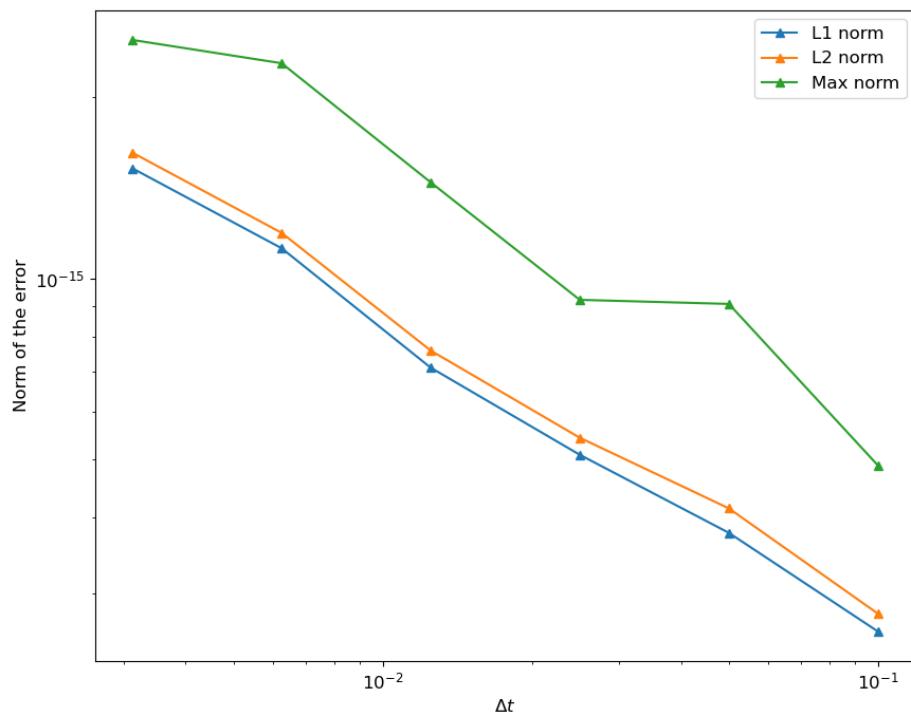


FIG. 3.5. Error in the solution of the Rossby-Haurwitz wave at  $t = 2$  for different time steps ( $\Delta t$ ) grid sizes.

## ENHANCING LINEAR SOLVERS THROUGH ADVANCED MATRIX CLUSTERING TECHNIQUES

NICHOLE ETIENNE\*, GRAHAM HARPER †, AND CHRIS SIEFERT‡

**Abstract.** This research aims to enhance the efficiency of patch-based linear solvers by uncovering dataset structures through matrix clustering methods. The study specifically investigates the impact of advanced clustering techniques, focusing on affinity propagation clustering, in comparison to conventional methods like K-means. By evaluating these advanced techniques, the study aims to enhance the efficiency and accuracy of linear solvers, contributing to advancements in data analysis, pattern recognition, and computational methodologies. The findings highlight the potential benefits of incorporating advanced clustering insights to optimize linear solver performance and drive progress in data-driven solutions

**1. Introduction.** In the field of computational mathematics, linear solvers are a vital aspect of simulations both scientific and engineering-based [3, 13]. This is mainly due to their ability to accurately approximate complex mathematical models resulting in the analysis of intricate phenomena across various domains. Among the advanced techniques that have gained popularity, patch-based linear solvers offer a versatile and efficient approach to addressing challenges associated with irregular geometries, discontinuities, and multiscale interactions [5, 9, 2]. The high-level goal of this paper is to provide an in-depth exploration of machine learning methods, particularly in detecting inherent structures within data sets to further enhance the computational efficiency of these patch-based linear solvers.

**1.1. Background.** Primarily suited for scenarios where traditional solvers are limited, patch-based solvers present a revolutionary framework in multiple areas but specifically in the realm of computational mathematics. At the core, these solvers have the ability to partition complex domains into localized patches, each conducive to more straightforward handling. This is an effective approach when dealing with multi-scale interactions, geometries, and discontinuities. By localizing computations, these solvers not only enhance accuracy but also mitigate the computational overhead associated with complex simulations [3, 5].

The capabilities of patch-based linear solvers are not only limited to their use in computational mathematics. Interestingly the solvers are quite versatile. Its versatility is seen in its application within multiple computational domains, For example, in the field of computational mechanics, patching methods were used to manage the upwind finite element democratization on non-matching grids [7]. Within subcategories of multi-scale finite element methodologies, patch-based solvers were also used to handle elliptic problems within composite materials and porous media [9]. The illustration of these examples depicts the role of solvers in achieving more accurate and realistic simulations.

In addition to the increased use of patch-based solvers, there has also been an increase in detecting structures within data sets. Data sets have the ability to expand in size and complexity. As such occurs, there is a dire need for pattern recognition, given a specific application. This is where multigrid methods come into play. Multigrid techniques are an indispensable part of the computational mathematics , known for their mastery in tackling a wide array of linear and nonlinear partial differential equations. They operate on the principle of hierarchical grids, strategically employing coarser levels to expedite convergence on finer ones. By facilitating the exchange of information across different scales of the problem, multigrid methods have the capability to significantly alleviate the computational

---

\*Emory University, nichole.etienne@emory.edu,

†Sandia National Labs, gharpe@sandia.gov,

‡Sandia National Labs, csiefer@sandia.gov

burden. Within the integration of algebraic multigrid techniques with patch-based solvers, there is a process that allows for the exploitation of structural attributes such as sparsity and hierarchy. This ultimately leads to expedited convergence rates. By leveraging these structures, computational algorithms can achieve a significant reduction in computational costs, without compromising solution accuracy [1, 5, 9].

**1.2. Research Objectives.** This paper sets out with a twofold purpose that stems from the core attributes of patch-based linear solvers. First and foremost, the overarching goal is to enhance the speed and efficiency of patch-based linear solvers, which serve as crucial components in addressing complex computational issues. The concept involves harnessing the inherent structure present within data sets to streamline problem-solving processes. This aim aligns with the knowledge that extracting and utilizing the inherent structure in data can possibly enhance speed and efficiency within certain computational tasks.

The overall goal of this paper is to increase the speed and efficiency of patch-based linear solvers by detecting structure within the data set. To achieve this, there was an aim to explore and develop novel matrix clustering methods/techniques and assess their impact on the performance of linear solvers. Specifically, an investigation of whether more advanced clustering algorithms, such as affinity propagation clustering, yield improved results compared to simpler methods like K-means clustering. Through the evaluation of these advanced techniques, the intent was to enhance the efficiency and accuracy of linear solvers, contributing to advancements in data analysis, pattern recognition, and computational methods.

## 2. Literature Review.

**2.1. Matrix Clustering.** Matrix clustering techniques, an uncommonly used term, refers to a technique by which clustering methods are applied to data consisting of matrices or groups of matrices. This technique has emerged as a powerful approach to improving the efficiency and accuracy of linear solvers by detecting underlying patterns and structures within data sets [8]. Given that this is the application of clustering, more common / simpler applications have been explored. One of which is K-means.

**2.1.1. K-means.** K-Means refers to one of the most common clustering algorithms that partition a data set into a predetermined number of clusters by iteratively optimizing the distances between data points and cluster centroids (center). Each iteration involves two steps: assigning each data point to the nearest center and recalculating the centroids based on the assigned data points. The algorithm was introduced by J. MacQueen in his paper "Some Methods for Classification and Analysis of Multivariate Observations" [11], Ideally, K-Means seeks to minimize the sum of squared distances within clusters, making it an essential technique for data segmentation and exploration in numerous fields, producing an output similar to that in figure 2.1.

Despite being applicable to forms of data clustering and somewhat powerful, advanced techniques have displayed more potential in their ability to unveil complex relationships initially overlooked by traditional methods [10]. This infers that the application of advanced algorithms may have the potential to impact performance within specific domains such as that of patch-based linear solvers.

**2.1.2. Comparison: K-means vs Affinity Propagation.** Comparative studies have delved into the efficacy of advanced clustering algorithms, particularly affinity propagation, in comparison to conventional methods like K-means. A study conducted by [10] provides a detailed analysis aimed at comprehensively assessing how affinity propagation, a more advanced clustering algorithm, compared to the widely used K-means algorithm. The literature demonstrated that affinity propagation excels in identifying intricate clusters that K-means might struggle to capture.

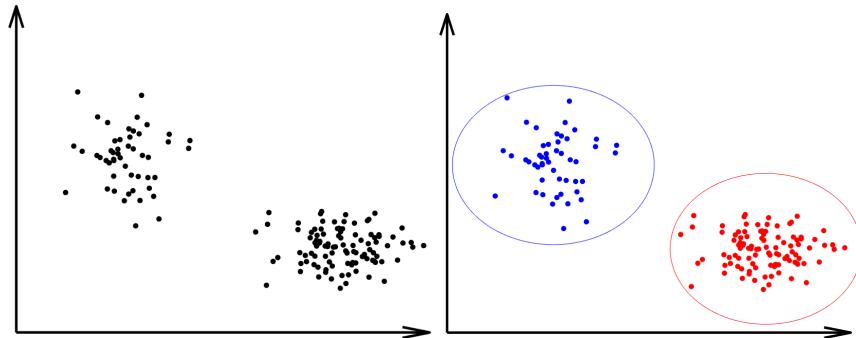


FIG. 2.1. *K*-means simplified visualization before (left) and after (right) clustering showing raw data and distinct clusters, respectively.

Affinity propagation refers to a less commonly used clustering algorithm introduced by Brendan J. Frey and Delbert Dueck in 2007 [6]. Unlike K-means, the algorithm doesn't require a predefined number of clusters. Instead, it automatically determines the number of clusters and assigns data points to those clusters based on their similarity [6]. The algorithm utilizes the concept of message-passing among data points. This works by iteratively sending "responsibility" and "availability" messages between data points to determine which points should be the exemplars (representatives) for clusters [6]. The responsibility message reflects the suitability of a data point to be the exemplar for another point, while the availability message measures the appropriateness of one data point choosing another point as its exemplar. Affinity propagation is based on set of equations. More specifics related to these equations (responsibility and availability) can be found in Equation (2.1) and Equation (2.2).

**Responsibility Equation:** The responsibility ( $r$ ) of data point  $i$  to serve as the exemplar for data point  $j$  is given by:

$$r(i, j) = s(i, j) - \max_{k \neq j} \{a(i, k) + s(i, k)\} \quad (2.1)$$

Where:

- $r(i, j)$  is the responsibility of point  $i$  to point  $j$ .
- $s(i, j)$  represents the similarity (or negative dissimilarity) between data points  $i$  and  $j$ .
- $a(i, k)$  represents the current availability of point  $k$  (explained in the availability equation).

**Availability Equation:** The availability ( $a$ ) of data point  $i$  to choose data point  $j$  as its exemplar is given by:

$$a(i, j) = \min\{0, r(j, j) + \sum_{k \neq i, j} \max\{0, r(k, j)\}\} \quad (2.2)$$

Where :

- $a(i, j)$  is the availability of point  $i$  to point  $j$ .
- $r(j, j)$  represents the responsibility of point  $j$  to be its own exemplar.
- The summation term represents the sum of positive responsibilities of other data points  $k$  choosing data point  $j$  as their exemplar.

It is believed that the uniqueness of Affinity Propagation, specifically its responsibility and availability formulation has led to impressive performance when capturing complex and non-spherical clusters [6]. It is this performance that may have significant implications for applications such as patch-based linear solvers. In the context of these solvers, the identification of meaningful data structures is essential for enhancing solver accuracy and efficiency. Affinity propagation's capability to handle intricate clusters suggests that integrating this algorithm into patch-based linear solvers could lead to improved performance by better capturing the underlying patterns within the data.

**2.1.3. Matrix Clustering and Linear Solvers.** Recently, the interplay between matrix clustering techniques and linear solvers has gained attention among researchers. This is due to its possible ability to significantly enhance solver performance by leveraging the inherent structure within data. In the past, Research has recognized that the incorporation of clustering methodologies can lead to more efficient and accurate solutions in patch-based linear solvers [12]. This stems from the fact that clustering similar matrix patches enables the solver to exploit patterns and relationships within the data, subsequently resulting in reduced computational overhead and improved accuracy.

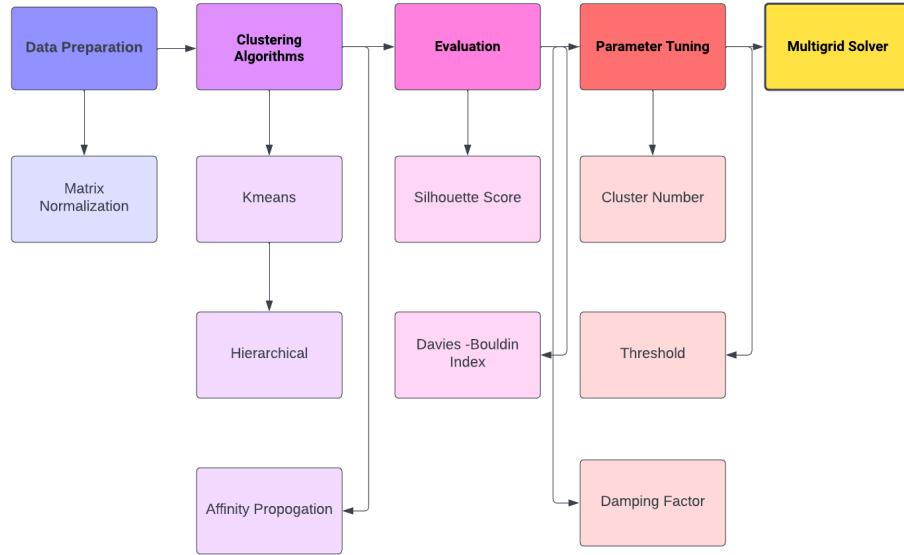
In particular, [12] infers that there are benefits in combining hierarchical clustering with patch-based linear solvers. Grouping matrix patches based on their similarity demonstrated a substantial reduction in computational time while maintaining high solver accuracy. This highlights the notion that understanding the data structure through clustering not only streamlines the solver's computational burden but also enhances its performance by addressing intricacies present in the data.

All in all, the integration of matrix clustering techniques and linear solvers has the potential to identify data structures that may enhance the performance of a solver. It is possible that clustering can make computation simpler, enhance accuracy and when dealing with complex data patterns. This ultimately translates to the important role of clustering in making computations simpler, enhancing accuracy, and dealing with complex data patterns highlighting how crucial it is for improving patch-based linear solvers and data-driven methods.

It is for this reason that this paper explores the application of simple and advanced clustering algorithms against patch-based linear solvers, aiming to understand their impact on solver performance and uncover new avenues for enhancing data analysis and pattern recognition.

**3. Methodology.** In this section, the methodology employed to cluster patch matrices using three distinct algorithms: k-means, hierarchical clustering, and Affinity Propagation are presented. As the point of this paper is to explore the application of simple and advanced clustering algorithms against patch-based linear solvers, aiming to understand their impact on solver performance and uncover new avenues for enhancing data analysis and pattern recognition. The clustered patches are then integrated into a multi-grid solver for further evaluation. Figure 3.1 provides a visual representation of the methodological process.

**3.1. Data Prepossessing.** The data used for exploration consists of a set of patch matrices, where each matrix represents a localized data point within a larger problem domain. Each element within the matrix corresponds to a feature associated with that patch. In order to ensure uniformity, normalization is applied prior to the actual application of clustering algorithms. As it relates to the matrix used, the matrix was obtained from a finite element discretization of Poisson's equation:

FIG. 3.1. *Methodology Overview*

$$\rho(x, y) = \sin^2(\pi x) \sin^2(\pi y) + 0.1, \text{ with } u = \sin(\pi x) \sin(\pi y), \text{ and}$$

$$f = \sin(\pi x) \sin(\pi y) (2\pi^2 \sin^2(\pi x) \sin^2(\pi y) - 2\pi^2 \sin^2(\pi x) \cos^2(\pi y) - 2\pi^2 \cos^2(\pi x) \sin^2(\pi y) + 0.2\pi^2)$$

We utilize order  $p = 2$  Lagrangian polynomials on a  $60 \times 60$  quadrilateral mesh, meaning the global linear system has 14641 unknowns. We later perform affinity propagation clustering on the 3600 patch matrices of size  $9 \times 9$  obtained by extracting the 9 rows and columns associated with each of the 3600 mesh cells.

**3.2. Clustering Algorithms.** For exploration and observation purposes, a series of algorithms were applied to the specified domain. These algorithms included K-means, Hierarchical Clustering, and Affinity Propagation.

The **k-means** function within the algorithm (1), takes as an input a data matrix  $A$ , the number of clusters  $k$ , a maximum number of iterations  $maxits$ , and a random seed **seed**. The function begins by initializing cluster centroids randomly and calculating the pairwise distances between data points in matrix A. It then enters a loop where it repeatedly updates the cluster assignments and repositions the cluster centroids (center) to minimize the variance within each cluster. The loop continues for a maximum of  $maxits$  iterations or until no further changes in cluster assignments occur. It is important to note that the value of  $k$  was selected in a brute-force manner. However, further exploration depended on covariance matrix values, quantifying the degree and direction of the linear relationship between two variables. Such values also determine whether or not it is effective to merge the cluster outputs.

Based on the expected effectiveness of merged clusters, hierarchical clustering was executed. Hierarchical clustering displayed a tree-like structure of clusters by iteratively merging or splitting clusters based on a selected linkage criterion. The function takes the clusters

**Algorithm 1** K-Means Clustering

---

```

1: procedure KMEANS(data, indices, k, normarg, maxits, seed)
2:   Initialize clusters with random data points
3:   repeat
4:     Assign each data point to the nearest cluster
5:     Recompute cluster centers as the mean of assigned data points
6:     if no data points change clusters then
7:       break
8:     end if
9:   until maximum iterations reached (maxits)
10:  return the final cluster centers and assignments
11: end procedure

```

---

obtained from the k-means algorithm and a specified merging threshold (a maximum allowable distance) as input. It aims to merge clusters that are close to each other based on the distances between their centroids. It does this by first calculating the centroids of the k-means clusters. It then iterates through the clusters, comparing their centroids and merging those that are within the defined threshold distance. To avoid double mergers, it keeps track of merged clusters using a list of merged flags. The result is a list of merged clusters, visualized as a dendrogram, where each merged cluster may contain multiple original k-means clusters that are close enough to be considered a single merged cluster.

Following the observation of k-means and merged clusters, a more advanced clustering algorithm was explored. **Affinity Propagation** identifies exemplars (representative patches) within a collection of patch matrices [4]. The algorithm takes as input a data matrix  $A$  containing patches, a set of patch indices patch indices, and several parameters including damping to prevent oscillations during iterations, maxits for the maximum number of iterations, and an optional seed for the random number generator to ensure reproducibility.

A similarity matrix  $S$  is computed based on the pairwise similarities between patches. It then iteratively updates responsibility and availability matrices  $r$  and  $a$  and identifies exemplars. The damping factor is used to stabilize the updates. Then, it assigns each sample to an exemplar and returns the indices of selected exemplars (exemplars) and the assigned exemplars for each sample (indices). The resulting exemplars and their associated clusters formed the final clusters.

**3.3. Multi-Grid Solver Integration.** From the clustering algorithm application, the clustered patches were fed into a multi-grid solver and observed. The multi-grid solver utilized the structure and relationships between clustered patches to efficiently solve the problem at different scales.

**3.4. Evaluation.** The term evaluation here refers solely to the quality of the clusters given a specific clustering algorithm. The quality of clustering was assessed using metrics such as silhouette score and the Davies-Bouldin index. These metrics play a crucial role in assessing how well a clustering algorithm has performed in terms of grouping similar data points together.

**3.5. Parameter Tuning.** In clustering algorithms where the number of clusters ( $k$ ) has to be defined, there is a constant hit or miss with what is the appropriate value. In other algorithms where  $k$  does not have to be defined, there are other modifiable values that can severely affect the output of an algorithm. For this reason,  $K$  was optimized in K-means, a

**Algorithm 2** Affinity Propagation Clustering

---

```

1: Initialize similarity matrix  $S$  based on data  $A$ .
2: Initialize responsibility matrix  $r$  with zeros.
3: Initialize availability matrix  $a$  with zeros.
4: Set the damping factor to a value between 0 and 1.
5: for  $iter = 1$  to maxits do
6:   for each data point  $i$  and potential exemplar  $k$  do
7:     Calculate the net responsibility for  $i$  to choose  $k$ .
8:     Update  $r(i, k)$  using the net responsibility.
9:   end for
10:  for each data point  $i$  and potential exemplar  $k$  do
11:    Calculate the net availability for  $i$  from  $k$ .
12:    Update  $a(i, k)$  using the net availability.
13:  end for
14:  Update exemplars by finding data points that maximize responsibility plus availability.
15:  Assign data points to their corresponding exemplars.
16:  Check for convergence (e.g., if exemplars and assignments do not change significantly).
17:  if converged then
18:    Exit the loop.
19:  end if
20: end for
21: Return the indices of selected exemplars and the assigned exemplars for each data point.
22: Optionally, post-process results (e.g., merge small clusters or refine exemplar selection).

```

---

threshold in hierarchical clustering, and the damping factor in Affinity Propagation.

#### 4. Results.

**4.1. Performance Comparison - Traditional Method.** This section presents the experimental results related to application of the traditional k-means algorithm. The results obtained reveal a notable tendency of K-means to generate clusters overlap. This is visible in both the covariance matrix analysis (Figure 4.1) and the Silhouette score (Table 4.1) computation.

Evaluating the covariance matrix of the clustered data, it becomes evident that certain data points within different clusters share a considerable degree of covariance. This suggests that the clusters formed by K-means often overlap upon each other's territories, blurring the boundaries between them. This poses a challenge when dealing with complex data, like that of patch matrices.

The Silhouette score, a well-regarded metric for assessing the quality of clusters, further verifies the observation found within the covariance matrix. A Silhouette score of 0.357, obtained through the K-means application, points to a sub-optimal cluster separation. This score indicates that while some data points are correctly assigned to their respective clusters, a significant portion finds themselves in close proximity to points from neighboring clusters, contributing to the observed overlaps.

In light of the overlapping clusters, the idea of merging clusters through hierarchical clustering gained traction as a potential remedy. By considering the linkage between clusters based on covariance similarity, hierarchical clustering presented an avenue to rectify the issue of overlapping clusters initially generated by the K-means. Figure 4.3 illustrates an example

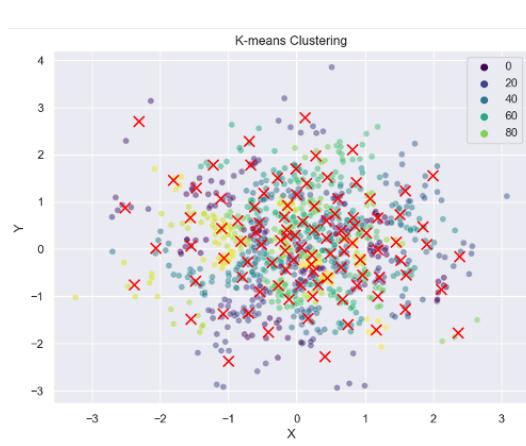


FIG. 4.1. *K*-means clustering, with  $K=100$ : Red X's mark centroids, while vibrant colors represent distinct clusters. Unveiling the hidden patterns in data

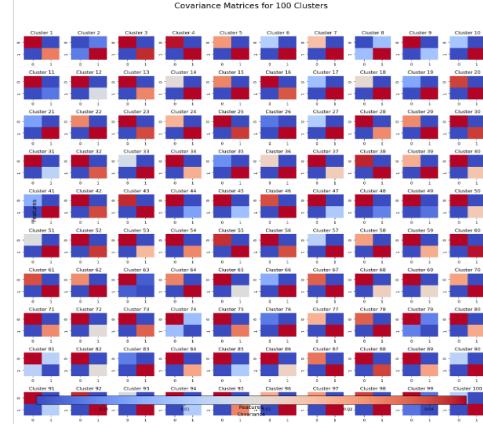


FIG. 4.2. Heat map of *K*-means Covariance, where  $K=100$ . Darker colors represent higher covariance values, inferring stronger relationships among clusters.

of cluster merging on 100 clusters from the initial k-means algorithm, with a threshold of 0.6, while Figure 4.4 has a threshold of 0.1.

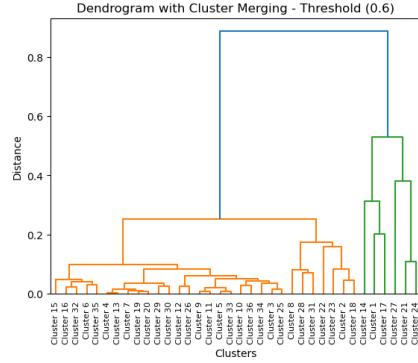


FIG. 4.3. Hierarchical Clusters producing 4 clusters from 100 original created by *k*-means. ]

To explore this avenue further, a threshold evaluation was employed as part of a brute force approach. By varying the covariance threshold, clusters that were potentially suitable for merging were identified. Hierarchical clustering was then applied to these merged clusters, resulting in more distinct and separated clusters. Interestingly, the overlap observed initially was significantly reduced, with the resulting Silhouette score improving to 0.29.

However, even with this improvement, some residual overlap remained. This is visible in the Davies-Bouldin Index of 0.615. This highlights the complexity of the overlapping patterns within the data. The scenario mentioned presents the possible challenges of utilizing Kmeans as the sole starting point for clustering when dealing with intricate data structures. While Kmeans serves as a valuable initial step, its limitations become apparent when confronted with data possessing intricate relationships and overlaps such as the patch-matrices. Ideally , more advanced algorithms may be more well suited.

**4.2. Performance Comparison - Advanced Techniques.** This section presents the outcomes achieved through the application of a more advanced clustering technique: Affinity Propagation. This aims to identify and harness underlying data relationships in

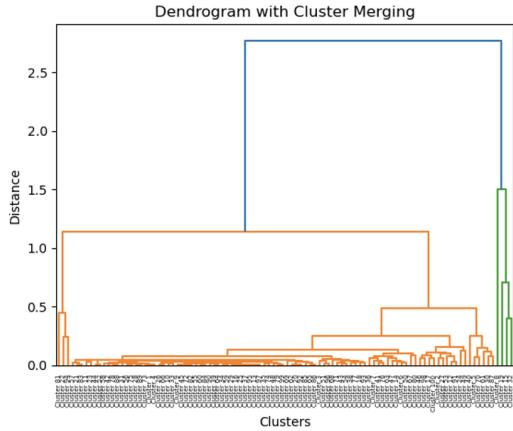


FIG. 4.4. *Hierarchical Clusters producing 18 clusters from 100 original created by k-means.*

a more advanced manner. The Affinity Propagation algorithm refrains from the explicit

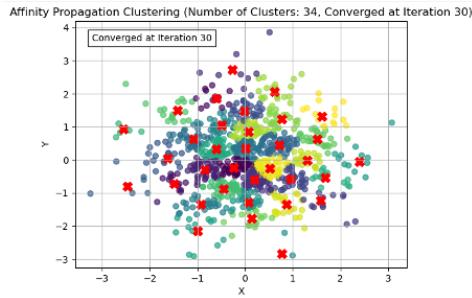


FIG. 4.5. *Affinity Propagation (damping factor= 0.6)*

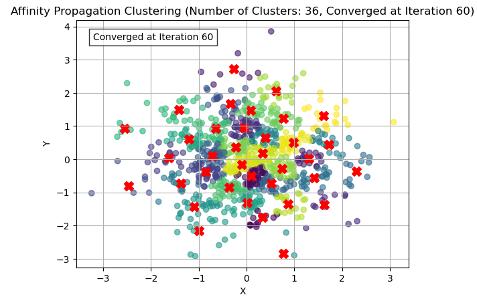


FIG. 4.6. *Affinity Propagation (damping factor= 0.7)*

notion of cluster centroids and assigns data points as exemplars based on a similarity matrix. Through a process of message passing, data points converge toward exemplars, forming clusters around them.

The results obtained from applying Affinity Propagation show intriguing insights. Notably, the clustering outcomes reveal a higher degree of distinctiveness among clusters compared to the results from traditional methods. This distinctiveness is evident in both the covariance matrix analysis and the evaluation metrics. The covariance matrix demonstrates reduced covariance scores between data points in different clusters, signifying clearer boundaries and more separation.

Quantitative metrics corroborate the distinct clusters' observations. A Silhouette score of 0.545 indicates an improvement in cluster separation and cohesion, underscoring the enhanced quality of clusters generated by Affinity Propagation. Additionally, the Davies-Bouldin Index, with a value of 0.903, further reinforces the effectiveness of this advanced technique in forming well-separated and cohesive clusters.

**4.3. Impact on Linear Solvers.** This section presents the implications of enhanced clustering techniques. This includes the comparison of all clusters mentioned in previous sections on the performance of multigrid solvers. The insights gained from these advanced clustering methodologies can have tangible benefits when applied to the realm of linear

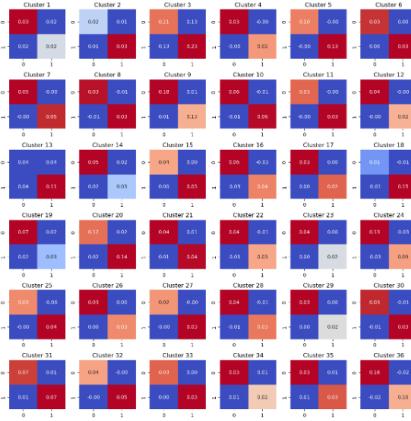


FIG. 4.7. Heat map of Covariance Matrix for Affinity Propagation Clustering

solving.

In reference to examining the performance of multigrid solvers using cluster assignments, interesting observations were made. Beginning with the K-means algorithm, despite variations in the clustering techniques, the number of iterations required for solving at a specific level in a multigrid, K-means clustering and its merged counterpart yield comparable outcomes in terms of solver convergence behavior at this multigrid level.

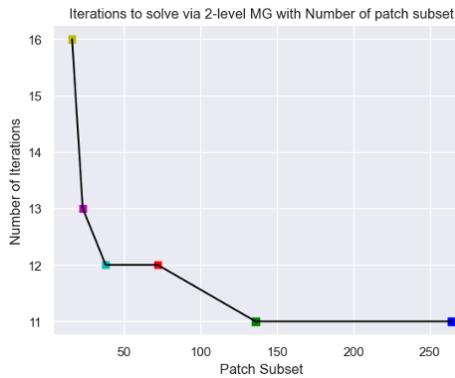


FIG. 4.8. Full Means clusters fed into multi-grid solver)

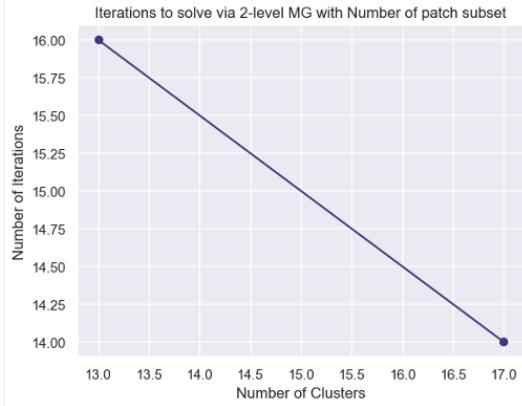


FIG. 4.9. Merged Clusters fed into multi-grid solver)

In comparison, the application of Affinity Propagation introduces a distinctive advantage. The number of iterations necessary for solving, as well as the quantity of clusters utilized, is reduced compared to K-means. This reduction in computational overhead stems from the nature of Affinity Propagation to form clusters that are more representative of the dataset's underlying structure. The clusters require fewer iterations for convergence, translating to a faster multigrid-solving process.

The benefits of employing advanced algorithms like Affinity Propagation extend beyond computational efficiency. The ability to generate clusters with clearer boundaries and minimized overlap directly influences multigrid solvers' accuracy and convergence rates. The reduced complexity in inter-cluster relationships, as evidenced by lower Davies-Bouldin

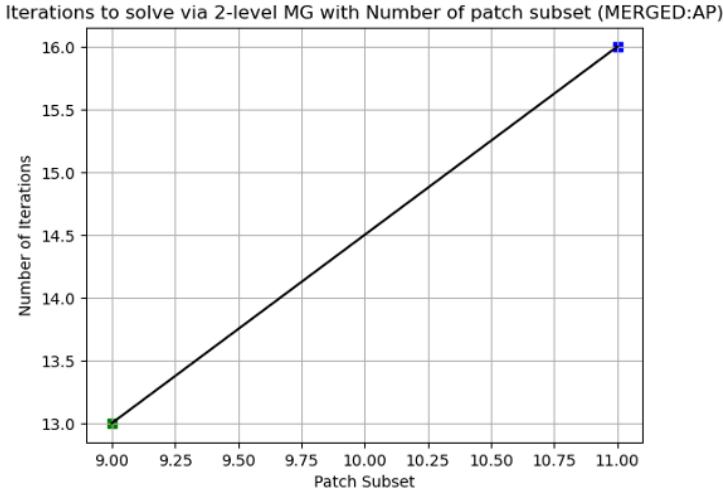


FIG. 4.10. *Affinity Propagation fed into multigrid solver*

Index values, guides the solver's trajectory with greater precision, contributing to more effective linear solving.

Algorithm	Silhouette Score (S)	Davies-Bouldin Index (DBI)
K-means	0.3574	0.8935
Hierarchical	0.2912	0.6148
Affinity Propagation	0.5455	0.9028

TABLE 4.1

*Clustering Performance Evaluation Comparison, Each performance metric corresponds to the clusters in the result section*

**5. Conclusion.** With the goal of accelerating patch-based linear solvers by uncovering inherent dataset structures, this project explored matrix clustering techniques. The specific aim was to assess the impact of advanced clustering algorithms on linear solver performance, enriching data analysis, pattern recognition, and computational methods.

At the center of the investigation was a comparison of clustering techniques, such as affinity propagation clustering, with simpler ones like K-means. While the idea is promising, it's important to acknowledge a limitation concerning cluster count determination. No formal techniques like the elbow method were implemented, rather cluster counts in traditional approaches were simply done via brute force. This may have potentially affected precision.

Despite this, the potential of advanced clustering is clear, exemplified by affinity propagation's effectiveness in reducing solver iterations and clusters. The contribution of this work can extend to data analysis and computational techniques, emphasizing the combination of clustering insights and linear solver efficiency.

## REFERENCES

- [1] P. B. BOCHEV, C. R. DOHRMANN, AND R. S. TUMINARO, *A new algebraic multigrid approach for saddle point problems*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1266–1288.

- [2] P. BRUBECK AND P. FARRELL, *A scalable and robust vertex-star relaxation for high-order fem*, SIAM Journal on Scientific Computing, 44 (2022), pp. A2991–A3017.
- [3] C. R. DOHRMANN AND P. B. BOCHEV, *Patch-based algebraic multigrid for quadrilateral meshes*, SIAM Journal on Scientific Computing, 35 (2013), pp. A219–A243.
- [4] N. ETIENNE AND E. AGU, *Investigating transfer learning of smartphone-sensed stress in university populations*, 2020 IEEE International Conference on Big Data (Big Data), (2020), pp. 4850–4858.
- [5] H. FINKEL, T. FISHER, AND A. JENSEN, *Patching technique for parallel matrix factorization in structural mechanics*, International Journal for Numerical Methods in Engineering, 44 (1999), pp. 175–194.
- [6] B. J. FREY AND D. DUECK, *Clustering by passing messages between data points*, Science, 315 (2007), pp. 972–976.
- [7] O. GHATTAS AND M. TOBIS, *Analysis of upwind finite element discretizations on nonmatching grids*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 1215–1230.
- [8] G. HARPER AND R. TUMINARO, *Compression and reduced representation techniques for patch-based relaxation*, arXiv preprint arXiv:2306.10025, (2023).
- [9] T. Y. HOU AND X.-H. WU, *A multiscale finite element method for elliptic problems in composite materials and porous media*, Journal of Computational Physics, 134 (1997), pp. 169–189.
- [10] J. LEE AND E. KIM, *Comparative analysis of affinity propagation and k-means clustering for discerning intricate clusters*, Journal of Data Science and Analysis, 10 (2022), pp. 123–140.
- [11] J. MACQUEEN, *Some methods for classification and analysis of multivariate observations*, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1 (1967), pp. 281–297.
- [12] A. B. SMITH, C. D. JOHNSON, AND E. F. WILLIAMS, *Investigating the application of hierarchical clustering for improving patch-based linear solvers*, Journal of Computational Mathematics, 15 (2018), pp. 567–586.
- [13] Y. VASSILEVSKI AND R. D. LAZAROV, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, SIAM Journal on Scientific Computing, 29 (2006), pp. 1791–1817.

## AN OPTIMIZATION-BASED APPROACH FOR COUPLING PROJECTION-BASED REDUCED ORDER MODELS

ELIZABETH HAWKINS, PAVEL BOCHEV, AND PAUL KUBERRY

**Abstract.** In this paper we develop a partial differential equation (PDE) constrained optimization technique, based on [7] and developed in the context of the finite element method (FEM), for a coupled advection-diffusion equation and extend it to reduced order models (ROMs). This coupling method [7] utilizes the adjoint problem to define a gradient descent algorithm, but can require a large amount of iterations and can be computationally expensive for coupling full order finite element models. Exploiting the reduced computational complexity of each optimization iteration through projection-based reduced order modeling, we recast the PDE-constrained optimization problem in a ROM-ROM coupled setting and explore dependence on the penalty parameter, the reduced basis size and the choice of reduced basis for the accompanying adjoint solves.

**1. Introduction.** PDE-constrained optimization is an established technique for the solution of coupled problems with different governing equations and physical parameters [1, 6]. Additionally, these approaches can be extended to non-coincident interfaces where subdomain meshing is performed independently, resulting in gaps and overlaps for non-planar interfaces [13]. The solution of stationary points of the Lagrangian involves primal and adjoint solves, which can be prohibitively expensive for traditional full-order models (FOMs). Recent developments in projection-based reduced order models (ROMs), machine learned surrogates such as physics informed neural networks (PINNs), and system identification surrogate modeling techniques such as dynamic mode decomposition (DMD) give an occasion for revisiting these coupling techniques in order to evaluate their adequacy in addressing multifidelity coupling. Iterative solution techniques for PDE-constrained optimization problems can be slow to converge and expensive to compute depending on the algorithm and problem, while the ROM analogs are less computationally expensive.

Applications of PDE-constrained optimization as a domain decomposition technique was introduced in [8] in the context of a finite element discretization of Poisson's equation. This was later extended to Navier-Stokes equations in [7] and later to fluid-structure interaction in [14]. Both the gradient descent algorithm and Gauss-Newton with the conjugate gradient algorithm were developed and applied in these. The gradient descent approach utilized the adjoint problem of the PDE to find a descent direction. M. Gunzberger and H. K. Lee, in [7], showed that the finite element implementation of this gradient descent algorithm solved for the true solution as accurately as their Gauss-Newton method with convergence rates and accuracy depending on the mesh size. These methods developed in [7] involve solving finite element formulations of the primal and adjoint problems.

Projection-based reduced order modeling using the Proper Orthogonal Decomposition (POD)/Galerkin method was first developed for applications to turbulence in [15] and [11]. This has since been extended to many other applications of finite elements by projecting the finite element operators onto a reduced space, with approaches for coupling problems with overlapping and nonoverlapping interfaces being no exception. In [12], a port and bubble basis are introduced, enabling static condensation of the bubble degrees of freedom (DOF) and writing the loss function entirely with respect to the port DOF. Their approach uses the port DOF as the control, unlike our approach which uses the interface traction force. Additionally, it is developed in the context of overlapping subdomains, whereas we are targeting a non-overlapping subdomain setting.

Applying existing FOM-FOM coupling techniques for ROM-FOM and ROM-ROM couplings can result in unexpected difficulties that require consideration specific to the introduction of ROMs. An example of this is [3]. This paper considers the problem of domain decomposition (DD) of the physical geometry into non-overlapping subdomains applied to an advection-diffusion equation that is then coupled together. This problem was solved using a direct Lagrange multiplier method, but the introduction of ROMs introduced a Ladyzhenskya-Babuska-Brezzi (LBB) instability requiring an alternate construction for the reduced order basis. The purpose of this project was to make a similarly motivated investigation into the amenability of adapting the existing coupling technique in [8] to the ROM setting.

In this paper, we use the gradient descent algorithm to solve the PDE-constrained optimization problem for the advection-diffusion equation, similar to [8]. We then adapt the method to a projection-based ROM setting in order to improve the computational time in each optimization iteration. Numerical results are provided, comparing the optimization iterations and accuracy of the ROM-ROM and FOM-FOM coupling approaches.

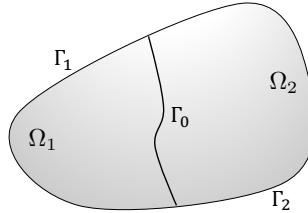


FIG. 1.1. Example of two non-overlapping domains sharing an interface

**1.1. Model problem and coupling formulation.** Let  $\Omega_1$  and  $\Omega_2$  be non-overlapping, bounded domains in  $\mathbb{R}^2$  with the boundary  $\partial\Omega_i$ , and let  $\Gamma_0$  denote the interface between  $\Omega_1$  and  $\Omega_2$ . We define  $\Gamma_1 = \partial\Omega_1/\Gamma_0$  and similarly  $\Gamma_2 = \partial\Omega_2/\Gamma_0$ . Let  $u_i$  denote the concentration,  $\nu_i$  is the diffusivity constant, and  $\mathbf{a}_i$  the advection field in  $\Omega_i$ . Then the advection-diffusion equation on  $\Omega_i$  is given by

$$\begin{cases} u_{i,t} - \nabla \cdot \sigma_i(u_i) = f_i & \text{in } \Omega_i \\ u_i = \beta_i & \text{on } \partial\Omega_i \end{cases}$$

where  $\sigma_i(u) = \nu_i \nabla u - \mathbf{a}_i \cdot u$ . We couple these equations using a boundary conditions on  $\Gamma_0$ . This coupled system is then, for  $i = 1, 2$

$$\begin{cases} u_{i,t} - \nabla \cdot \sigma_i(u_i) = f_i & \text{in } \Omega_i \\ u_i = \beta_i & \text{on } \Gamma_i \\ u_1 = u_2 & \text{on } \Gamma_0 \\ \sigma_1(u_1) \cdot n_1 = -\sigma_2(u_2) \cdot n_2 & \text{on } \Gamma_0 \end{cases}$$

where  $n_i$  is the outward unit normal vector on  $\Gamma_i$ . These types of scenarios occur in multi-physics problems, domain decomposition problems, and transmission problems. A benefit of this setting is that it allows either a ROM or a FOM to be used in either domain.

We can rewrite and decouple this system using a control. While there are many choices on how to introduce a control, we follow the method by Gunzburger et al [8] and introduce an unknown function  $g \in L^2(\Gamma_0)$  that satisfies

$$\begin{cases} u_{i,t} - \nabla \cdot \sigma_i(u_i) = f_i & \text{in } \Omega_i \\ u_i = \beta_i & \text{on } \Gamma_i \\ \sigma_i(u_i) \cdot n_i = (-1)^i g & \text{on } \Gamma_0. \end{cases}$$

Note that the interface boundary condition  $u_1 = u_2$  on  $\Gamma_0$  is left out. This is enforced later in the optimization problem, by the introduction of the measure of violation of this constraint in the loss functional. Letting  $g = \sigma_2(u_2) \cdot n_2$ , it is clear that this system is equivalent to the original coupled system. Furthermore, if  $g$  is known, then both of the advection-diffusion equations are independent and consequently can be solved independently.

Denote  $(\cdot, \cdot)$  as the  $L^2$  inner product on  $\Omega_i$  for  $i = 1, 2$  and  $(u, v)_{\Gamma_i}$  as the  $L^2$  inner product on  $\Gamma_i$  for  $i = 0, 1, 2$ . We discretize in time using backward Euler to get the weak formulation.

Find  $u_i^n \in X_i^n$  satisfying

$$\frac{1}{\Delta t}(u_i^n - u_i^{n-1}, v) + (\sigma_i(u_i^n), \nabla v) = (f_i^n, v) + (-1)^i(g^n, v)_{\Gamma_0} \quad \forall v \in V_i, \quad (1.1)$$

where  $X_i^n = \{u \in H^1(\Omega_i) | u = \beta(x, t^n) \text{ on } \Gamma_i\}$ ,  $V_i = \{v \in H^1(\Omega_i) : v|_{\Gamma_i} = 0\}$ ,  $g^n := g(x, t^n)$ , and  $f_i^n := f_i(x, t^n)$  for  $i = 1, 2$ . Note that we will discretize in space via the finite element method later in the paper.

We define the functional

$$J_\delta(u_1^n, u_2^n, g^n) := \frac{1}{2}\|u_1^n - u_2^n\|_{\Gamma_0}^2 + \frac{1}{2}\delta\|g^n\|_{\Gamma_0}^2. \quad (1.2)$$

This functional enforces the interface boundary condition  $u_1 = u_2$  on  $\Gamma_0$  at time  $t^n$  by minimizing  $\|u_1^n - u_2^n\|_{\Gamma_0}^2$  and also includes regularization of the optimization problem by minimizing the  $L^2$  norm of  $g^n$ . The purpose of the stabilization term is to assist convergence of the gradient descent approach. Additionally, this term is of value when considering nonlinear governing equations, although they are not considered in this paper. In the case of  $\delta = 0$ , an optimal solution will only enforce  $u_1 = u_2$  weakly on  $\Gamma_0$ . However, for  $\delta > 0$ , the second term competes with the first, ultimately limiting how well we can enforce  $u_1 = u_2$ . This balance between terms should be considered when choosing a value for  $\delta$ .

We want to solve the minimization problem

$$\begin{aligned} & \min J_\delta(u_1^n, u_2^n, g^n) \\ & (u_1^n, u_2^n, g^n) \in X_1^n \times X_2^n \times L^2(\Gamma_0) \\ & \text{s.t. (1.1).} \end{aligned} \quad (1.3)$$

To do this, we find the optimality conditions given by the partial derivatives of the minimization problems Lagrangian. The Lagrangian is given by

$$\begin{aligned} \mathcal{L}(u_1^n, u_2^n, g^n, \mu_1, \mu_2) := & J_\delta(u_1^n, u_2^n, g^n) + \sum_{i=1}^2 \left[ \frac{1}{\Delta t}(u_i^n - u_i^{n-1}, \mu_i) + (\sigma_i(u_i), \nabla \mu_i) \right. \\ & \left. - (f_i^n, \mu_i) + (-1)^{i+1}(g^n, \mu_i)_{\Gamma_0} \right] \end{aligned} \quad (1.4)$$

where  $\mu_i \in V_i$  is the adjoint variable. We take the gradient of  $\mathcal{L}$  with respect to  $u_i^n$  and set it equal to 0 to get the adjoint equation

$$\frac{1}{\Delta t}(\eta, \mu_i) + (\sigma_i(\eta), \nabla \mu_i) = (-1)^i(\eta, u_1^n - u_2^n)_{\Gamma_0} \quad \forall \eta \in X_i^n. \quad (1.5)$$

Similarly, the gradient of  $\mathcal{L}$  with respect to  $g^n$  set equal to 0 yields

$$\delta(\psi, g^n)_{\Gamma_0} = -(\psi, \mu_1 - \mu_2)_{\Gamma_0} \quad \forall \psi \in L^2(\Gamma_0). \quad (1.6)$$

Note that the gradient of  $\mathcal{L}$  with respect to  $\mu_i$  will return (1.1). These equations (1.5), (1.6), and (1.1) are the optimality conditions that the minimizer must satisfy.

**2. Solution of the optimization problem via the gradient descent method.** It is clear that we can define  $u_i^n$  as a function of  $g^n$  because  $g^n$  determines a boundary condition of  $u_i^n$ . So, we can define  $\mathcal{M}_\delta$  by rewriting  $J_\delta$  as a function of  $g^n$

$$\mathcal{M}_\delta(g^n) := J_\delta(u_1^n(g^n), u_2^n(g^n), g^n)$$

where  $u_i^n(g^n)$  is the solution to (1.1) for the given  $g^n$ . The gradient of  $\mathcal{M}_\delta(g^n)$  with respect to variation in the control,  $\tilde{g}^n$ , is given by

$$\frac{d\mathcal{M}_\delta(g^n)}{dg^n} \cdot \tilde{g}^n = \delta(g^n, \tilde{g}^n)_{\Gamma_0} + (u_1^n - u_2^n, \tilde{u}_1^n - \tilde{u}_2^n)_{\Gamma_0} \quad \forall \tilde{g}^n \in L^2(\Gamma_0), \quad (2.1)$$

where  $\tilde{u}_i^n$  is the solution to the sensitivity equation

$$\frac{1}{\Delta t}(\tilde{u}_i^n, v) + (\sigma_i(\tilde{u}_i^n), \nabla v) = (-1)^i(\tilde{g}^n, v)_{\Gamma_0} \quad \forall v \in V_i. \quad (2.2)$$

In (2.2) let  $v = \mu_i$  and in (1.5) let  $\eta = \tilde{u}_i^n$ . Combining these we get

$$(\tilde{g}^n, \mu_1 - \mu_2)_{\Gamma_0} = (\tilde{u}_1^n - \tilde{u}_2^n, u_1^n - u_2^n)_{\Gamma_0}.$$

Using this we get that the gradient of the functional is

$$\frac{d\mathcal{M}_\delta(g^n)}{dg^n} \cdot \tilde{g}^n = \delta(g^n, \tilde{g}^n)_{\Gamma_0} + (\tilde{g}^n, \mu_1 - \mu_2)_{\Gamma_0} \quad \forall \tilde{g}^n \in L^2(\Gamma_0). \quad (2.3)$$

Because  $\tilde{g}^n$  is an arbitrary function, it must hold that

$$\frac{d\mathcal{M}_\delta(g^n)}{dg^n} = \delta g^n + (\mu_1 - \mu_2)_{\Gamma_0}.$$

This allows us to use the gradient descent method where the  $m^{th}$  iteration is defined by

$$g^{n,(m)} = g^{n,(m-1)} - \alpha \frac{d\mathcal{M}_\delta(g^n)}{dg^n}$$

by simplifying the gradient descent formula to

$$g^{n,(m)} = (1 - \alpha \delta)g^{n,(m-1)} - \alpha(\mu_1^{(m-1)} - \mu_2^{(m-1)})|_{\Gamma_0},$$

where  $\alpha$  is some step size and  $\mu_i^{(m-1)}$  is the solution to the adjoint equation for  $g^{n,(m-1)}$ . This method follows the approach in [8] which has been generalized in [10].

For any iterative optimization algorithm, a stopping criteria is required. We halt the optimization algorithm when  $\frac{\|u_1^n - u_2^n\|_{\Gamma_0}}{\|u_1^n\|_{\Gamma_0}} < tol$ , where  $u_i^n$  is the state equation solution on  $\Omega_i$  for a particular  $g^{n,(m)}$ . Recall that the purpose of the functional  $J_\delta$  is to minimize  $\|u_1^n - u_2^n\|_{\Gamma_0}$ . The choice for  $\alpha$  at each optimization iteration can be chosen by the user, either empirically or through a more sophisticated approach such as the use of a line search.

---

**Algorithm 1** Continuous form of the gradient descent method

---

Given  $g^{n,(0)}$

**for**  $m = 0, 1, \dots$  **do**

Solve (1.1) using  $g^{n,(m)}$  for  $u_1^n$  and  $u_2^n$

Solve (1.5) using  $g^{n,(m)}$ ,  $u_1^n$ , and  $u_2^n$  for  $\mu_1$  and  $\mu_2$

$$g^{n,(m+1)} = (1 - \alpha\delta)g^{n,(m)} - \alpha(\mu_1^{(m)} - \mu_2^{(m)})|_{\Gamma_0}$$

**end for**

---

**2.1. Application to the coupling of finite element models.** We discretize in space using FEM basis functions to get the Galerkin formulation of (1.1).

Find  $u_i^{h,n} \in X_i^{n,h}$  satisfying

$$\frac{1}{\Delta t}(u_i^{h,n} - u_i^{h,n-1}, v^h) + (\sigma_i(u_i^{h,n}), \nabla v^h) = (f_i^n, v^h) + (-1)^i(g^n, v^h)_{\Gamma_0}, \quad \forall v^h \in V_i^h. \quad (2.4)$$

Using the finite element basis of  $V_i^h$  given by  $\{\phi_{i,k}\}_{k=1}^{N_i}$  in (2.4) gives the following matrix equation.

Find  $\bar{u}_i^n \in \mathbb{R}^{N_i}$  satisfying

$$\frac{1}{\Delta t}M_i\bar{u}_i^n + (\nu_i K_i + A_i)\bar{u}_i^n = \bar{f}_i^n + \frac{1}{\Delta t}M_i\bar{u}_i^{n-1}, \quad (2.5)$$

where  $M_i$  is the mass matrix,  $K_i$  is the stiffness matrix,  $(A_i)_{k,j} := (a \cdot \phi_{i,k}, \nabla \phi_{i,j})$ , and  $(\bar{f}_i^n)_k := (f_i^n, \phi_{i,k}) + (-1)^i(g^n, \phi_{i,k})_{\Gamma_0}$ . Without loss of generality, let  $g^n \in V_1^h$  such that  $\bar{g}^n \in \mathbb{R}^{N_1}$  and  $g^n|_{\Omega_1 \setminus \Gamma_0} = 0$ .

One popular way of dealing with non-homogeneous Dirichlet boundary conditions in finite elements is to eliminate known degrees of freedom from the rows and columns of the matrix, adjusting the right hand side accordingly. The  $M_i$ 's above do not reflect this modification, and this is only mentioned here as it becomes relevant for their reference in the ROM-specific section that follows.

We will also need to solve the adjoint equation (1.5) using FEM. So we discretize in space using the same basis as (2.4) to get the discrete adjoint equation

$$\frac{1}{\Delta t}(\eta^h, \mu_i^h) + (\sigma_i(\eta^h), \nabla \mu_i^h) = (-1)^i(\eta^h, u_1^{h,n} - u_2^{h,n})_{\Gamma_0}, \quad \forall \eta^h \in X_i^{n,h}. \quad (2.6)$$

Using the same process as with the weak form, the discrete adjoint equation is equivalent to the matrix equation

$$\frac{1}{\Delta t} M_i \bar{\mu}_i + (\nu_i K_i + A_i^T) \bar{\mu}_i = (-1)^i M_{\Gamma_0, i} (\bar{u}_1 - \bar{u}_2). \quad (2.7)$$

We can use (2.4) and (2.6) to solve (1.1) and (1.5) with the continuous gradient descent algorithm (2). Then, the FEM gradient descent algorithm solves for the coefficient vector  $\bar{g}^n$  and uses the vector solutions to (2.4) and (2.6) to update  $\bar{g}^n$  at each iteration. The algorithm that follows uses  $I_{i \rightarrow 0}$  to denote an interpolation operator from  $X_i^{n,h}$  to the space of the control  $\bar{g}^n$ , for  $i=1,2$ .

---

**Algorithm 2** Finite element form of the gradient descent method

---

Given  $\bar{g}^{n,(0)}$

**for**  $m = 0, 1, \dots$  **do**

Solve (2.4) using  $\bar{g}^{n,(m)}$  for  $\bar{u}_1^{h,n}$  and  $\bar{u}_2^{h,n}$

Solve (2.6) using  $\bar{g}^{n,(m)}$ ,  $\bar{u}_1^{h,n}$ , and  $\bar{u}_2^{h,n}$  for  $\bar{\mu}_1^h$  and  $\bar{\mu}_2^h$

$$\bar{g}^{n,(m+1)} = (1 - \alpha\delta)\bar{g}^{n,(m)} - \alpha(I_{1 \rightarrow 0}\bar{\mu}_1^{h,(m)} - I_{2 \rightarrow 0}\bar{\mu}_2^{h,(m)})|_{\Gamma_0}$$

**end for**

---

REMARK 1. Project work for this proceeding only included investigating matched meshes at the interface, with  $I_{1 \rightarrow 0}$  as the identity operator and  $I_{2 \rightarrow 0}$  as a permutation matrix that matches degrees of freedom from  $X_2^{n,h}$  on the interface consistent with DOF ordering for  $\Omega_1$  (consistent with the definition of  $g^n$ ).

REMARK 2. Applying the Galerkin finite element method to (1.1) on moderately refined meshes can produce strongly oscillatory solutions. We stabilize using the Streamline upwind Petrov-Galerkin (SUPG) stabilization approach [2]. This introduces an option to either include the SUPG stabilization terms in the state equations and then derive the adjoint equation (discretize-then-optimize), or to formulate the optimization problem as in (1.3) and then add SUPG stabilization to (1.1) and (1.5), consistent with the respective subproblem (optimize-then-discretize). We choose to optimize-then-discretize later in this work, which is similar in accuracy for linear finite elements and more accurate for higher order finite elements when compared with the discretize-then-optimize approach. A thorough investigation contrasting the discretize-then-optimize and optimize-then-discretize approaches, applied to advection-diffusion equations, can be found in [4].

**2.2. Application to the coupling of projection-based ROMs.** Before describing adaptation of the PDE-constrained coupling approach from FOM-FOM to ROM-ROM coupling, we will briefly describe decisions made in our projection-based reduced order modeling. The generation of our ROM basis closely follows the detailed description in [5]. Specific to this work, we run a monolithic FEM solver over a uniform timestep and store snapshots of the solution by segregating the DOFs from the monolithic solution into two snapshot matrices, each snapshot matrix containing portions relevant to that subdomain. We note that there are many other ways to generate these snapshot matrices corresponding to each subdomain. Similar to [5], we remove contributions to the snapshot columns for DOFs corresponding to Dirichlet nodes. A singular value decomposition of each of the snapshot

matrices, altered as described and related to boundary conditions, is performed. A threshold criteria is often used to determine how many singular values and left singular vectors are retained. In this work, we choose how many singular vectors to keep independent of a threshold criteria.

Let  $\{\psi_{i,k}\}_{k=1}^{N_{i,r}}$  be a ROM basis associated with the solution to (2.4) where  $N_{i,r} << N_i$  and  $N_i$  is the size of the FEM discretized system in  $\Omega_i$ . We want to solve the weak form (2.4) using this reduced basis instead of the finite element basis and we do this through a change of basis operation from the FEM basis to the ROM basis. The change of basis operation takes the form

$$\Psi_i \hat{u}_i^n + \bar{\beta}_i^n = \bar{u}_i^n \text{ for some } \hat{u}_i \in \mathbb{R}^{N_{i,r}} \quad (2.8)$$

where  $\Psi_i \in \mathbb{R}^{N_i \times N_{i,r}}$  is the matrix whose columns are the ROM basis elements and  $\bar{\beta}_i^n \in \mathbb{R}^{N_i}$  is a column vector such that

$$(\bar{\beta}_i^n)_k := \begin{cases} 0 & \text{if } x_k \text{ is not a boundary node on } \Gamma_i \\ \beta_i(x_k, t^n) & \text{if } x_k \text{ is a boundary node on } \Gamma_i \end{cases}.$$

Note that this change of basis accounts for the non-homogeneous Dirichlet boundary conditions so the matrix equations do not need to be modified in the way previously described for FEM. This method of boundary condition enforcement for ROM is described in greater detail as Method 1 in [9].

We use this substitution for  $\bar{u}_i^n$  and left multiply by  $\Psi_i^T$  in (2.5) to get the ROM matrix equation for the Galerkin weak form.

$$\text{Find } \hat{u}_i^n \in \mathbb{R}^{N_r} \text{ satisfying } \frac{1}{\Delta t} \hat{M}_i \hat{u}_i^n + (\nu_i \hat{K}_i + \hat{A}_i) \hat{u}_i^n = \hat{f}_i^n + \frac{1}{\Delta t} \hat{M}_i \hat{u}_i^{n-1} \quad (2.9)$$

where  $\hat{M}_i := \Psi_i^T M_i \Psi_i$ ,  $\hat{K}_i = \Psi_i^T K_i \Psi_i$ ,  $\hat{A}_i := \Psi_i^T A_i \Psi_i$ , and  $\hat{f}_i^n := \Psi_i^T \bar{f}_i^n - \Psi_i^T M_i (\bar{\beta}_i^n - \bar{\beta}_i^{n-1}) - (\nu_i K_i + A_i) \bar{\beta}_i^n$ .

We also use the change of basis operation on  $\bar{\mu}_i$  given by

$$\Psi_i \hat{\mu}_i = \bar{\mu}_i \text{ for some } \hat{\mu}_i \in \mathbb{R}^{N_r}$$

and apply the same process to (2.7) to get the ROM adjoint matrix equation

$$\frac{1}{\Delta t} \hat{M}_i \hat{\mu}_i + (\nu_i \hat{K}_i + \hat{A}_i^T) \hat{\mu}_i = (-1)^i (\Psi_i^T M_{\Gamma_0, i} (\Psi_1 \hat{u}_1^n - \Psi_2 \hat{u}_2^n) + \Psi_i^T (\bar{\beta}_1^n - \bar{\beta}_2^n)). \quad (2.10)$$

**REMARK 3.** Note that the change of basis for  $\bar{\mu}_i$  is the same as what was used in (2.8) for  $\hat{u}_i^n$ . This is because, in this work, we use a reduced basis for the adjoint problem which is identical to the reduced basis for the state problem. Obviously, it would be ideal to collect snapshots of the adjoint problem, however this would require solving the FOM-FOM coupled problem, which is what we are trying to avoid. Future work will include investigating alternative snapshot sources from which to generate the adjoint reduced bases.

We can use (2.9) and (2.10) to solve (2.4) and (2.6) in the finite element gradient descent method (2). However, because we've chosen to keep  $\bar{g}^n$  in the full order space we must project the reduced space solutions to the adjoint problems into the full order space in order to subtract them from  $\bar{g}^n$ . This projection is simply  $\bar{\mu}_i^h = \Psi_i \hat{\mu}_i^h$ . Again, in the algorithm

that follows, we use  $I_{i \rightarrow 0}$  to denote an interpolation operator from  $X_i^{n,h}$  to the space of the control  $\bar{g}^n$ , for  $i=1,2$ .

---

**Algorithm 3** ROM form of the gradient descent method

---

Given  $\bar{g}^{n,(0)}$

**for**  $m = 0, 1, \dots$  **do**

Solve (2.9) using  $\bar{g}^{n,(m)}$  for  $\hat{u}_1^{h,n}$  and  $\hat{u}_2^{h,n}$

Solve (2.10) using  $\bar{g}^{n,(m)}$ ,  $\hat{u}_1^{h,n}$ , and  $\hat{u}_2^{h,n}$  for  $\hat{\mu}_1^h$  and  $\hat{\mu}_2^h$

$$\bar{g}^{n,(m+1)} = (1 - \alpha\delta)\bar{g}^{n,(m)} - \alpha(I_{1 \rightarrow 0}\Psi_1\hat{\mu}_1^{h,(m)} - I_{2 \rightarrow 0}\Psi_2\hat{\mu}_2^{h,(m)})|_{\Gamma_0}$$

**end for**

---

**3. Numerical Results.** Consider the problem of a cylinder rotating in a given domain  $\Omega = (0, 1) \times (0, 1)$ . We arbitrarily split  $\Omega$  into non-overlapping domains  $\Omega_1 = (0, 0.5) \times (0, 1)$  and  $\Omega_2 = (0.5, 1) \times (0, 1)$  and denote their interface as  $\Gamma_0$ . This gives the coupled system of PDEs

$$\begin{cases} u_{i,t} - \sigma_i(u_i) = 0 & \text{in } \Omega_i \\ u_i(x, t) = 0 & \text{on } \Gamma_i \\ \nabla u_i \cdot n_i = (-1)^i g & \text{on } \Gamma_0 \end{cases}$$

where  $\nu_i = 10^{-5}$ ,  $\mathbf{a}_i(x, t) := (0.5 - y, x - 0.5)$ , and  $g$  is an unknown function defined on  $\Gamma_0$ .

We consider the rotating cylinder problem with  $T = 2\pi$  and discretize in time using backward Euler with  $\Delta t = 1.122398 \cdot 10^{-3}$ . This time step size is determined from the Courant-Friedrichs-Lowy (CFL) condition. A  $64 \times 64$  mesh is used to spatially discretize  $\Omega$  resulting in 4225 DOFs; this mesh is specially constructed so that it is easily split into distinct meshes for  $\Omega_1$  and  $\Omega_2$ . Standard Lagrange  $\mathbb{Q}^1$  elements are used to solve the monolithic problem which we use as a reference solution. For the coupled problem, we use equally refined but separate meshes, which results in 2145 DOFs in both subdomains. Because we use the same basis elements for the FOM-FOM coupled problem as for the monolithic problem, and cell boundaries are coincident with the physical interface  $\Gamma_0$ , it is straight-forward to decompose the monolithic solution onto the subdomains.

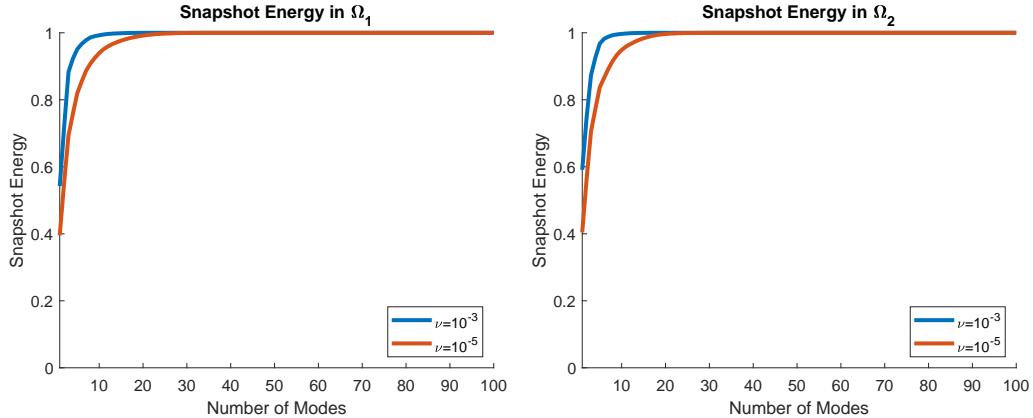


FIG. 3.1. Comparison of snapshot energy in  $\Omega_1$  (left) and  $\Omega_2$  (right) for values of  $\nu$  as a function of the number of modes retained for the reduced basis

In Figure 3.1, we plot the snapshot energy as a function of the number of modes retained in each subdomain and compare between values of  $\nu$ , corresponding to the amount of diffusion in the advection-diffusion problem. As expected, fewer modes are needed to capture equal amounts of snapshot energy with increased diffusion.

We first set the penalty parameter  $\delta = 10^{-7}$  and the initial gradient descent step size as  $\alpha = 3.2$ , which is reduced in a line search manner during each gradient descent method iteration. We choose  $g$  to be defined on  $\Omega_1$ , restricted to it is trace on  $\Gamma_0$ , and choose the initial  $g^{(0)}$  to be a vector of zeros. We compare the solution to the FOM-FOM coupled problem  $u_c$ , where  $u_c$  is the re-combination of the solutions in  $\Omega_1$  and  $\Omega_2$ , to the FEM solution to the monolithic problem,  $u_m$ . At time  $t = 2\pi$  we have

$$\begin{aligned} \frac{\|u_c - u_m\|_{L^2(\Omega)}}{\|u_m\|_{L^2(\Omega)}} &= 1.2768 \cdot 10^{-9} \\ \frac{\|u_c - u_m\|_{H^1(\Omega)}}{\|u_m\|_{H^1(\Omega)}} &= 2.9513 \cdot 10^{-9}. \end{aligned}$$

Hence, the solution to the FOM-FOM coupled problem gives nearly the same solution as the monolithic problem, where the error is on the order of the penalty term ( $\delta$ ). This is also shown by the graph of  $u_c$  and  $u_m$  on the interface, right on Figure 3.2, and the graph of  $u_c$  on the entire domain, left on Figure 3.2. The monolithic solution shown on the right of Figure 3.3 has initial conditions shown on the left of Figure 3.3, demonstrating that the monolithic FEM solution gives an accurate representation of what the objects should look like after rotation with a small amount of diffusion.

REMARK 4. We also applied the FOM-FOM coupling and gradient descent algorithm to a problem with a linear in space and time manufactured solution. For  $\delta = 10^{-14}$  we achieved a solution with error on the order of machine precision.

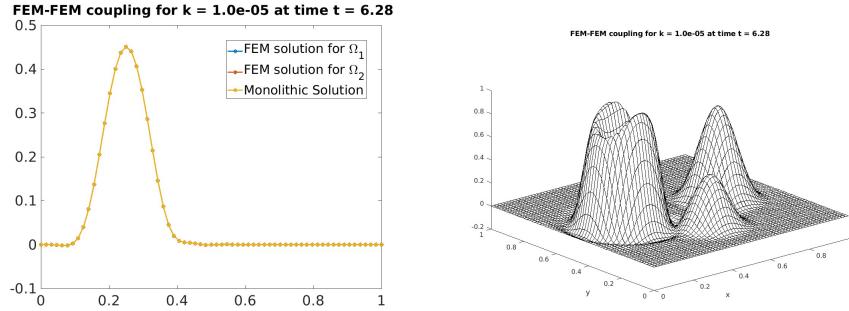
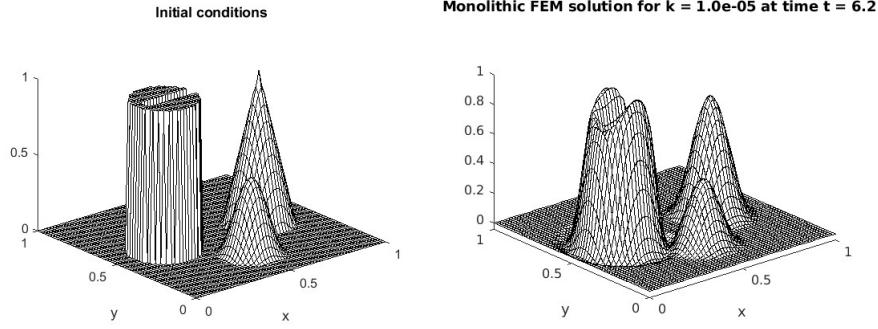
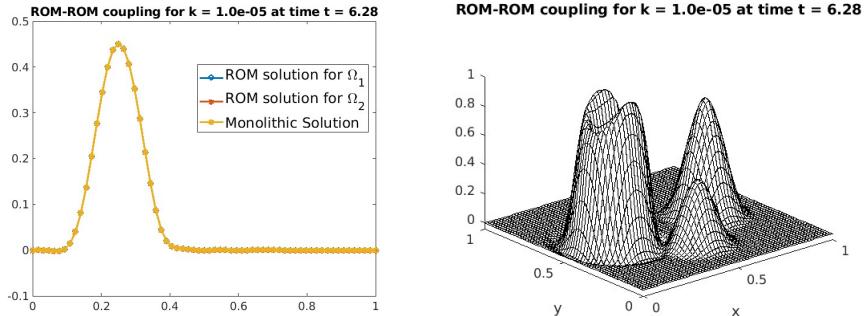
FIG. 3.2.  $u_m$  and  $u_c$  on the interface and  $u_c$  on the entire domain

FIG. 3.3. Monolithic FEM Solution on the entire domain and Initial Conditions

We next solve the ROM-ROM coupled system. The ROM basis for the problem has a maximum of 2016 modes in each subdomain. We use 1800 modes in each subdomain and use the same constants as the full order model. We compare the ROM-ROM solution,  $u_r$ , to the monolithic FEM solution,  $u_m$ , at time  $t = 2\pi$  and see that

$$\begin{aligned} \frac{\|u_r - u_m\|_{L^2(\Omega)}}{\|u_m\|_{L^2(\Omega)}} &= 4.8937 \cdot 10^{-10}, \\ \frac{\|u_r - u_m\|_{H^1(\Omega)}}{\|u_m\|_{H^1(\Omega)}} &= 1.1106 \cdot 10^{-9}. \end{aligned}$$

Hence, the ROM-ROM coupled solution accurately reproduces the monolithic FEM solution. This is also shown in the graph of  $u_r$  and  $u_m$  on interface, left of Figure 3.4, and the graph of  $u_r$  on the entire domain, right on Figure 3.4.

FIG. 3.4.  $u_m$  and  $u_r$  on the interface and  $u_r$  on the entire domain

Next, we compare the performance of ROM and the finite element FOM. For the FOM-FOM coupled system, we have that each time step takes approximately 25 gradient descent iterations and finding the solution at each time step takes about 8 seconds. For the ROM-ROM coupled system, we have that each time step needs approximately 40 gradient descent iterations and it takes about 10 seconds to find the solution. So ROM using 1800 modes is slightly slower than the FOM. The ROM-ROM coupled problem is slower in part due to the ROM matrices being dense. Ideally, less modes will be used thereby improving the computational time of each optimization iteration in the ROM-ROM coupled system, but this causes convergence issues that will be discussed later. There is another issue that is unrelated to density which is hindering the ROM-ROM coupled problem, but this will also be discussed later.

We use different amounts of modes and compare ROM-ROM gradient descent and the FOM-FOM gradient descent. To do this, we ran the FOM-FOM gradient descent and the ROM-ROM gradient descent with  $\delta = 10^{-7}$  for one time step. In Table (3.4), we see that when using less modes the ROM-ROM gradient descent needs an increasing amount of iterations to attain the same accuracy as the FOM-FOM gradient descent. If less than 1700 modes are used, the same accuracy cannot be reached. Instead, we see a stagnation behavior where the relative change between iterations is less than or equal to  $O(10^{-5})$ . This causes the accuracy to remain nearly constant for thousands of iterations. This sequence may converge eventually, but it is clear that the FOM-FOM gradient descent performed significantly better so allowing it to continue was not practical. When this happened, we noted the accuracy at which it stagnated and denoted the behavior by \* in the iteration column of the tables below.

We also solve the problem by using a ROM only for the state problem and using a FOM for the adjoint problem. For  $\delta = 10^{-7}$ , we find that a solution can be found with the same accuracy as the FOM-FOM gradient descent, without the stagnation behavior previously seen, even when widely varying the amount of modes for the ROM. However, as less modes are used it takes more iterations to achieve the same optimal value as the FOM-FOM coupled problem at that time step. This hints at there being an issue with using the current ROM for the adjoint problem.

Finally we consider the FOM-FOM gradient descent method and the ROM-ROM gradient descent method with different  $\delta$ . As expected the FOM-FOM coupled problem with smaller  $\delta$  achieves a smaller optimal value and consequently better accuracy while requiring

slightly more optimization iterations. For  $\delta = 10^{-9}$  and  $10^{-14}$ , the behavior of the ROM-ROM gradient descent methods is the same as for  $\delta = 10^{-7}$  as shown in Tables 3.2 and 3.3. We see that using 1800 modes results in more optimization iterations than the FOM-FOM gradient descent method in order to achieve the same accuracy as the FOM-FOM coupled problem. We also see an overall pattern that for each amount of modes, as  $\delta$  decreases, the ROM-ROM coupled problem takes slightly more iterations to achieve its desired optimal value. We also solve the ROM-ROM gradient descent method using the ROM state equation and the FOM adjoint equation using  $\delta = 10^{-9}$  and  $10^{-14}$ . For both  $\delta$  we see that the ROM-ROM gradient descent method converges with the same accuracy as the FOM-FOM coupled problem until 1000 modes are used. Using 1000 modes or less, the ROM-ROM gradient descent method using the ROM state equation and the FOM adjoint equation shows stagnating behavior. Otherwise, the gradient descent method using the ROM state equation and FOM adjoint equation for these  $\delta$  values behaves the same as with  $\delta = 10^{-7}$ .

Both the FOM-FOM coupled finite element solution and the ROM-ROM coupled finite element solution using the gradient descent algorithm accurately reproduce the monolithic finite element solution to the advection-diffusion equation. For  $\delta = 10^{-7}$  in the FOM-FOM coupled problem, we can attain an optimal value for the optimization problem of order  $O(10^{-10})$ . This is equivalent to solving for a  $g$  which gives  $\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{Gz}} = O(10^{-10})$ . This optimal value decreases as  $\delta$  decreases, as shown in Table 3.5, so  $\|u_1 - u_2\|_{\Gamma_0}$  can be made arbitrarily small by decreasing  $\delta$ . The ROM-ROM coupled problem needs 1800 out of the 2016 stored modes in order to attain the same optimal value in each optimization iteration as the FOM-FOM coupled problem. Using less modes in the state and adjoint solve further increases the speed of each optimization iteration as expected so that it is faster than the FOM-FOM coupled problem's optimization iterations. However, the adjoint solve in the ROM-ROM coupled problem cause it to not to be able to attain the same accuracy for less than 1800 modes. This issue with the adjoint in the ROM-ROM coupled problem is illustrated by the Tables 3.2, 3.3, and 3.4.

We repeat the same problem and change only the diffusivity constant to  $\nu = 10^{-3}$  and give the results in Table 3.6, Table 3.7, Table 3.8, and Table 3.9. In the FOM-FOM coupled problem we see the same accuracy as the previous results, except the optimization algorithm requires slightly fewer iterations to converge. We see that the ROM-ROM coupled problem for each  $\delta$  has a lower limit, below which the optimization algorithm exhibits stagnation behavior; this lower limit seems to be around 1000 modes. Compared to the problem with  $\nu = 10^{-5}$ , we are able to avoid stagnation while keeping fewer modes. Furthermore, when the FOM adjoint equation and ROM state equation is used for the ROM-ROM coupled problem, we see that less modes can be used without stagnation. Similar behavior is also seen with  $\nu = 10^{-5}$ .

These results point to a problem with using a ROM basis for the adjoint problem which is generated from snapshots of the state problem. Our intuition is that the adjoint variable is closely related to the flux of the state and not the state itself, hence snapshots of the state are not likely to capture the adjoint behavior well. To verify that the ROM generated from state problem snapshots is deficient for use in the adjoint, we check the projection error in each domain  $\Omega_i$  for a given adjoint solution  $\mu$  using

$$\mathcal{E}(\mu, \Psi_i) = \frac{\|\mu - \Psi_i(\Psi_i^T \Psi_i)^{-1} \Psi_i^T \mu\|_{L^2(\Omega_i)}}{\|\mu\|_{L^2(\Omega_i)}}.$$

We will only check the projection error using solutions to the first optimization iteration on the first time step, as the results should be similar for all optimization steps and for all time steps. For  $\nu = 10^{-5}$ , we first verify that using all 2016 modes results in machine precision. For  $\delta = 10^{-7}, 10^{-9}$ , and  $10^{-14}$  we get that

$$\mathcal{E}(\mu_1, \Psi_1) = O(10^{-15}) \text{ and } \mathcal{E}(\mu_2, \Psi_2) = O(10^{-15}).$$

Next we use 2015 modes with the same  $\delta$ 's and get

$$\mathcal{E}(\mu_1, \Psi_1) = O(10^{-3}) \text{ and } \mathcal{E}(\mu_2, \Psi_2) = O(10^{-3}),$$

This immediate increase in the projection error when using 2015 out of 2016 modes shows that the ROM basis does not capture the behavior of the adjoint well. Recall that this ROM basis is constructed using solutions to the monolithic FEM problem; in the frame of the coupled problem, this is equivalent to using solutions to the state problem in each domain.

We noted that projection error did not vary with respect to  $\delta$ , so we only provide results in Table 3.1 by varying  $\nu$ . We see that there is little improvement in the projection error as more modes are kept, with an abrupt increase when all modes are retained. At less than or equal to 1999 modes we have the projection error is  $O(10^{-1})$ .

Modes	$\mathcal{E}(\mu_1, \Psi_1)$	$\mathcal{E}(\mu_2, \Psi_2)$	Modes	$\mathcal{E}(\mu_1, \Psi_1)$	$\mathcal{E}(\mu_2, \Psi_2)$
10	$O(10^{-1})$	$O(10^{-1})$	10	$O(10^{-1})$	$O(10^{-1})$
2000	$O(10^{-1})$	$O(10^{-1})$	1999	$O(10^{-1})$	$O(10^{-1})$
2001	$O(10^{-2})$	$O(10^{-2})$	2000	$O(10^{-2})$	$O(10^{-2})$
2014	$O(10^{-2})$	$O(10^{-2})$	2014	$O(10^{-2})$	$O(10^{-2})$
2015	$O(10^{-3})$	$O(10^{-3})$	2015	$O(10^{-3})$	$O(10^{-3})$
2016	$O(10^{-15})$	$O(10^{-15})$	2016	$O(10^{-15})$	$O(10^{-15})$

TABLE 3.1  
Projection error in  $\Omega_1$  and  $\Omega_2$  for  $\nu = 10^{-5}$  (left) and  $\nu = 10^{-3}$  (right)

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
1000	$O(10^{-7})$	*	$O(10^{-9})$	*
1500	$O(10^{-8})$	*	$O(10^{-15})$	3230
1600	$O(10^{-8})$	*	$O(10^{-16})$	1000
1700	$O(10^{-16})$	3224	$O(10^{-16})$	227
1800	$O(10^{-16})$	44	$O(10^{-16})$	28
2016	$O(10^{-16})$	29	$O(10^{-16})$	30

TABLE 3.2  
 $\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints for  $\delta = 10^{-14}$  and  $\nu = 10^{-5}$

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
1000	$O(10^{-7})$	*	$O(10^{-9})$	*
1500	$O(10^{-8})$	*	$O(10^{-12})$	2029
1600	$O(10^{-8})$	*	$O(10^{-12})$	739
1700	$O(10^{-12})$	2011	$O(10^{-12})$	180
1800	$O(10^{-12})$	38	$O(10^{-12})$	27
2016	$O(10^{-12})$	29	$O(10^{-12})$	28

TABLE 3.3

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints for  $\delta = 10^{-9}$  and  $\nu = 10^{-5}$

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
250			$O(10^{-8})$	1327
500	$O(10^{-7})$	*	$O(10^{-10})$	1027
1000	$O(10^{-7})$	*	$O(10^{-10})$	1265
1500	$O(10^{-8})$	*	$O(10^{-10})$	1174
1600	$O(10^{-8})$	*	$O(10^{-10})$	479
1700	$O(10^{-10})$	1437	$O(10^{-10})$	136
1800	$O(10^{-10})$	38	$O(10^{-10})$	28
2016	$O(10^{-10})$	23	$O(10^{-10})$	24

TABLE 3.4

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints using  $\delta = 10^{-7}$  and  $\nu = 10^{-5}$

$\delta$	$\ u_1 - u_2\ _{\Gamma_0}$	Iterations
$10^{-7}$	$O(10^{-10})$	22
$10^{-9}$	$O(10^{-12})$	30
$10^{-14}$	$O(10^{-16})$	28

TABLE 3.5

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the FOM-FOM coupled problem using varying  $\delta$  and  $\nu = 10^{-5}$

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
50	$O(10^{-5})$	*	$O(10^{-6})$	*
100	$O(10^{-7})$	*	$O(10^{-9})$	*
250	$O(10^{-7})$	*	$O(10^{-11})$	*
500	$O(10^{-8})$	*	$O(10^{-15})$	3654
1000	$O(10^{-12})$	*	$O(10^{-16})$	473
1400	$O(10^{-15})$	1274	$O(10^{-16})$	142
1500	$O(10^{-16})$	609	$O(10^{-16})$	105
1600	$O(10^{-16})$	48	$O(10^{-16})$	31
2016	$O(10^{-16})$	23	$O(10^{-16})$	23

TABLE 3.6

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints for  $\delta = 10^{-14}$  and  $\nu = 10^{-3}$

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
50	$O(10^{-5})$	*	$O(10^{-6})$	*
250	$O(10^{-7})$	*	$O(10^{-11})$	4357
500	$O(10^{-8})$	*	$O(10^{-12})$	1565
1000	$O(10^{-12})$	$\approx 4100$	$O(10^{-12})$	313
1400	$O(10^{-12})$	767	$O(10^{-12})$	107
1500	$O(10^{-12})$	398	$O(10^{-12})$	79
1600	$O(10^{-12})$	41	$O(10^{-12})$	27
2016	$O(10^{-12})$	25	$O(10^{-12})$	24

TABLE 3.7

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints for  $\delta = 10^{-9}$  and  $\nu = 10^{-3}$

Modes	Error using ROM Adjoint	Iterations	Error using FOM Adjoint	Iterations
50	$O(10^{-5})$	*	$O(10^{-6})$	1032
100	$O(10^{-7})$	*	$O(10^{-8})$	1715
250	$O(10^{-7})$	*	$O(10^{-9})$	1422
500	$O(10^{-8})$	*	$O(10^{-10})$	485
1000	$O(10^{-10})$	$\approx 1600$	$O(10^{-10})$	189
1400	$O(10^{-10})$	524	$O(10^{-10})$	95
1500	$O(10^{-10})$	289	$O(10^{-10})$	71
1600	$O(10^{-10})$	40	$O(10^{-10})$	23
2016	$O(10^{-10})$	24	$O(10^{-10})$	23

TABLE 3.8

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for ROM-ROM coupled problem using ROM adjoints and the ROM-ROM coupled problem using FOM adjoints using  $\delta = 10^{-7}$  and  $\nu = 10^{-3}$

$\delta$	$\ u_1 - u_2\ _{\Gamma_0}$	Iterations
$10^{-7}$	$O(10^{-10})$	25
$10^{-9}$	$O(10^{-12})$	25
$10^{-14}$	$O(10^{-16})$	22

TABLE 3.9

$\frac{\|u_1 - u_2\|_{\Gamma_0}}{\|u_1\|_{\Gamma_0}}$  for the FOM-FOM coupled problem using varying  $\delta$  and  $\nu = 10^{-3}$

**4. Conclusion.** We recast the FOM-FOM PDE-constrained optimization problem introduced in [3], adapted to advection-diffusion, for ROM-ROM coupled problems in order to investigate computational efficiency gains and other potential improvements to the approach. We extended an in-house MATLAB code to implement Algorithms 2 and 3 and demonstrated that this was done correctly with a numerical patch test for a manufactured solution. We also computed the monolithic FEM solution of the fully-coupled problem and compared it to our approach, investigating relationships between choice of  $\delta$  values, amount of diffusivity ( $\nu$ ), number of retained modes for the ROM bases, and use of ROM bases generated from solutions to the primal problem. We noted that snapshots of the primal problem do not seem to produce a quality reduced basis for the adjoint, which we verified by calculating projection errors of the FOM adjoint solution against its projection onto the reduced basis. We also compared the ROM-ROM coupling with a ROM adjoint against a

ROM-ROM coupling with a FOM adjoint that did not share similar requirements for an unreasonably large number of retained reduced basis modes. In future work, we will consider different snapshots variables from which to generate a basis for the adjoint problem.

## REFERENCES

- [1] S. BERRONE, D. GRAPPEIN, AND S. SCIALO, *A PDE-constrained optimization method for 3D-1D coupled problems with discontinuous solutions*, Numer Algor., (2023).
- [2] A. N. BROOKS AND T. HUGHES, *Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations*, Comp. Meth. Appl. Mech. Engng., 32 (1982), pp. 199–259.
- [3] A. CASTRO, P. BOCHEV, P. KUBERRY, AND I. TEZAUR, *Explicit synchronous partitioned scheme for coupled reduced order models based on composite reduced bases*, arXiv, (2023).
- [4] S. S. COLLIS AND M. HEINKENSCHLOSS, *Analysis of the streamline upwind/Petrov Galerkin method applied to the solution of optimal control problems*, CAAM TR02-01, (2002).
- [5] 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*, Earth and Space 2022, (2023), pp. 475–489.
- [6] M. D’ELIA AND P. BOCHEV, *Formulation, analysis and computation of an optimization-based local-to-nonlocal coupling method*, Results Appl. Math., (2020).
- [7] M. GUNZBURGER AND H. LEE, *An optimization-based domain decomposition method for the Navier–Stokes equations*, SIAM J. Numer. Analysis, 37 (2000), p. 1455–1480.
- [8] M. GUNZBURGER, J. PETERSON, AND H. KWON, *An optimization-based domain decomposition method for partial differential equations*, Computers and Math. with Applications, 37 (1999), p. 77–93.
- [9] M. GUNZBURGER, J. PETERSON, AND J. SHADID, *Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data*, Comp. Meth. Appl. Mech. Engng., 196 (2007), pp. 1030–1047.
- [10] M. D. GUNZBURGER, *Perspectives in Flow Control and Optimization*, SIAM, 2003.
- [11] P. HOLMES, J. LUMLEY, AND G. BERKOOZ, *Turbulence, coherent structures, dynamical systems and symmetry*, Cambridge University Press, (1996).
- [12] A. IOLLO, G. SAMBATARO, AND T. TADDEI, *A one-shot overlapping Schwarz method for component-based model reduction: application to nonlinear elasticity*, Comp. Meth. Appl. Mech. Engng., 404 (2023).
- [13] P. KUBERRY, P. BOCHEV, AND K. PETERSON, *An optimization-based approach for elliptic problems with interfaces*, SIAM Journal on Scientific Computing, 5 (2017).
- [14] P. KUBERRY AND H. LEE, *A decoupling algorithm for fluid-structure interaction problems based on optimization*, Comput. Methods Appl. Mech. Engrg., (2013), pp. 594–605.
- [15] L. SIROVICH, *Turbulence and the dynamics of coherent structures, part iii: dynamics and scaling*, Q. Appl. Math, 45 (1987), p. 583–590.

## RANDOMIZED HOUSEHOLDER-CHOLESKY QR FACTORIZATION WITH MULTI-SKETCHING

ANDREW J. HIGGINS\*, ERIK G. BOMAN†, DANIEL B. SZYLD\*, AND ICHITARO YAMAZAKI†

**Abstract.** CholeskyQR2 and shifted CholeskyQR3 are two state-of-the-art algorithms for computing tall-and-skinny QR factorizations since they obtain high performance on current computer architectures. However, in some applications CholeskyQR2 faces a prohibitive restriction on the condition number of the underlying matrix to factorize, while shifted CholeskyQR3 is stable but has  $1.5 \times$  higher computational and communication costs than CholeskyQR2. Here, we investigate a randomized QR algorithm called Randomized Householder-Cholesky (`rand_cholQR`). Using one or two random sketch matrices, we show experimentally that its orthogonality error is bounded by unit roundoff for any numerically full-rank matrix with high probability, and hence it is at least as stable as shifted CholeskyQR3 in practice. We also show the performance of `rand_cholQR` on a NVIDIA A100 GPU, demonstrating that for tall-and-skinny matrices, `rand_cholQR` with multiple sketch matrices is nearly as fast as, or in some cases faster than, CholeskyQR2. Hence, compared to CholeskyQR2, `rand_cholQR` is more stable with almost no extra computational or memory cost, and is therefore a superior algorithm.

**1. Introduction.** Computing the QR factorization of tall-and-skinny matrices is a critical component of many scientific and engineering applications, including the solution of least squares problems, block orthogonalization kernels for solving linear systems and eigenvalue problems within block or  $s$ -step Krylov methods, dimensionality reduction methods for data analysis like Principal Component Analysis, and many others. Modern high-performance machines enable extremely fast floating point operations (FLOPs), but are limited by relatively slow communication (data transfers) between processors and through the memory hierarchy. Current popular QR algorithms for tall-and-skinny matrices are the CholeskyQR2 and shifted CholeskyQR3 algorithms, due to their communication-avoiding properties along with their exploitation of vendor-provided highly-optimized dense linear algebra subroutines [7, 8]. However, CholeskyQR2 may fail to factorize a matrix  $V$  when  $\kappa(V) < O(\mathbf{u}^{-1/2})$  where  $\kappa(V)$  is the condition number of  $V$  and  $\mathbf{u}$  is unit roundoff [18]. While shifted CholeskyQR3 is numerically stable as long as  $\kappa(V) < O(\mathbf{u}^{-1})$ , it requires over 50% more computational and communication cost than CholeskyQR2 [7]. Although more stable communication-avoiding algorithms exist, such as TSQR, they rely on Householder QR factorizations, and are often significantly slower than CholeskyQR2 in practice [5, 8].

In this paper, we propose a randomized algorithm called `rand_cholQR` for computing the QR factorization of a tall-and-skinny matrix  $V$  using one or more random sketch matrices. We perform a brief numerical study indicating that its orthogonality error is bounded on the order of unit roundoff for any numerically full-rank matrix  $V$  with high probability, and hence it is at least as stable as shifted CholeskyQR3 and is significantly more numerically stable than CholeskyQR2 in practice. In addition, the `rand_cholQR` algorithm may be implemented using the same basic linear algebra kernels as CholeskyQR2. Hence, it is simple to implement and has the same communication-avoiding properties thereby making it a comparable algorithm in terms of performance on current computer architectures, especially when two random sketch matrices (“Multi-Sketch”) are used. To support this, we include a computational study on a state-of-the-art NVIDIA GPU to demonstrate that `rand_cholQR` with a Multi-Sketch strategy can perform up to 4% faster than CholeskyQR2 and 56.6% faster than shifted CholeskyQR3, while significantly improving the robustness of CholeskyQR2.

---

\*Temple University, Philadelphia, Pennsylvania, USA ([andrew.higgins@temple.edu](mailto:andrew.higgins@temple.edu), [szyld@temple.edu](mailto:szyld@temple.edu)).

†Center for Computing Research, Sandia National Laboratories, Albuquerque, New Mexico, USA ([egboman@sandia.gov](mailto:egboman@sandia.gov), [iyamaza@sandia.gov](mailto:iyamaza@sandia.gov)).

## 2. Preliminaries.

**2.1. Random Sketching.** Suppose one would like to compress  $V \in \mathbb{R}^{n,m}$  into a smaller matrix with nearly the same singular values. We denote the *sketch matrix* by  $S \in \mathbb{R}^{s,n}$  for  $s \ll n$ . The sketch matrix is typically chosen to be a  $\epsilon$ -*subspace embedding*, or a linear map to a lower dimensional space that preserves  $\ell_2$ -inner products and norms of all vectors within the subspace up to a factor of  $\sqrt{1 \pm \epsilon}$  for  $\epsilon \in [0, 1)$  [3, 10, 16].

**DEFINITION 2.1** ( $\epsilon$ -subspace embedding). *Given  $\epsilon \in [0, 1)$ , the sketch matrix  $S \in \mathbb{R}^{s,n}$  is an  $\epsilon$ -subspace embedding for the subspace  $\mathcal{V} \subset \mathbb{R}^n$  if  $\forall x, y \in \mathcal{V}$ ,*

$$|\langle x, y \rangle - \langle Sx, Sy \rangle| \leq \epsilon \|x\|_2 \|y\|_2. \quad (2.1)$$

**COROLLARY 2.1.1.** *If the sketch matrix  $S \in \mathbb{R}^{s,n}$  is an  $\epsilon$ -subspace embedding for the subspace  $\mathcal{V} \subset \mathbb{R}^n$ , then  $\forall x \in \mathcal{V}$ ,*

$$\sqrt{1 - \epsilon} \|x\|_2 \leq \|Sx\|_2 \leq \sqrt{1 + \epsilon} \|x\|_2. \quad (2.2)$$

**COROLLARY 2.1.2.** *If the sketch matrix  $S \in \mathbb{R}^{s,n}$  is an  $\epsilon$ -subspace embedding for the subspace  $\mathcal{V} \subset \mathbb{R}^n$ , and  $V$  is a matrix whose columns form a basis of  $\mathcal{V}$ , then*

$$\begin{aligned} (1 + \epsilon)^{-1/2} \sigma_{\min}(SV) &\leq \sigma_{\min}(V) \leq \sigma_{\max}(V) \\ &\leq (1 - \epsilon)^{-1/2} \sigma_{\max}(SV). \end{aligned} \quad (2.3)$$

Thus,

$$\kappa(V) \leq \sqrt{\frac{1 - \epsilon}{1 + \epsilon}} \kappa(SV). \quad (2.4)$$

The proof for Corollary 2.1.1 is trivial, and the proof for Corollary 2.1.2 is given in [3]. Corollary 2.1.2 implies that the singular values of any full-rank  $V \in \mathbb{R}^{n,m}$  are bounded by those of  $SV$ . Hence if  $SV$  is well conditioned, then so is  $V$ .

While  $\epsilon$ -subspace embeddings require knowledge of the subspace  $\mathcal{V} \subset \mathbb{R}^n$  a priori,  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embeddings do not [3].

**DEFINITION 2.2** (( $\epsilon, d, m$ ) oblivious  $\ell_2$ -subspace embedding).  *$S \in \mathbb{R}^{s,n}$  is an  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embedding if it is an  $\epsilon$ -subspace embedding for any fixed  $m$ -dimensional subspace  $\mathcal{V} \subset \mathbb{R}^n$  with probability at least  $1 - d$ .*

an example of an  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embedding is  $S = \frac{1}{\sqrt{s}} G$  for a Gaussian matrix  $G \in \mathbb{R}^{s,n}$  and  $s = \Omega(\epsilon^{-2} \log m \log(1/d))$  [16]. Sparse  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embeddings exist, including CountSketch, which consists of a single  $\pm 1$  per column, where the row storing the entry and its sign are chosen uniformly at random [4, 17]. In order to be an  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embedding, the CountSketch matrix must satisfy  $s \geq \frac{m^2 + m}{\epsilon^2 d}$  [9]. Other popular  $(\epsilon, d, m)$  oblivious  $\ell_2$ -subspace embeddings include sub-sampled randomized Hadamard and Fourier transforms, and “sparse dimension reduction maps” [3, 10]. However, obtaining high performance with these is difficult, and the complexity of applying them is higher than CountSketch.

Next consider the case of applying two sketch matrices. This can be a computationally beneficial strategy to reducing the size of a matrix  $V \in \mathbb{R}^{n,m}$ , as discussed in more detail in Section 2.3.

**COROLLARY 2.1.3.** *Let  $S_1 \in \mathbb{R}^{s_1,n}$  be an  $(\epsilon_1, d_1, m)$  oblivious  $\ell_2$ -subspace embedding in  $\mathbb{R}^n$ ,  $S_2 \in \mathbb{R}^{s_2,s_1}$  be an  $(\epsilon_2, d_2, m)$  oblivious  $\ell_2$ -subspace embedding in  $\mathbb{R}^{s_1}$ , generated independently. Then for any  $m$ -dimensional subspace  $\mathcal{V} \subset \mathbb{R}^n$  and  $\forall x \in \mathcal{V}$ ,*

$$\sqrt{1 - \epsilon_L} \|x\|_2 \leq \|S_2 S_1 x\|_2 \leq \sqrt{1 + \epsilon_H} \|x\|_2, \quad (2.5)$$

with probability at least  $1 - d$ , where  $\varepsilon_L = \varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2$ ,  $\varepsilon_H = \varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2$ , and  $d = d_1 + d_2 - d_1d_2$ .

*Proof.*  $S_2$  is an  $(\varepsilon_2, d, m)$  oblivious  $\ell_2$ -subspace embedding of  $S_1V \in \mathbb{R}^{s_1}$ , and that if  $x \in \mathcal{V}$ , then  $S_1x \in S_1\mathcal{V}$ . Therefore, for any  $x \in \mathcal{V}$ ,

$$\begin{aligned} \sqrt{1 - (\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2)}\|x\|_2 &= \sqrt{(1 - \varepsilon_2)(1 - \varepsilon_1)}\|x\|_2 \leq \sqrt{1 - \varepsilon_2}\|S_1x\|_2 \\ &\leq \|S_2S_1x\|_2 \leq \sqrt{1 + \varepsilon_2}\|S_1x\|_2 \\ &\leq \sqrt{(1 + \varepsilon_2)(1 + \varepsilon_1)}\|x\|_2 = \sqrt{1 + (\varepsilon_1 + \varepsilon_2 + \varepsilon_1\varepsilon_2)}\|x\|_2 \end{aligned}$$

with probability at least  $(1 - d_1)(1 - d_2) = 1 - (d_1 + d_2 - d_1d_2)$ .  $\square$

If  $S_1, S_2$  are  $\varepsilon_1, \varepsilon_2$  embeddings respectively, then by Corollaries 2.1.2–2.1.3,

$$\begin{aligned} (1 + \varepsilon_H)^{-1/2} \sigma_{min}(S_2S_1V) &\leq \sigma_{min}(V) \leq \sigma_{max}(V) \\ &\leq (1 - \varepsilon_L)^{-1/2} \sigma_{max}(S_2S_1V), \end{aligned} \tag{2.6}$$

and so,

$$\kappa(V) \leq \sqrt{\frac{1 - \varepsilon_L}{1 + \varepsilon_H}} \kappa(S_2S_1V). \tag{2.7}$$

**2.2. Description and Motivation of `randQR` and `rand_cholQR`.** Let  $V \in \mathbb{R}^{n,m}$ , and suppose  $S_1 \in \mathbb{R}^{s_1,n}$  and  $S_2 \in \mathbb{R}^{s_2,s_1}$  are  $(\varepsilon_1, d_1, m)$  and  $(\varepsilon_2, d_2, m)$  oblivious  $\ell_2$ -subspace embeddings, respectively. We define the Randomized Householder QR algorithm (`randQR`) in Algorithm 1. We provide motivation for using two sketches in Section 2.3.

---

**Algorithm 1** Randomized Householder QR:  $[Q, R] = \text{randQR}(V, S_1, S_2)$ 


---

**Input:** Matrix  $V \in \mathbb{R}^{n,m}$ , sketch matrices  $S_1 \in \mathbb{R}^{s_1,n}$ ,  $S_2 \in \mathbb{R}^{s_2,s_1}$

**Output:**  $S_2S_1$ -Orthogonal factor  $Q \in \mathbb{R}^{n,m}$ , Triangular factor  $R \in \mathbb{R}^{m,m}$

- 1: Apply sketches  $W = S_2S_1V$
  - 2: Perform Householder QR:  $[Q_{tmp}, R] = \text{hhqr}(W)$
  - 3: Recover  $S_2S_1$ -orthogonal matrix:  $Q = VR^{-1}$
- 

In exact arithmetic, provided that  $V \in \mathbb{R}^{n,m}$  is full rank, then `randQR` produces a matrix  $Q$  that is  $S_2S_1$ -orthogonal; i.e., it is orthogonal with respect to  $(S_2S_1, S_2S_1)$ , or  $(S_2S_1Q)^T(S_2S_1Q) = I$ , as  $S_2S_1Q = S_2S_1VR^{-1} = WR^{-1} = Q_{tmp}$ , where  $Q_{tmp}$  is the orthogonal factor produced by the Householder QR factorization of  $W = S_2S_1V$ . Unlike traditional Householder QR, even in exact arithmetic  $V$  must have full rank, since step 3 of algorithm 1 requires  $\text{rank}(V) = \text{rank}(R) = m$ . In finite precision, intuition suggests that an inevitable requirement of `randQR` is that  $V$  must be numerically full rank.

Let  $Q$  be the  $S_2S_1$ -orthogonal factor from `randQR`. By Corollary 2.1.2 and the fact that  $S_2S_1Q$  is orthogonal,  $\kappa(Q) \leq \sqrt{\frac{1 - \varepsilon_L}{1 + \varepsilon_H}} \kappa(S_2S_1Q) = \sqrt{\frac{1 - \varepsilon_L}{1 + \varepsilon_H}} = O(1)$ . Thus, if one re-orthogonalizes  $Q$  using Cholesky QR (`cholQR`, Algorithm 2), it can be shown that the result has a loss of orthogonality on the order of machine precision [18]. We call this `rand_cholQR`, provided in Algorithm 3.

The computational cost of step 2 of `randQR` (Algorithm 1) is negligible compared to steps 1 and 3, since  $W \in \mathbb{R}^{s_2,m}$  with  $s_2 \ll n$ . The cost of step 1 is dependent on the type of sketch matrices used.

For simplicity, suppose one replaces  $S_2S_1$  with a single dense Gaussian sketch matrix  $S \in \mathbb{R}^{s,n}$ . This is conceptually straightforward, very efficient in parallel, but computationally

**Algorithm 2** Cholesky QR:  $[Q, R] = \text{cholQR}(V)$ 


---

**Input:** Matrix  $V \in \mathbb{R}^{n,m}$   
**Output:** Orthogonal factor  $Q \in \mathbb{R}^{n,m}$ , Triangular factor  $R \in \mathbb{R}^{m,m}$  such that  $QR = V$ .

- 1: Compute Gram matrix  $G = V^T V$
- 2: Perform Cholesky on  $G$ :  $R = \text{chol}(G)$
- 3: Recover orthogonal matrix:  $Q = VR^{-1}$

---

**Algorithm 3** Rand. Householder-Cholesky:  $[Q, R] = \text{rand\_cholQR}(V, S_1, S_2)$ 


---

**Input:** Matrix  $V \in \mathbb{R}^{n,m}$ , sketch matrices  $S_1 \in \mathbb{R}^{s_1, n}$ ,  $S_2 \in \mathbb{R}^{s_2, s_1}$   
**Output:** Orthogonal factor  $Q \in \mathbb{R}^{n,m}$ , Triangular factor  $R \in \mathbb{R}^{m,m}$  such that  $QR = V$ .

- 1: Recover  $S_2 S_1$ -orthogonal matrix  $Q_0$ :  $[Q_0, R_0] = \text{randQR}(V, S_1, S_2)$
- 2: Perform Cholesky QR on  $Q_0$ :  $[Q, R_1] = \text{cholQR}(Q_0)$
- 3: Return  $R$ :  $R = R_1 R_0$

---

expensive and memory inefficient since  $S$  is fully dense. In this case, the computational cost of `randQR` and `rand_cholQR` (in FLOPs) are:

$$\text{randQR FLOPs: } \underbrace{sm(2n - 1)}_{\text{Sketching}} + \underbrace{2sm^2 - \frac{2}{3}m^3}_{\text{Householder QR}} + \underbrace{nm^2}_{\text{Tri. solve}} \approx 2nms + nm^2, \quad (2.8)$$

$$\text{rand\_cholQR FLOPs: } \underbrace{2nms + nm^2}_{\text{randQR}} + \underbrace{2nm^2}_{\text{cholQR}} + \underbrace{m^2(2m - 1)}_{\text{Matrix mult.}} \approx 2nms + 3nm^2. \quad (2.9)$$

Provided that  $s = O(m)$ , e.g.,  $s \approx 2m$ , then `rand_cholQR` FLOPs  $\approx 7nm^2$ . In contrast, performing `cholQR` twice (`cholQR2`, Algorithm 4) incurs a cost of

$$\text{cholQR2 FLOPs: } \underbrace{2nm^2}_{\text{cholQR}} + \underbrace{2nm^2}_{\text{cholQR}} + \underbrace{m^2(2m - 1)}_{\text{Matrix mult.}} \approx 4nm^2. \quad (2.10)$$

**Algorithm 4** CholeskyQR2:  $[Q, R] = \text{cholQR2}(V)$ 


---

**Input:** Full rank matrix  $V \in \mathbb{R}^{n,m}$   
**Output:** Orthogonal factor  $Q \in \mathbb{R}^{n,m}$ , Triangular factor  $R \in \mathbb{R}^{m,m}$

- 1: Perform Cholesky QR on  $W$ :  $[Q_0, R_0] = \text{cholQR}(V)$
- 2: Perform Cholesky QR on  $Q_0$ :  $[Q, R_1] = \text{cholQR}(Q_0)$
- 3: Return  $R$ :  $R = R_1 R_0$

---

The computational costs of `randQR` (using a dense Gaussian sketch) and `cholQR` are about the same asymptotically. In addition, the dominant costs incurred in steps 1 and 3 of `randQR` are nearly identical to `cholQR`, in the sense that both perform a product of tall and skinny matrices, followed by a triangular solve of a tall and skinny matrix, and therefore the algorithms should incur the same number of processor synchronizations. Moreover, `rand_cholQR` and `cholQR2` simply build on these algorithms, adding passes of `cholQR` to matrices of the same size for both algorithms. Thus, in a large scale parallel setting, one can expect `rand_cholQR` to run slightly slower, but on the same order of runtime, as `cholQR2`, and scale in the same way. However, as we show in Section 3.2, `rand_cholQR` is significantly more stable.

**2.3. Motivation for Multi-Sketching.** The Gaussian sketch framework described in Section 2.2 performs a dense matrix-matrix multiply with the sketch matrix  $S$  of dimension  $s \times n$ . Hence, we need to store and load the  $s \times n$  sketch matrix. As shown in Section 2.2, the time to sketch the matrix becomes dominant in the total `randQR` factorization time both because of the FLOPs incurred and because memory access is expensive on a modern computer architecture like a GPU.

One can reduce the sketching cost using a CountSketch matrix [4]. Since the CountSketch matrix has only one non-zero per column, the cost of applying a CountSketch matrix to  $V \in \mathbb{R}^{n,m}$  is only  $O(nm)$ , and it only needs to store  $O(n)$  numerical values. Additionally, CountSketch can be implemented using the sparse-matrix multiple-vector multiply (SpMM), whose optimized implementation is often available on a specific architecture. A clever implementation can exploit the fact that applying the CountSketch matrix is equivalent to adding/subtracting subsets of rows of  $V$ , and can therefore be parallelized well using batched BLAS-1 kernels or a highly-optimized sparse linear algebra library. Hence, CountSketch could obtain high performance using only readily available linear algebra libraries. However, a CountSketch matrix requires sketch size  $s = O(m^2)$  to maintain the  $\varepsilon$ -embedding properties, so one is left to factorize  $W \in \mathbb{R}^{s,m}$  with Householder QR, which incurs  $O(m^4)$  FLOPs. In contrast, the Gaussian sketch ensures that  $S$  is an  $\varepsilon$ -subspace embedding with  $s = O(m)$ , meaning the cost of the Householder QR factorization is only  $O(m^3)$  FLOPs. Householder QR imposes high communication costs and does not parallelize well [5]. As a result, on current computers, it obtains much lower performance than BLAS-3 operations like the dense matrix product (`gemm`), and these  $O(m^4)$  FLOPs for Householder QR become a performance bottleneck.

Ideally, we want an embedding that offers low computational and storage costs like CountSketch, while returning a  $W \in \mathbb{R}^{s,m}$  with  $s = O(m)$  like the Gaussian sketch does to avoid a performance bottleneck from Householder QR. This is possible by using a “Multi-Sketch” framework with first a CountSketch and then a Gaussian sketch. To see this, suppose  $S_1 \in \mathbb{R}^{s_1,n}$  is a CountSketch matrix with  $s_1 = \frac{m^2+m}{\varepsilon_1^2 d_1}$ , and suppose  $S_2 \in \mathbb{R}^{s_2,s_1}$  is a Gaussian sketch where  $s_2 = 2m$ . By Corollary 2.1.3, for any  $V \in \mathbb{R}^{n,m}$ , with probability at least  $1 - d$

$$\sqrt{1 - \varepsilon_L} \|V\|_2 \leq \|S_2 S_1 V\|_2 \leq \sqrt{1 + \varepsilon_H} \|V\|_2.$$

We split the computation of  $W = S_2 S_1 V$  into two steps: first computing  $W_1 = S_1 V$ , then  $W = S_2 W_1$ . Storing  $S_1$  only requires  $O(n)$  bytes of memory, and the sparse matrix product  $W_1 = S_1 V$  costs  $O(nm)$  FLOPs. The cost to compute  $W = S_2 W_1$  costs  $O(m^4)$  FLOPs, but since the dense matrix product (`gemm`) obtains much higher performance than the Householder QR, this cost became negligible in our performance studies with a GPU. The storage of  $S_2$  only requires  $O(m^3)$  bytes of memory, and the Householder QR factorization of the  $O(m) \times m$  matrix  $W$  incurs negligible computational cost as well.

The  $O(nm + m^4)$  total FLOPs incurred using the Multi-Sketch framework can actually be lower than the  $O(nm^2)$  FLOPs required to perform `cholQR`, making `rand_cholQR` sometimes cheaper than `cholQR2` under the Multi-Sketch framework. Thus, the Multi-Sketch framework provides an avenue for an extremely efficient, stable QR factorization that can potentially outperform `cholQR2` in terms of both stability and practical speed on modern parallel machines.

**3. Numerical Experiments.** We conducted numerical experiments with two concerns in mind. First, we compare the performance of `rand_cholQR` with the performance of `cholQR2`, shifted CholeskyQR3 (`sCholQR3`), and Householder QR on a state-of-the-art GPU

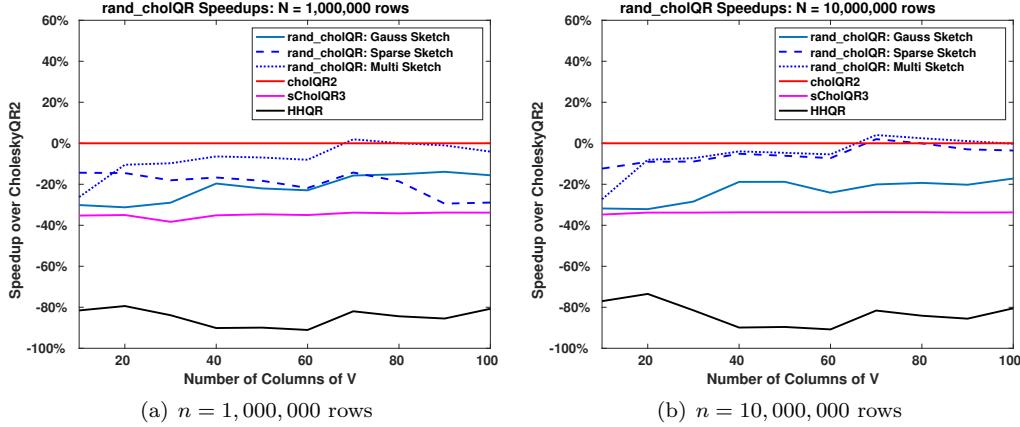


FIG. 3.1. *Performance of QR factorizations of  $V$  with  $\kappa(V) = 10^6$  for a fixed number of rows as the number of columns vary. Performance is relative to  $\text{cholQR2}$ , where we show the “Speedup over CholeskyQR2”, which is  $(\text{cholQR2 runtime}/\text{algorithm runtime} - 1) \times 100\%$ . Negative speedup indicates a slowdown.*

leveraging vendor-optimized libraries. Second, we compare the stability of `rand_cholQR` to the stability of `cholQR2`, `sCholQR3`, and Householder QR.

**3.1. Implementation Details.** We implemented `rand_cholQR`, `cholQR2`, `sCholQR3`, and Householder QR in C++. To be portable to a GPU, we used the Kokkos Performance Portability Library [6] and Kokkos Kernels [15]. For our experiments on an NVIDIA GPU, we configured and built our code such that Kokkos Kernels calls NVIDIA’s cuBLAS and cuSPARSE linear algebra libraries for optimized dense and sparse basic linear algebra routines [11, 14]. To perform LAPACK routines that are not currently available natively within Kokkos Kernels (i.e., `geqrf` and `orgqr` for computing the Householder QR factorization, and `potrf` for the Cholesky factorization), we directly called NVIDIA’s cuSOLVER linear algebra library [2, 12, 13]. Test results were obtained using Kokkos 3.7.01, Cuda 11.7.99, and GCC 7.2.0 on an AMD EPYC 7742 64-Core 2.25GHz CPU with a NVIDIA A100-SXM4 40GB GPU in double precision, so  $\mathbf{u} = 2^{-52} \approx 10^{-16}$ . Performance tests for each method were repeated 100 times and all reported runtimes are the average of these tests.

We tested a variety of sketching strategies. The simplest was the case of a Gaussian sketch  $S = \frac{1}{\sqrt{s}}G \in \mathbb{R}^{s,n}$ , whose entries were generated within a parallel for loop. The sketch size chosen for a Gaussian to embed  $V \in \mathbb{R}^{n,m}$  was  $s = \lceil 74.3 \log(m) \rceil$ , which can be shown to produce a  $(0.49, 1/m, m)$  oblivious  $\ell_2$ -subspace embedding [1, Lemma 4.1]. To test using CountSketch, we explicitly constructed a sparse matrix and applied the sketch using a sparse-matrix vector product. The sketch size used to embed  $V \in \mathbb{R}^{n,m}$  with a  $S \in \mathbb{R}^{s,n}$  CountSketch matrix was  $s = \lceil 8.24(m^2 + m) \rceil$ . This can be shown to be a  $(0.9, 0.15, m)$  oblivious  $\ell_2$ -subspace embedding [9, Theorem 1].

Our implementation of Multi-Sketching chose  $S_1 \in \mathbb{R}^{s_1, m}$  as a CountSketch described above with  $\varepsilon_1 = 0.9$ , and  $S_2 \in \mathbb{R}^{s_2, s_1}$  a Gaussian sketch with  $s_2 = \lceil 74.3 \log(s_1) \rceil$  giving  $\varepsilon_2 = 0.49$ . Thus,  $S_2 S_1$  produced an embedding with  $\varepsilon_L \approx 0.9490$ ,  $\varepsilon_H \approx 1.8310$ , and  $d \approx 0.15$ . Runtimes of `rand_cholQR` did not include the time to generate the sketch, as this was assumed to be a fixed overhead, though this cost is negligible for the Multi-Sketch and CountSketch frameworks.

Average Speedup over cholQR2		
	1,000,000 rows	10,000,000 rows
rand_cholQR: Gauss Sketch	-21.5%	-23.1%
rand_cholQR: CountSketch	-19.5%	-5.3%
rand_cholQR: Multi-Sketch	-7.1%	-4.9%
sCholQR3	-35.9%	-33.8%
Householder	-84.8%	-83.4%

TABLE 3.1

Average speedups over cholQR2, taken from experiments shown in Figure 3.1. Negative speedup indicates a slowdown.

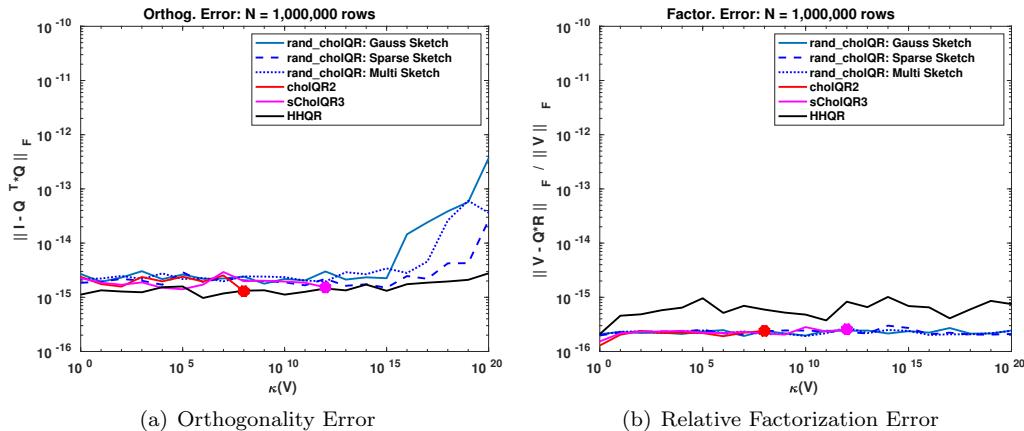


FIG. 3.2. Orthogonality (left) and relative factorization error (right) of the QR factorization of a matrix  $V$  with varying condition number. To explicitly control  $\kappa(V)$ ,  $V := L\Sigma R^T \in \mathbb{R}^{n,m}$  using random orthogonal matrices  $L, R$ , and a diagonal  $\Sigma$  with log-equispaced entries in the range  $[\kappa^{-1/2}(V), \kappa^{1/2}(V)]$ . Indicated by a large dot, lines for cholQR2 and sCholQR3 end at  $\kappa(V) = 10^8$  and  $\kappa(V) = 10^{12}$  respectively, as the methods fail beyond these points.

**3.2. Performance & Numerical Results.** Figure 3.1 and Table 3.1 show the relative performance of each QR method compared to cholQR2 for test problems with  $n = 10^6$  and  $n = 10^7$  rows, and  $m = 10\text{--}100$  columns. The results indicate that the Multi-Sketch strategy causes the least slowdown of rand\_cholQR relative to cholQR2, averaging only a 4.9–7.1% slowdown. In some cases, the Multi-Sketch rand\_cholQR actually outperforms cholQR2. In particular, for  $n = 10^7$  rows and  $m = 70$  columns, Multi-Sketch rand\_cholQR achieves a 4% speedup over cholQR2. In all cases, rand\_cholQR using the Multi-Sketch strategy is significantly faster than sCholQR3.

Figure 3.2 shows the orthogonality error  $\|I - \hat{Q}^T \hat{Q}\|_F$  and the relative factorization error  $\|V - \hat{Q}\hat{R}\|_F / \|V\|_F$  for condition number  $\kappa(V) \in [1, 10^{20}]$ . The results demonstrate that rand\_cholQR maintains  $O(\mathbf{u})$  orthogonality error and  $O(\mathbf{u})\|V\|_2$  factorization error<sup>1</sup> while  $\kappa(V) < O(\mathbf{u}^{-1})$ , and is more robust than cholQR2 and sCholQR3. In practice, it appears that rand\_cholQR is stable even when  $V$  is numerically rank-deficient. Additionally, Figures 3.1 and 3.2 demonstrate that rand\_cholQR with a Multi-Sketch framework significantly improves the robustness of cholQR2 and sCholQR3 at little to no cost, therefore making rand\_cholQR a superior high-performance QR algorithm.

<sup>1</sup>This follows because  $\|V - \hat{Q}\hat{R}\|_F / \|V\|_F = O(\mathbf{u})$  and  $\|V\|_2 \leq \|V\|_F \leq \sqrt{m}\|V\|_2$ .

**4. Conclusions.** Section 3.2 indicates that `rand_cholQR` using one or two sketch matrices orthogonalizes any numerically full-rank matrix  $V$  up to  $O(\mathbf{u})$  error, which is a significant improvement over CholeskyQR2, which requires  $\kappa(V) < O(\mathbf{u}^{-1/2})$ . Our performance results in Section 3.2 indicate that the significantly better stability properties of `rand_cholQR` over `cholQR2` come at virtually no increase in the factorization time on a modern GPU. Additionally, `rand_cholQR` is in practice more stable than `sCholQR3`, while being substantially faster. This is due to the fact that `rand_cholQR` and `cholQR2` incur the same number of processor synchronizations, while leveraging mostly BLAS-3 or optimized sparse matrix-vector routines for most of the required computation. In fact, `rand_cholQR` can perform better than `cholQR2` when using the Multi-Sketch framework. Of the sketching strategies considered, the Multi-Sketch framework is the most advantageous, likely because it requires little additional storage compared to `cholQR2`, and applying the sketches in this framework is extremely cheap.

A rigorous roundoff error analysis for `randQR` and `rand_cholQR` under both a single sketch and Multi-Sketch strategy is forthcoming. Future work includes applying `rand_cholQR` to Krylov subspace methods that require tall-and-skinny QR factorizations, particularly block,  $s$ -step, and enlarged Krylov methods. Finally, applying the CountSketch matrix could potentially be optimized better than using a sparse-matrix vector multiplication by using a custom routine to add/subtract subsets of randomly selected rows in parallel using batched BLAS-1 routines, which should be investigated.

**5. Acknowledgements.** 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. This work was in part 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.

## REFERENCES

- [1] D. ACHLIOPHAS, *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, Journal of Computer and System Sciences, 66 (2003), pp. 671–687. Special Issue on PODS 2001.
- [2] E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, Third ed., 1999.
- [3] O. BALABANOV AND L. GRIGORI, *Randomized Gram-Schmidt process with application to GMRES*, SIAM J. Sci. Comput., 44 (2022), pp. A1450–A1474.
- [4] M. CHARIKAR, K. CHEN, AND M. FARACH-COLTON, *Finding frequent items in data streams*, in Automata, Languages and Programming, Berlin, Heidelberg, 2002, Springer Berlin Heidelberg, pp. 693–703.
- [5] J. DEMMEL, L. GRIGORI, M. HOEMMEN, AND J. LANGOU, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM Journal on Scientific Computing, 34 (2012), pp. A206–A239.
- [6] H. C. EDWARDS, C. R. Trott, AND D. SUNDERLAND, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202 – 3216. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.
- [7] T. FUKAYA, R. KANNAN, Y. NAKATSUKASA, Y. YAMAMOTO, AND Y. YANAGISAWA, *Shifted Cholesky QR for computing the QR factorization of ill-conditioned matrices*, SIAM Journal on Scientific Computing, 42 (2020), pp. A477–A503.
- [8] T. FUKAYA, Y. NAKATSUKASA, Y. YANAGISAWA, AND Y. YAMAMOTO, *CholeskyQR2: A simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system*, in 2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, 2014, pp. 31–38.

- [9] X. MENG AND M. W. MAHONEY, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC '13, New York, NY, USA, 2013, Association for Computing Machinery, p. 91–100.
- [10] Y. NAKATSUKASA AND J. A. TROPP, *Fast & accurate randomized algorithms for linear systems and eigenvalue problems*, 2021. arXiv:2111.00113.
- [11] NVIDIA, *cuBLAS documentation*. <https://docs.nvidia.com/cuda/cUBLAS/index.html>. Accessed: 2023-06-21.
- [12] ———, *CUDA toolkit documentation*. <https://docs.nvidia.com/cuda/>. Accessed: 2023-06-21.
- [13] ———, *cuSOLVER documentation*. <https://docs.nvidia.com/cuda/cusolver/index.html>. Accessed: 2023-06-21.
- [14] ———, *cuSPARSE documentation*. <https://docs.nvidia.com/cuda/cusparse/index.html>. Accessed: 2023-06-21.
- [15] S. RAJAMANICKAM, S. ACER, L. BERGER-VERGIAT, V. Q. DANG, N. D. ELLINGWOOD, E. HARVEY, B. KELLEY, C. R. TROTT, J. J. WILKE, AND I. YAMAZAKI, *Kokkos kernels: Performance portable sparse/dense linear algebra and graph kernels*, CoRR, abs/2103.11991 (2021).
- [16] T. SARLOS, *Improved approximation algorithms for large matrices via random projections*, in 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 143–152.
- [17] D. P. WOODRUFF, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014), p. 1–157.
- [18] Y. YAMAMOTO, Y. NAKATSUKASA, Y. YANAGISAWA, AND T. FUKAYA, *Roundoff error analysis of the Cholesky QR2 algorithm*, Electronic Transactions on Numerical Analysis, 44 (2015), pp. 306–326.

## COMPUTATIONAL ASPECTS OF OPTIMAL EXPERIMENTAL DESIGN FOR LARGE-SCALE BAYESIAN LINEAR INVERSE PROBLEMS

NICK NEUBERGER\*, BART G VAN BLOEMEN WAANDERS†, AND ALEN ALEXANDERIAN ‡

**Abstract.** The advent of high-performance-computing and scalable methods for tackling complex PDE-driven problems has opened the door to the use of powerful tools such as Bayesian inversion and optimal experimental design (OED) on a new scale. This document presents several applications of OED tools within the context of an example model: the steady-state convection-diffusion PDE. Estimating the source term of this equation is an example of a linear inverse problem. We first focus on deterministic inversion with the introduction of fundamental concepts and operators needed later in Bayesian inversion. Bayesian inversion is then used to estimate the source in a probabilistic sense. Solving the Bayesian problem motivates the OED formulation. In this context, a sensor configuration is sought such that uncertainty in the estimated parameter is minimized. Specifically, we focus on computing A-optimal sensor placements, where we seek to minimize average posterior variance. Our emphasis is in computational aspects of OED for infinite-dimensional inverse problems. We conclude our computational experiments with a demonstration of the effectiveness of optimal design strategy. Specifically, we demonstrate that the A-optimal design outperforms naive or random sensor placements.

**1. Introduction.** We consider optimal experimental design (OED) for infinite-dimensional Bayesian linear inverse problems governed by partial differential equations (PDEs). Specifically, we focus on optimal placement of sensors, where measurement data are collected. Solving such problems requires an intricate combination of methods and theories from probability, linear algebra, PDEs, and mathematical and numerical analysis. Also, in practice, one has to work within a computational budget. In this work, we explore computational aspects of OED for large-scale Bayesian linear inverse problem with an eye toward implementations on HPC architectures.

**1.1. The model problem.** We focus on a stationary convection-diffusion model for example dynamics. It is sufficiently complex to produce some interesting results, yet simple enough to facilitate the omission of unnecessary problem-dependent matters. Take  $\Omega \subset \mathbb{R}^2$  to be a bounded domain and consider the following stationary PDE:

$$\begin{aligned} -\alpha \Delta u + \boldsymbol{\nu} \cdot \nabla u &= m(\boldsymbol{x}) && \text{in } \Omega, \\ u &\equiv 0 && \text{on } \Gamma_1, \\ \nabla u \cdot \boldsymbol{n} &\equiv 0 && \text{on } \Gamma_2. \end{aligned} \tag{1.1}$$

Here,  $u$  is the state variable. We assume that  $\Omega = (0, 1)^2$  and that both the diffusion  $\alpha$  and the advection field  $\boldsymbol{\nu}$  are constant. Additionally, the composite boundary is such that  $\Gamma_1 \cup \Gamma_2 = \partial\Omega$ , with  $\boldsymbol{n}$  being the outward normal unit vector to the boundary. We let the source  $m \in L^2(\Omega)$  be the inversion parameter that we seek to estimate.

One can formulate an inverse problem in the deterministic and Bayesian settings. The deterministic formulation seeks to find a representation of the inversion parameter that reproduces the data most accurately under the governing model. On the other hand, the Bayesian approach is used to construct a statistical description of the parameter that accounts for uncertainties in parameter estimation. Realizations of the Bayesian posterior law may be propagated through the model resulting in a probability distribution on the state variable. In both approaches, we must contend with noisy data and the ill-posedness of the problem, due to sparse measurements and smoothing properties of the forward mapping.

---

\*Department of Mathematics, North Carolina State University, jnneuber@ncsu.edu

†Center for Computing Research, Sandia National Labs, bartv@sandia.gov

‡Department of Mathematics, North Carolina State University, alexanderian@ncsu.edu

In what follows, we discuss both deterministic and Bayesian formulations of the inverse problem, before tackling the OED problem of determining an optimal sensor placement. We will see that the OED problem is built on the Bayesian inverse problem. Thus, the understanding and manipulation of the components of the Bayesian formulation is required before we can develop the OED problem formulation and its solution methods. However, before discussing the matter of uncertainty in parameter estimation, it is instructive to consider the deterministic setting. This will allow us to build the key operators and components required in subsequent problems. Once this is done, converting the deterministic approach into the Bayesian formulation is straightforward, in the case of linear inverse problems.

**1.2. Related work.** The field of Bayesian inverse problems and optimal experimental design contain many open areas of research. A major motivation for the development of these fields are the numerous practical engineering and scientific applications. The formulation and solution of such problems requires an assortment of theoretical and computational tools at one's disposal. This is the reason why research pertaining to Bayesian inverse problems and OED necessitates rigorous knowledge of underlying theory and efficient implementation of algorithms. While the present work focuses on linear Bayesian inverse problems and A-optimal designs, there exists extensive works on design of nonlinear Bayesian inverse problem; see e.g., [9, 5, 1, 12].

The works [4, 2, 3], which concern OED for Bayesian linear inverse problems are closely related to the present work. In particular, [2] provides an accessible introduction on finite dimensional Gaussian random variables, linear inverse problems in the finite dimensional setting, and *Bayes Risk*, which we discuss later. The paper [3] provides a detailed perspective on OED for Bayesian linear inverse problems in Hilbert space with focus on Bayesian A- and D-optimal designs. The article [4] details a scalable approach for tackling the linear OED problem in a manner suitable for HPC architectures.

**1.3. Finite element method and the observation operator.** We discretize (1.1) in the space defined by  $\mathbb{V}_n := \text{span}\{\phi_i(\mathbf{x})\}_{i=1}^n$ , which is the space spanned by the finite-element (FE) basis  $\{\phi_i\}_{i=1}^n$ . For all computations, we choose  $\phi_i$  to be Lagrange nodal-basis-functions. For  $v \in L^2(\Omega)$ , FE coefficients can be obtained such that

$$v \approx v_h := \sum_{i=1}^n v_i \phi_i.$$

In particular, the discretized inversion parameter is given by  $m_h = \sum_i m_i \phi_i$ . In a Bayesian formulation, we use measurement data in conjunction with prior knowledge about the inversion parameter to estimate the vector  $\mathbf{m} = [m_1 \ m_2 \ \dots \ m_n]^\top$  of FE coefficients of  $m_h$ .

Let  $\mathbf{d} \in \mathbb{R}^d$  denote the vector of measurement data, collected at locations  $\{\mathbf{x}_j\}_{j=1}^d$ , where  $\mathbf{x}_j \in \Omega$ . Data are often collected at uniform or random sensor locations. This is sufficient for demonstrating the mechanics of the inverse problem and solution methods, but may be suboptimal in applications. We discuss this later when formulating the optimal experimental design problem, which concerns placing sensors in “optimal” locations.

We now introduce the first of several operators that appear in formulation of the inverse problem. Let  $\mathcal{B} : L^2(\Omega) \rightarrow \mathbb{R}^d$  be the observation map. This operator takes a function in  $L^2(\Omega)$  and produces measurements at the locations described by  $\{\mathbf{x}_j\}$ . Note that the observation map does not account for noise when collecting data.

**2. Deterministic inversion.** Much of the problem structure developed in the deterministic formulation will translate to the Bayesian setting. For this reason, various constructs are carefully built in the following section. In a deterministic formulation, we

formulate the inverse problem as that of finding an  $m$  that minimizes

$$\mathcal{J}(m) := \frac{1}{2} \|\mathcal{B}u - \mathbf{d}\|_2^2 + \mathcal{R}(m; \gamma), \quad (2.1)$$

where  $u$  is obtained by solving (1.1). The first term of the functional is referred to as the data-misfit. This term is a measure of the error between model observations and data  $\mathbf{d}$ . It is assumed that  $\mathbf{d} = \mathcal{B}u + \boldsymbol{\eta}$ , where the additive noise is normally distributed with mean zero and covariance  $\sigma^2 \mathbf{I}$ . That is,  $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ . Note that this assumes uncorrelated measurements. The second term of  $\mathcal{J}$  is a regularization term. For illustration in this section, we consider an  $L^2$ -regularization,  $\mathcal{R}(m; \gamma) := \frac{\gamma}{2} \|m\|_{L^2(\Omega)}^2$ . This term is required to manage the ill-posedness of the problem. The deterministic inverse problem maybe solved numerically in one of the following ways:

1. Use variational methods to form a system that results in  $m^* \in L^2(\Omega)$ , the minimizer of  $\mathcal{J}$ . Then, discretize this system to obtain the FE coefficients  $\mathbf{m}^*$ , which uniquely characterize the approximation  $m_h^*$  of  $m^*$ .
2. Discretize (1.1) and (2.1), and then use variational methods to obtain a system that uniquely determines  $\mathbf{m}^*$ , the FE coefficients of  $m_n^*$ .

Approach 1 is referred to as the optimize-then-discretize method (OTD), and 2 as discretize-then-optimize (DTO) approach. Due to the linearity of the present inverse problem, both OTD and DTO methods result in the symmetric positive definite system:

$$(\mathbf{F}^* \mathbf{F} + \gamma \mathbf{I}) \mathbf{m} = \mathbf{F}^* \mathbf{d}. \quad (2.2)$$

As discussed below,  $\mathbf{F}$  is the discretized parameter-to-observable map and  $\mathbf{F}^*$  is its adjoint.

**2.1. Discretization issues.** To define the adjoint operator  $\mathbf{F}^*$  in (2.2), we need to pay close attention to the inner product on the discretized inversion parameter space. This discretized space is  $\mathbb{R}^n$  equipped with the following *mass-weighted* inner product:

$$(\mathbf{u}, \mathbf{v})_{\mathbf{M}} = \mathbf{v}^T \mathbf{M} \mathbf{u},$$

where  $\mathbf{M}$  is the finite element mass matrix,  $M_{ij} = \int_{\Omega} \phi_i \phi_j$  with  $i, j \in \{1, \dots, n\}$ . Note that this mass weighted inner product is none but the discretized  $L^2(\Omega)$  inner product. To see this, let  $(\cdot, \cdot)$  be the  $L^2(\Omega)$  inner product and let  $u_h$  and  $v_h$  be in  $\mathbb{V}_n$ . Then,

$$(u_h, v_h) = \left( \sum_{j=1}^n u_j \phi_j, \sum_{i=1}^n v_i \phi_i \right) = \sum_{i,j=1}^n v_i u_j (\phi_i, \phi_j) = \mathbf{v}^T \mathbf{M} \mathbf{u} = (\mathbf{u}, \mathbf{v})_{\mathbf{M}}.$$

Note that  $\mathbf{u}$  and  $\mathbf{v}$  are the vectors of FE coefficients corresponding to  $u_h$  and  $v_h$ , respectively. In what follows, we let  $\mathbb{R}_{\mathbf{M}}^n$  denote the inner product space  $\mathbb{R}^n$  equipped with the mass-weighted inner product.

Next, we describe the remaining components of (2.2), the matrices  $\mathbf{F}$  and  $\mathbf{F}^*$ . The map  $\mathbf{F} : \mathbb{R}_{\mathbf{M}}^n \rightarrow \mathbb{R}^d$  is a discretization of the parameter-to-observable map  $\mathcal{F} : L^2(\Omega) \rightarrow \mathbb{R}^d$  and  $\mathbf{F}^* : \mathbb{R}^d \rightarrow \mathbb{R}_{\mathbf{M}}^n$  is a discretization of the adjoint map  $\mathcal{F}^* : \mathbb{R}^d \rightarrow L^2(\Omega)$ . Additionally, denote  $\bar{\mathbf{F}} : \mathbb{R}_{\mathbf{M}}^n \rightarrow \mathbb{R}_{\mathbf{M}}^n$  as the discretization of the parameter-to-state map  $\bar{\mathcal{F}} : L^2(\Omega) \rightarrow H^1(\Omega)$ . Obtaining these matrices (or their applications) is discussed shortly below. The last operator we discretize is the observation map. Denote  $\mathbf{B}$  as the discretization of the observation operator  $\mathcal{B}$ . It is important to note that constructing this operator is not only problem-dependent, but also dependent upon the selected software. A reasonable approach to extract a measurement value from a function  $v \in \mathbb{V}_n$  at a point  $\mathbf{x} \in \Omega$  is to numerically integrate  $v$  against a sufficiently scaled Gaussian-like function that approximates the  $\delta$  distribution

centered at  $\mathbf{x}$ . Doing so is equivalent to obtaining a local average of  $v$  about the point  $\mathbf{x}$ . This process is linear and a single matrix  $\mathbf{B} : \mathbb{R}_{\mathbf{M}}^n \rightarrow \mathbb{R}^d$  can perform simultaneous observations, approximating the action of  $\mathcal{B}$ .

Following a DTO strategy, the discretized objective functional with  $L^2$ -regularization is

$$\mathcal{J}(\mathbf{m}) := \frac{1}{2} \|\mathbf{Bu} - \mathbf{d}\|_2^2 + \frac{\gamma}{2} \mathbf{m}^T \mathbf{M} \mathbf{m}, \quad (2.3)$$

where  $\mathbf{u}$  solves the discretized state equation, which we discuss next. Define the matrix  $A_{ij} := \alpha(\nabla\phi_i, \nabla\phi_j) + (\phi_i, \boldsymbol{\nu} \cdot \nabla\phi_j)$ . The discretized state equation is described as follows:

$$\mathbf{v}^T \mathbf{A} \mathbf{u} = \mathbf{v}^T \mathbf{M} \mathbf{m}, \quad \forall \mathbf{v} \in \mathbb{R}_{\mathbf{M}}^n \iff \mathbf{A} \mathbf{u} = \mathbf{M} \mathbf{m}. \quad (2.4)$$

This system gives an explicit representation of the discretized parameter-to-state and parameter-to-observable maps:

$$\begin{aligned} \text{parameter-to-state: } & \bar{\mathbf{F}} := \mathbf{A}^{-1} \mathbf{M}, \\ \text{parameter-to-observable: } & \mathbf{F} := \mathbf{B} \bar{\mathbf{F}}. \end{aligned}$$

In practice, an application of  $\mathbf{F}$  (a state solve) requires solving a linear system. Using  $\mathbf{F}$ ,  $\mathcal{J}$  can be redefined in terms of  $\mathbf{m}$ :

$$\mathcal{J}(\mathbf{m}) := \frac{1}{2} \|\mathbf{F} \mathbf{m} - \mathbf{d}\|_2^2 + \frac{\gamma}{2} \mathbf{m}^T \mathbf{M} \mathbf{m}. \quad (2.5)$$

It can be shown that  $\nabla \mathcal{J}(\mathbf{m}) = 0$  results in (2.2).

**2.2. Computational results.** Computing the minimizer of  $\mathcal{J}$  requires implementing routines that perform applications of the forward and adjoint operators,  $\mathbf{F}$  and  $\mathbf{F}^*$  and then solving a symmetric positive definite (SPD) system using a Krylov iterative method. This process is matrix free. It does not require building the forward and adjoint operators, which is an important consideration in large-scale problems. The expressions that define the forward/adjoint equations can be assembled using FE software. We use the FEnCS 2019 library [6] in Python for discretization, PDE solves, and assembly of relevant matrices. In the case of the steady-state convection/diffusion model, applications of the forward/adjoint operators require solving linear systems.

Performing deterministic inversion is equivalent to minimizing (2.3) over  $\mathbb{R}_{\mathbf{M}}^n$ . Once the FE coefficients that minimize  $\mathcal{J}$  are obtained, the expansion of the minimizer over  $\mathbb{V}_n$  may be plotted and relative error calculated. Hence, the true parameter may be compared with its approximation. We implement an equally spaced FE mesh of size  $n = n_x^2$  with  $k^2$  uniformly distributed interior sensors. The sub-boundary  $\Gamma_2$  is the  $x = 1$  side of the unit square. Taking  $\boldsymbol{\nu} = [1.5, 0]$  provides “flow” out of the boundary. In the present study, the true parameter is given by  $m_{\text{true}}(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x})$ , where

$$f(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in R \\ 0, & \mathbf{x} \notin R \end{cases} \quad \text{and} \quad g(\mathbf{x}) = 1.5 \left\{ -\frac{1}{2(0.15)^2} [(x_1 - 0.5)^2 + (x_2 - 0.5)^2] \right\},$$

and  $R$  is the rectangle  $R(\mathbf{x}) = [0.125, 0.5] \times [0.375, 0.625]$ . Physical and problem-specific parameters are provided in the Table 2.1. Note that  $15^2$  uniformly distributed sensors are selected. Results are obtained with synthetic data such that  $\|\mathbf{d} - \mathbf{Bu}\|_2 / \|\mathbf{d}\|_2 = 0.0016$ . This fraction is viewed as the relative magnitude of the noise.

Parameter	$n_x$	$k$	$\alpha$	$\gamma$
Value	20	15	0.06	0.007

Table 2.1: Parameters for the deterministic inversion problem.

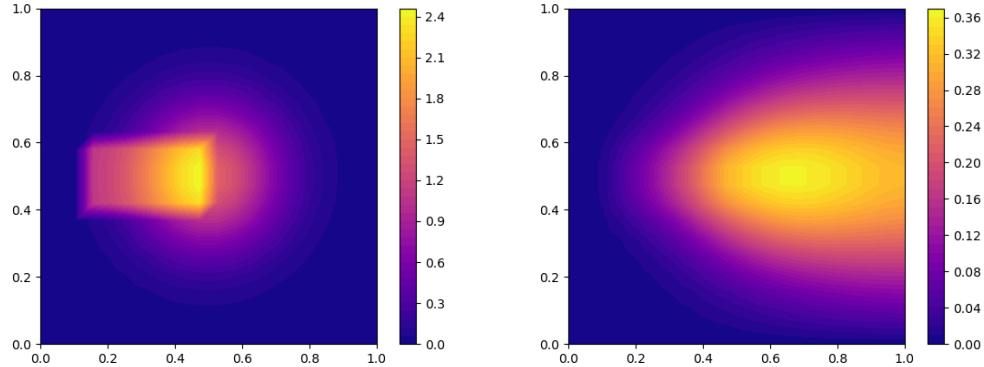


Fig. 2.1: The true parameter  $m_{\text{true}}$  used to generate data (left) and the corresponding state solution  $u$  (right).

We see that the true parameter has been advected in the positive  $x$ -direction, resulting in outflow. In addition to this, the amplitude of the parameter is decreased by the diffusive properties of the model. Performing the inversion, we see the estimate captures the features

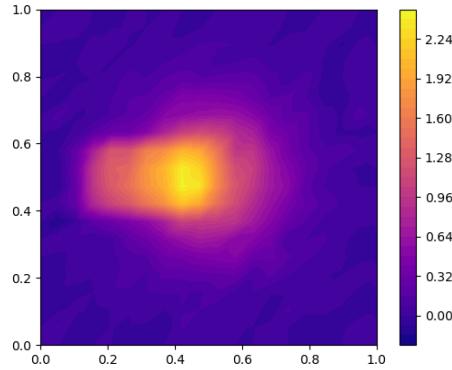


Fig. 2.2: Solution to the deterministic inverse problem.

of the true parameter relatively well. Quantitatively, a mass weighted relative error of 0.0790 is achieved. We use the mass weighted inner product to measure the relative error since we are concerned about the error in approximating the true solution in the space  $\mathbb{V}_n$ . The true parameter has sharp edges, which are notoriously difficult to reconstruct in diffusion models. Nonetheless, all major features in  $m_{\text{true}}$  are present in the reconstruction.

**3. Bayesian inversion.** Here, we discuss the Bayesian formulation of the inverse problem under study. The objective of the Bayesian inverse problem is to obtain a probabilistic description of the inversion parameter from noisy data. The present discussion is based

on the developments in [7], which outlines a computational framework for solving infinite-dimensional Bayesian linear inverse problems. For theory of Bayesian inverse problems in the infinite-dimensional setting, the readers can refer to [10].

Recall the infinite-dimensional representation of the parameter-to-observable map  $\mathcal{F} : L^2(\Omega) \rightarrow \mathbb{R}^d$ . It is assumed that noise is additive and Gaussian with mean zero and covariance denoted by  $\boldsymbol{\Gamma}_{\text{noise}}$ :

$$\mathbf{d} = \mathcal{F}\mathbf{m} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(0, \boldsymbol{\Gamma}_{\text{noise}}). \quad (3.1)$$

Note that for the present example, the parameter-to-observable map is a continuous linear transformation. The Bayesian formulation entails forming a prior law—a prior statistical description of the inversion parameter informed by whatever information is available. The prior, in conjunction with the data-likelihood, the law of the data conditioned on the inversion parameter, facilitates the use of Bayes' Theorem. Assuming a Gaussian prior, application of Bayes' theorem implies that the posterior law, for the present inverse problem, is Gaussian with analytic expressions for the posterior mean and covariance. In this case, the posterior mean coincides with the *maximum a posteriori probability (MAP) point*. This and the posterior covariance are sufficient in describing the posterior law. Before we provide the formulas for the MAP point and the posterior covariance, we discuss the definition of a suitable Gaussian prior in the present infinite-dimensional setting.

**3.1. The prior.** The prior mean, given by  $m_0 \in L^2(\Omega)$ , may be informed by problem-specific information. When no such information is available, the prior mean may be set as the zero function. This is our decision for all subsequent computations. A clever choice of prior covariance operator is the square of the solution operator of an elliptic PDE. This provides advantageous smoothing properties and the important property of being *trace-class*, positive, and selfadjoint. Additionally, the prior covariance being the *square* of the inverse of the elliptic operator makes applications of the square root of the covariance straightforward [7]. Let  $s \in L^2(\Omega)$  and  $a_1, a_2 > 0$ . Then, the mentioned elliptic PDE is

$$\begin{aligned} -a_1 \Delta m + a_2 m &= s, & \text{in } \Omega, \\ \nabla m \cdot \mathbf{n} &= 0, & \text{on } \partial\Omega. \end{aligned}$$

This corresponds to the weak form

$$a_1(\nabla m, \nabla p) + a_2(m, p) = (s, p), \quad \forall p \in H^1(\Omega). \quad (3.2)$$

If the operator  $\mathcal{A}$  is such that  $\mathcal{A}m = s$ , then the solution operator of the PDE is  $\mathcal{A}^{-1}$ . The prior covariance is defined to be the square of this:  $\mathcal{C}_0 = \mathcal{A}^{-2}$ . It is also important to note the role of the parameters  $a_1$  and  $a_2$ , responsible for the behavior of the prior law. Here, the prior law describes a Gaussian random field, whose correlation length and pointwise variance are controlled by  $a_1$  and  $a_2$ , respectively.

For computational purposes, we need the FE coordinates of the prior mean,  $\mathbf{m}_0$ , and the discretized prior covariance,  $\boldsymbol{\Gamma}_{\text{pr}}$ . To arrive at the prior covariance, we discretize (3.2) with regards to the FE basis. Define the matrix  $\mathbf{K}$  according to  $K_{ij} = (\nabla \phi_i, \nabla \phi_j)$ . Then, the discretized elliptic equation is given by

$$\mathbf{M}^{-1}(a_1 \mathbf{K} + a_2 \mathbf{M})\mathbf{m} = \mathbf{s}.$$

Additionally, define  $\mathbf{E} := \mathbf{M}^{-1}(a_1 \mathbf{K} + a_2 \mathbf{M})$ , then it can be shown that  $\mathbf{E}$  and  $\mathbf{E}^{-1}$  are selfadjoint with respect to the mass-weighted inner product  $(\cdot, \cdot)_{\mathbf{M}}$ . Using the FE coefficients of the prior mean and the discretized elliptic operator, we introduce the prior distribution:

$$\pi_{\text{pr}}(\mathbf{m}) \propto \exp \left\{ -\frac{1}{2} \|\mathbf{E}(\mathbf{m} - \mathbf{m}_0)\|_{\mathbf{M}}^2 \right\}. \quad (3.3)$$

This also indicates  $\mathbf{\Gamma}_{\text{pr}} := \mathbf{E}^{-2}$ .

**3.2. The likelihood.** The likelihood law provides a probabilistic description of the data conditioned on the inversion parameter. Since the noise is Gaussian with mean zero, the likelihood is also Gaussian:

$$\pi_{\text{like}}(\mathbf{d}|\mathbf{m}) \propto \exp \left\{ -\frac{1}{2} \|\mathbf{F}\mathbf{m} - \mathbf{d}\|_{\mathbf{\Gamma}_{\text{noise}}^{-1}}^2 \right\}. \quad (3.4)$$

The fact that  $\mathbf{d}|\mathbf{m} \sim \mathcal{N}(\mathbf{F}\mathbf{m}, \mathbf{\Gamma}_{\text{noise}})$  follows from the independence of  $\mathbf{m}$  and  $\boldsymbol{\eta}$ . We also note the role of the noise covariance by viewing the formula above. In the case of uncorrelated measurements, or when  $\mathbf{\Gamma}_{\text{noise}} = \sigma^2 \mathbf{I}_d$ , sensor locations corresponding to high-noise measurements will not be influential. This is due to the fact the diagonal entry of  $\mathbf{\Gamma}_{\text{noise}}^{-1}$  corresponding to the high-noise measurement will effectively be zero. We will return to this idea later in the OED discussion.

**3.3. The posterior.** Since the prior and likelihood are given in discretized form, Bayes' Theorem can be applied as follows:

$$\pi_{\text{post}}(\mathbf{m}) \propto \pi_{\text{like}}(\mathbf{d}|\mathbf{m}) \pi_{\text{pr}}(\mathbf{m}). \quad (3.5)$$

By substituting the expressions for the likelihood and prior density in the above formula and some algebraic manipulations, we can describe the posterior law explicitly: the posterior law is a Gaussian with mean  $\mathbf{m}_{\text{map}}$  and covariance  $\mathbf{\Gamma}_{\text{post}}$  defined according to

$$\mathbf{m}_{\text{map}} = \mathbf{\Gamma}_{\text{post}} (\mathbf{F}^* \mathbf{\Gamma}_{\text{noise}}^{-1} \mathbf{d} + \mathbf{\Gamma}_{\text{pr}}^{-1} \mathbf{m}_0), \quad \mathbf{\Gamma}_{\text{post}} = (\mathbf{F}^* \mathbf{\Gamma}_{\text{noise}}^{-1} \mathbf{F} + \mathbf{\Gamma}_{\text{pr}}^{-1})^{-1}. \quad (3.6)$$

It is important to note several details about the posterior information:

- Applications of  $\mathbf{\Gamma}_{\text{post}}$  require a linear solve.
- Obtaining  $\mathbf{m}_{\text{post}}$  requires a linear solve.
- Defining  $\mathbf{\Gamma}_{\text{noise}} = \sigma^2 \mathbf{I}_d$  simplifies calculations.

**3.4. Computational considerations.** Being able to solve the Bayesian inversion problem is essential to constructing the OED problem. For this reason, it is prudent to perform various theoretical verifications such as verifying the implementation of the adjoint operator and the gradient and Hessian expressions. To make calculations as fast as possible, it is necessary to use suitable matrix factorizations and matrix-free approaches when available. Doing so vastly decreases computational time and results in an approach suitable for large scale problems (it may be practical to impose some computational budget).

To illustrate some of the computational issues, we consider the computation of the MAP point. This is done by solving the system

$$\mathbf{\Gamma}_{\text{post}}^{-1} \mathbf{m} = \mathbf{F}^* \mathbf{\Gamma}_{\text{noise}}^{-1} \mathbf{d} + \mathbf{\Gamma}_{\text{pr}}^{-1} \mathbf{m}_0$$

using a Krylov iterative approach. To enable this, we create a routine that applies  $\mathbf{\Gamma}_{\text{post}}^{-1}$  to a vector. An application of  $\mathbf{\Gamma}_{\text{post}}^{-1}$  requires both a state and adjoint solve, as well as a multiplication of  $\mathbf{\Gamma}_{\text{pr}}^{-1}$  with a vector. Matrix-free Krylov iterative methods are essential for obtaining the MAP point quickly. Due to SPD system structure, we rely on the (preconditioned) Conjugate Gradient (CG) for solving the resulting linear system. For more details on computational aspects of solving Bayesian linear inverse problems, see [7].

**3.5. Computational results.** We perform an illustrative computational study in the Bayesian framework as we did in the deterministic setting. The same discretization method, boundary conditions, advection, and true parameter are used. Doing so will allow for the comparison between the minimizer obtained in the deterministic formulation and the Bayesian MAP point. A significant difference between the two problems, related to our discussions of OED, is that measures for the quality of the reconstructions maybe obtained naturally in the Bayesian setting. Namely, we may consider various ways of characterizing the uncertainty in the estimated parameters. For example, the trace of the posterior covariance quantifies the average variance of the posterior distribution. We can also study the percentage of average variance reduction from performing inversion. This is represented by the quantity  $\Delta\text{Var} := 100 \cdot (1 - \text{tr}[\mathbf{\Gamma}_{\text{post}}]/\text{tr}[\mathbf{\Gamma}_{\text{prior}}])$ . We note that for large-scale problems, computing the trace directly is inefficient. For this reason, a randomized trace estimator is used. This will be discussed further in Section 4.

In the present example, we use a Gaussian prior as described in Section 3.1. Regarding the noise model, we assume an additive Gaussian noise model, and let the noise covariance be of the form  $\mathbf{\Gamma}_{\text{noise}} = \sigma^2 \mathbf{I}_d$ . We summarize the parameters defining the various aspects of the Bayesian inverse problem under study in Table 3.1.

Parameter	$n_x$	$k$	$\alpha$	$a_1$	$a_2$	$\sigma^2$
Value	20	15	0.06	0.01	0.08	0.001

Table 3.1: Parameters for the Bayesian inversion problem.

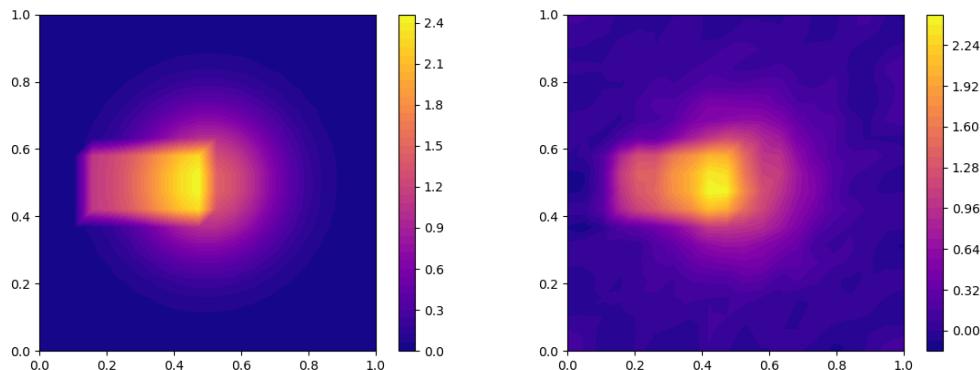


Fig. 3.1: The true parameter  $m_{\text{true}}$  used to generate data (left) and  $m_{\text{map}}$  (right).

Results are obtained with synthetic noise level is set such that  $\|\mathbf{d} - \mathbf{B}\mathbf{u}\|_2/\|\mathbf{d}\|_2 = 0.0021$ . Qualitatively, the MAP point captures the major features of  $m_{\text{true}}$ . The relative error between the true parameter and MAP point is calculated to be 0.0777. Note that this is in par with our results from deterministic inversion. Regarding uncertainty reduction, we observed an average variance reduction of  $\Delta\text{Var} = 99.8692\%$ . A variance reduction of this magnitude should not be expected in real applications. The reasons for such a large uncertainty reduction can be attributed to a “wide prior” and dense sensor network. By a

“wide prior” we mean that the parameters governing the prior ( $a_1$  and  $a_2$ ) are selected such that the prior has a large average variance.

**4. Optimal experimental design.** The purpose of solving the OED problem is to obtain optimal sensor locations in which to collect data. In this context, a design is the placement of measurement locations within the domain  $\Omega$  and optimality is determined through a measure of uncertainty associated with the inversion parameters. Our overall strategy aligns closely to that in [4]. Specifically, we consider Bayesian A-optimal designs that seek to minimize the average posterior variance.

Let  $\mathbb{S} := \{\mathbf{x}_j\}_{j=1}^{N_s}$ ,  $\mathbf{x}_i \in \Omega$ , be the set of possible sensor locations, otherwise known as the candidate sensor locations. To each location  $\mathbf{x}_j$ , we assign a weight,  $w_j$ . If  $\mathbf{w} = [w_1, \dots, w_{N_s}]^T$  is the vector containing the design weights, the optimal design is obtained by minimizing the OED functional

$$\Psi(\mathbf{w}) := \Theta(\mathbf{w}) + \Phi(\mathbf{w}; \beta), \quad \mathbf{w} \in [0, 1]^{N_s}. \quad (4.1)$$

The functional  $\Theta : \mathbb{R}^{N_s} \rightarrow \mathbb{R}$  is referred to as the *design criterion*. Deciding upon a criterion is typically problem-dependent. Specifically, the choice of criterion depends upon problem structure, the chosen definition of uncertainty, and related quantities of interest. The functional  $\Phi : \mathbb{R}^{N_s} \times \mathbb{R}_+ \rightarrow \mathbb{R}$  is the *penalty* function whose role is to promote sparsity. Increasing the penalty parameter  $\beta$  results in sparser designs.

**4.1. The design criterion.** To arrive at a design criterion, the design weights must be introduced into the Bayesian inversion problem. This is done by defining a suitably weighted data-likelihood. Let  $\mathbf{W} \in \mathbb{R}^{N_s \times N_s}$  be a diagonal matrix such that  $\text{diag}[\mathbf{W}] = \mathbf{w}$  and  $\mathbf{\Gamma}_{\text{noise}} = \sigma^2 \mathbf{I}_{N_s}$  for all subsequent calculations. Then,

$$\pi_{\text{like}}(\mathbf{d}|\mathbf{m}; \mathbf{w}) \propto \exp \left\{ -\frac{1}{2} \|\mathbf{W}^{1/2}(\mathbf{F}\mathbf{m} - \mathbf{d})\|_{\mathbf{\Gamma}_{\text{noise}}^{-1}}^2 \right\}. \quad (4.2)$$

Note that in the case when  $\mathbf{W} = \mathbf{I}_{N_s}$ , the original likelihood (3.4) is obtained. In the context of binary weights, this is equivalent to collecting data at all possible measurement locations. In the case of nonbinary weights, we view (4.2) as placing bias upon specific sensor locations. Candidate locations  $\mathbf{x}_j$  corresponding to larger  $w_j$  indicate these locations are highly influential in terms of reducing uncertainty. The corresponding “weighted” posterior covariance operator is given by

$$\mathbf{\Gamma}_{\text{post}}(\mathbf{w}) = (\sigma^{-2} \mathbf{F}^* \mathbf{W} \mathbf{F} + \mathbf{\Gamma}_{\text{pr}}^{-1})^{-1}. \quad (4.3)$$

It is worth noting that the data  $\mathbf{d}$  does not appear in the posterior covariance. In the present study, we focus on the Bayesian A-optimality criterion:

$$\Theta(\mathbf{w}) := \text{tr}[\mathbf{\Gamma}_{\text{post}}(\mathbf{w})]. \quad (4.4)$$

For the present linear Gaussian problem, this A-optimality criterion is equal to the Bayes Risk, which is an average mean square error of the posterior mean; see e.g., [2]. Thus, under our provided assumptions, reducing the average posterior variance corresponds to reducing parameter MAP estimation inaccuracy in an average sense. This will be revisited in our computational studies later in this section.

**4.2. Penalty of the OED objective.** As discussed, the role of the penalty term is to enforce sparsity in sensor placement. For simplicity, we consider an  $\ell_1$ -norm penalty:

$$\Phi(\mathbf{w}) := \beta \|\mathbf{w}\|_1. \quad (4.5)$$

This choice of  $\Phi$  is convex. However,  $w_j$  may take on any value in  $[0, 1]$ . This means that the  $\mathbf{w}^*$  that minimizes  $\Psi$  is not a binary vector. Our aim is to modulate  $\beta$  such that a desirable number of weights are nearly zero. Sensor locations corresponding to the remaining weights comprise the (approximate) optimal design. Since  $\mathbf{w} \in [0, 1]^{N_s}$ , the gradient is computed as  $\nabla\Phi(\mathbf{w}) = \beta\mathbf{1}$ . This will be necessary for utilizing a gradient-based optimization routine with box constraints in order to solve the OED problem. Although our choice of penalty function is sufficient for demonstration and application of the OED problem, more sophisticated penalty approaches exists. For example, in [4] a sequence of penalty functions is used to approximate the  $\ell_0$ -norm.

**4.3. Evaluating the criterion and its gradient.** As seen in the previous section, evaluating the penalty and its gradient for  $\mathbf{w} \in \mathbb{R}^{N_s}$  is straightforward. It is not the same case for the criterion. We outline a matrix-free approach, which is based on the developments in [4]. We begin by recalling that for a given weight vector  $\mathbf{w}$ , obtaining the MAP point is equivalent to minimizing

$$\hat{\mathcal{J}}(\mathbf{m}; \mathbf{w}) = \frac{1}{2\sigma^2} \|\mathbf{W}^{1/2}(\mathbf{F}\mathbf{m} - \mathbf{d})\|^2 + \frac{1}{2} \|\boldsymbol{\Gamma}_{\text{pr}}^{-1/2}(\mathbf{m} - \mathbf{m}_0)\|_{\mathbf{M}}^2. \quad (4.6)$$

A simple calculation show that the Hessian of  $\hat{\mathcal{J}}$  is

$$\mathbf{H}(\mathbf{w}) = \frac{1}{\sigma^2} \mathbf{F}^* \mathbf{W} \mathbf{F} + \boldsymbol{\Gamma}_{\text{pr}}^{-1}.$$

Defining the Hessian of the data-misfit term as  $\mathbf{H}_{\text{misfit}}(\mathbf{w}) := \frac{1}{\sigma^2} \mathbf{F}^* \mathbf{W} \mathbf{F}$  we write,

$$\mathbf{H}(\mathbf{w}) = \mathbf{H}_{\text{misfit}}(\mathbf{w}) + \boldsymbol{\Gamma}_{\text{pr}}^{-1}.$$

Thus, the posterior covariance can be defined in terms of the Hessian as

$$\boldsymbol{\Gamma}_{\text{post}}(\mathbf{w}) = \mathbf{H}^{-1}(\mathbf{w}).$$

Building the posterior covariance explicitly in order to compute the trace is not scalable. We will see shortly that having access to applications of  $\mathbf{H}^{-1}(\mathbf{w})$  is sufficient for estimating  $\Theta(\mathbf{w}) = \text{tr}[\boldsymbol{\Gamma}_{\text{post}}(\mathbf{w})]$  and its gradient. With Monte-Carlo methods, one can estimate the trace of an SPD matrix  $\mathbf{A}$ :

$$\text{tr}[\mathbf{A}] \approx \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} (\mathbf{y}_i, \mathbf{A} \mathbf{y}_i)_2,$$

where  $\mathbf{y}_i$  are independent random vectors. It is not as simple as applying this concept to  $\Theta(\mathbf{w})$ , given that an evaluation of  $\Theta$  at the point  $\mathbf{w}$  should approximate the trace of the infinite-dimensional representation of  $\boldsymbol{\Gamma}_{\text{post}}$ . For this reason, it is necessary to appropriately scale the random vectors  $\mathbf{y}_i$ . With this idea, we redefine the criterion in terms of its trace estimation:

$$\Theta(\mathbf{w}) := \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} (\mathbf{z}_i, \mathbf{H}^{-1} \mathbf{z}_i)_{\mathbf{M}}, \text{ for } \mathbf{z}_i := \mathbf{M}^{-1/2} \mathbf{y}_i. \quad (4.7)$$

See [4] for a derivation of the mass-weighted trace estimator.

Next, the gradient can be derived. The following equations and assumptions will be used:

$$1. \quad \frac{\partial \mathbf{H}^{-1}(\mathbf{w})}{\partial w_j} = -\mathbf{H}^{-1}(\mathbf{w}) \frac{\partial \mathbf{H}(\mathbf{w})}{\partial w_j} \mathbf{H}^{-1}(\mathbf{w})$$

2.  $\mathbf{H}^{-1}(\mathbf{w})$  is  $\mathbf{M}$ -symmetric
3.  $\frac{\partial \mathbf{H}_{\text{misfit}}(\mathbf{w})}{\partial w_j} = \mathbf{F}^* \mathbf{e}_j \mathbf{e}_j^T \mathbf{F}$ , where  $\mathbf{e}_j$  is the  $j^{\text{th}}$  standard unit vector in  $\mathbb{R}^{N_s}$ .

With the above assumptions and defining  $\mathbf{q}_i = \mathbf{H}^{-1} \mathbf{z}_i$ , we have that

$$\begin{aligned} \frac{\partial \Theta(\mathbf{w}_j)}{\partial w_j} &= \frac{1}{N_{tr}} \frac{\partial}{\partial w_j} \sum_{i=1}^{N_{tr}} (\mathbf{z}_i, \mathbf{H}^{-1}(\mathbf{w}) \mathbf{z}_i)_{\mathbf{M}} \\ &= -\frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left( \mathbf{z}_i, \mathbf{H}^{-1}(\mathbf{w}) \frac{\partial \mathbf{H}_{\text{misfit}}(\mathbf{w})}{\partial w_j} \mathbf{H}^{-1}(\mathbf{w}) \mathbf{z}_i \right)_{\mathbf{M}} \\ &= -\frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \left( \mathbf{q}_i, \frac{\partial \mathbf{H}_{\text{misfit}}(\mathbf{w})}{\partial w_j} \mathbf{q}_i \right)_{\mathbf{M}}. \end{aligned}$$

Note that  $\frac{\partial \mathbf{H}(\mathbf{w})}{\partial w_j} = \frac{\partial \mathbf{H}_{\text{misfit}}(\mathbf{w})}{\partial w_j}$  was used. Denote  $\mathbf{d}_i = \mathbf{F} \mathbf{q}_i$  then

$$\left( \mathbf{q}_i, \frac{\partial \mathbf{H}_{\text{misfit}}(\mathbf{w})}{\partial w_j} \mathbf{q}_i \right)_{\mathbf{M}} = \frac{1}{\sigma^2} (\mathbf{q}_i, \mathbf{F}^* \mathbf{e}_j \mathbf{e}_j^T \mathbf{F} \mathbf{q}_i)_{\mathbf{M}} = \frac{1}{\sigma^2} (\mathbf{F} \mathbf{q}_i, \mathbf{e}_j \mathbf{e}_j^T \mathbf{F} \mathbf{q}_i)_2 = \frac{1}{\sigma^2} [\mathbf{d}_i^{(j)}]^2.$$

Finally, the gradient of  $\Theta$  can be defined:

$$\frac{\partial \Theta(\mathbf{w})}{\partial w_j} = -\frac{1}{\sigma^2 N_{tr}} \sum_{i=1}^{N_{tr}} [\mathbf{d}_i^{(j)}]^2, \quad j = 1, \dots, N_s. \quad (4.8)$$

**4.4. Computational considerations.** The previous section provides enough information to implement a gradient-based optimization algorithm for the purpose of minimizing  $\Psi(\mathbf{w})$ . However, a practical implementation for large-scale problems is rather involved. See [4] for details. The implementations in the present study are simplistic and are intended for illustration. First,  $\mathbf{F}$ ,  $\mathbf{F}^*$ , and  $\Gamma_{pr}^{-1}$  are explicitly constructed. (In practical implementations, we need to implement routines that provide applications of the forward/adjoint operators.) A gradient-based optimization routine is then selected. In our implementations, we use the limited memory Broyden–Fletcher–Goldfarb–Shanno with bound constraints (*L-BFGS-B*) algorithm as implemented in the SciPy library [11] in Python. The algorithm was originally introduced in [13].

For a given  $\mathbf{w}$ ,  $\mathbf{q}_i$  are computed via SPD solves. In optimizing  $\Psi(\mathbf{w}) = \Theta(\mathbf{w}) + \Phi(\mathbf{w}; \beta)$ , evaluations of  $\Theta$  and its gradient are computed as described in the previous section.

Due to our choice of the penalty approach, the computed optimal design will, in general, be a nonbinary vector. However, the  $\ell_1$ -penalty ensures that some of the weights are nearly zero. In our numerical studies, we consider these weights as zero and select sensors corresponding to larger weights.

**4.5. Computational results.** In application, the number of sensors desired is likely to be a small number. The intuition behind this is as follows: if the number of sensors is such that the physical domain is saturated, moving the sensors around is unlikely to make a notable difference. On the other hand, when we have access to only a few sensors, their placement is crucial. For this reason, in our computational studies we seek Designs that are comprised of relatively few sensors.

Our results were obtained with a FE-mesh of size  $n = n_x^2$ . The Neumann boundary  $\Gamma_2$  represents the  $y = 0$  and  $x = 1$  sides of the domain, and the vector-field  $\boldsymbol{\nu} = [1.5, -1.5]$  produces outflow in the Southeast direction. Data is generated such that  $\|\mathbf{d} - \mathbf{B} \mathbf{u}\|_2 / \|\mathbf{d}\|_2 =$

0.006. Lastly, the sparsity parameter  $\beta$  was selected to produce relatively few sensors and the true parameter is given by

$$m_{\text{true}}(\mathbf{x}) = 8 \exp \left\{ -\frac{1}{2(0.17)^2} [(x_1 - 0.5)^2 + (x_2 - 0.5)^2] \right\}.$$

Parameter	$n_x$	$k$	$\alpha$	$a_1$	$a_2$	$\sigma^2$	$\beta$
Value	20	18	0.06	0.2905	0.151	0.1	0.2

Table 4.1: Parameters for the OED problem.

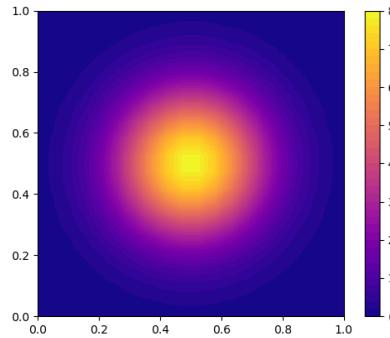


Fig. 4.1: The true source used to generate data:  $m_{\text{true}}$ .

The parameters in Table 4.1 result in an optimal design consisting of 9 sensor locations. We compare the OED result to results obtained using 9 uniform sensors located on the interior of  $\Omega$ , and 9 randomly selected locations. The average variance reduction of the

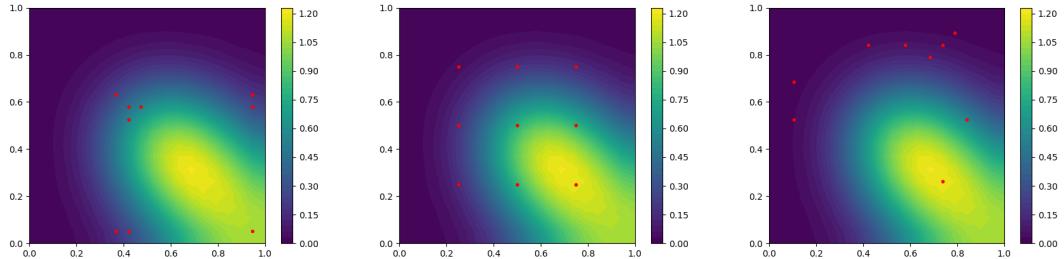


Fig. 4.2: The optimal design consisting of 9 sensors is depicted (left), a design of 9 uniform sensors (middle), and a design randomly distributed design (right). The designs are plotted over the state solution generated by the true parameter,  $m_{\text{true}}$ .

OED, uniform, and random solutions are 86.6488%, 78.4059%, and 73.0988% respectively.

It is important to highlight that all observed random placements were sub-optimal. Designs are plotted over the state solution. In this context, the optimal design is not surprising. We would not expect to see measurement locations in portions of the region devoid of dynamics, such as the Northwest corner of the plots. This is why the optimal placement is symmetric about the outflow direction. It is noteworthy that the random sensors performed poorly due to placement of sensors in uninformative locations and that the uniform sensors were much better than random (in most cases).

Next, we study the results of sensor placement as the number of desired sensors changes. As mentioned before, optimal sensor placement is critical when only a small number of sensors are used. On the other hand, as the number of sensors grow, even a random placement of sensor maybe be effective. This is studied in Figure 4.3. In that figure, each point corresponds to a design consisting of  $k$  randomly placed sensors. This random design is used to generate synthetic data for the Bayesian inverse problem using a fixed  $\mathbf{m}_{\text{true}}$ . We then compute the relative error between the true parameter and MAP point and the average posterior variance (i.e., the A-optimal criterion). The relative error is then plotted against the A-optimal criterion. For each value of  $k$ , 15 different randomly generated designs are considered, resulting in clusters of size 15 for each value of  $k$ . As we increase  $k$ , the variance of the clusters decreases and the clusters tend towards the origin. This indicates that random designs with a large number of sensors behave similarly and are generally effective in reducing posterior uncertainty and producing a good quality MAP point.

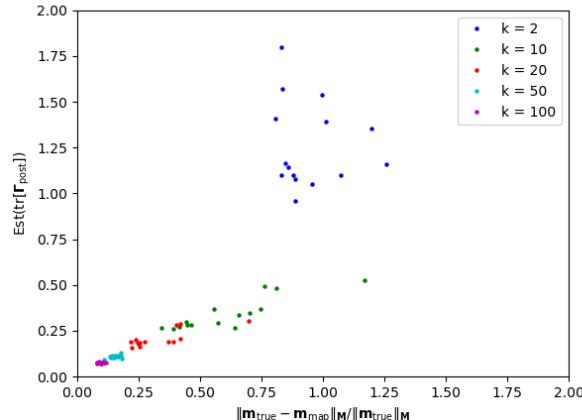


Fig. 4.3: Five clusters corresponding to different numbers of sensors. For example, each dot in the  $k = 2$  cluster corresponds to a random design consisting of 2 sensors. Using a fixed  $\mathbf{m}_{\text{true}}$ , synthetic data of length 2 is generated and the Bayesian inverse problem is solved. The mass weighted relative error between the MAP point and true parameter are then plotted against the posterior average variance.

The trend of the clusters in Figure 4.3 is generally linear. This is not a coincidence, but a consequence of the relationship between Bayes Risk and the posterior covariance. Take data  $\mathbf{y}$  and a Gaussian prior with law  $\mu_{\text{pr}} = \mathcal{N}(\mathbf{m}_{\text{pr}}, \boldsymbol{\Gamma}_{\text{pr}})$ . It can be shown that under the additional assumptions of a linear model and additive-Gaussian noise, the average mean square error between  $\mathbf{m}$  and  $\mathbf{m}_{\text{map}}^{\mathbf{y}}$  (Bayes Risk), is equal to the trace of the posterior

covariance [8, 2]. Specifically, if we denote Bayes Risk as  $\overline{\text{Risk}}(\mathbf{m}_{\text{map}}^y)$ , then

$$\begin{aligned}\overline{\text{Risk}}(\mathbf{m}_{\text{map}}^y) &:= \mathbb{E}_{\mu_{\text{pr}}} \mathbb{E}_{\mu_{\text{like}}} \{ \|\mathbf{m} - \mathbf{m}_{\text{map}}^y\|_2^2 \} \\ &= \int \int \|\mathbf{m} - \mathbf{m}_{\text{map}}^y\|_2^2 \pi_{\text{like}}(\mathbf{y}|\mathbf{m}) d\mathbf{y} \mu_{\text{pr}}(d\mathbf{m}) = \text{tr}[\mathbf{\Gamma}_{\text{post}}].\end{aligned}\quad (4.9)$$

We have simulated this theoretical expectation in a simple inverse problem governed by a one-dimensional heat equation; see Figure 4.4.

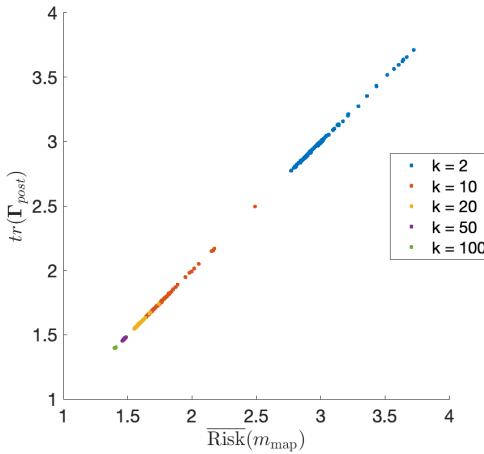


Fig. 4.4: Five clusters corresponding to different numbers of sensors.

In Figure 4.4, we see the theoretically expected result that the trace of the posterior covariance operator equals Bayes Risk. In this study, Bayes risk is computed via sampling: prior samples are used to generate noisy data required to solve the inverse problem, and the error in the MAP point is averaged over these samples. We see that the spread of the clusters corresponding to random designs of size  $k$  decrease as  $k$  grows. As in Figure 4.3, the clusters also tend towards the origin. This is yet another demonstration of the fact that when we have a large number of sensors, optimizing their placement is less consequential.

**5. Conclusions.** In our brief study we explored mathematical and computational aspects of solving Bayesian linear inverse problems and optimal sensor placement for these class of problems. To make the presentation accessible, we presented concepts in increasing level of complexity, starting from deterministic inversion and building up to formulation of an OED problem. The driving application for our studies was volume source inversion in an advection diffusion equation in a two-dimensional geometry. A key motivation for the present study was to demarcate computational techniques that allow for OED problem in the linear setting to be solved in a scalable way in an HPC setting.

Our most insightful computational results were in the case of the OED problem. We compared our optimal design against uniform and random designs of the same size, observing which design was most successful in reduction of average variance. It was seen that the optimal design performed better than the uniform design, which performed better than the random design. Obtaining optimal designs in the context of our problem lead to further discussion on the relationship between MAP estimation error and average variance reduction, in terms of the number of sensors used. We numerically demonstrated that as the size of the

design increases, the MAP point estimation error and average variance also decrease. This is intuitive and indicates that a practitioner may not need to solve an OED problem if sensor budget is not an issue. On the other hand, if only a few sensors are available, then optimal sensor placement is crucial. Additionally, we numerically demonstrated a theoretical result regarding equivalence of the standard A-optimality criterion and the Bayes Risk.

## REFERENCES

- [1] A. ALEXANDERIAN, *Optimal experimental design for infinite-dimensional Bayesian inverse problems governed by PDEs: a review*, Inverse Problems, 37 (2021), p. 043001.
- [2] A. ALEXANDERIAN, *On Bayes risk of the posterior mean in linear inverse problems*, 2023.
- [3] A. ALEXANDERIAN, P. J. GLOOR, AND O. GHATTAS, *On Bayesian A- and D-Optimal Experimental Designs in Infinite Dimensions*, Bayesian Analysis, 11 (2016), pp. 671 – 695.
- [4] A. ALEXANDERIAN, N. PETRA, G. STADLER, AND O. GHATTAS, *A-optimal design of experiments for infinite-dimensional Bayesian linear inverse problems with regularized  $\ell_0$ -sparsification*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2122–A2148.
- [5] ———, *A fast and scalable method for A-optimal design of experiments for infinite-dimensional Bayesian nonlinear inverse problems*, SIAM J. Sci. Comput., 38 (2016), pp. A243–A272.
- [6] M. S. ALNAES ET AL., *The fenics project version 1.5*, Archive of Numerical Software, 3 (2015).
- [7] T. BUI-THANH, O. GHATTAS, J. MARTIN, AND G. STADLER, *A computational framework for infinite-dimensional bayesian inverse problems part I: The linearized case, with application to global seismic inversion*, SIAM Journal on Scientific Computing, 35 (2013), pp. A2494–A2523.
- [8] K. CHALONER AND I. VERDINELLI, *Bayesian experimental design: A review*, Statistical science, (1995), pp. 273–304.
- [9] E. HABER, L. HORESH, AND L. TENORIO, *Numerical methods for the design of large-scale nonlinear discrete ill-posed inverse problems*, Inverse Problems, 26 (2010), p. 025002.
- [10] A. M. STUART, *Inverse problems: A Bayesian perspective*, Acta Numer., 19 (2010), pp. 451–559.
- [11] P. VIRTANEN, R. GOMMERS, T. E. OLIPHANT, M. HABERLAND, T. REDDY, D. COURNAPEAU, E. BUROVSKI, P. PETERSON, W. WECKESSER, J. BRIGHT, S. J. VAN DER WALT, M. BRETT, J. WILSON, K. J. MILLMAN, N. MAYOROV, A. R. J. NELSON, E. JONES, R. KERN, E. LARSON, C. J. CAREY, İ. POLAT, Y. FENG, E. W. MOORE, J. VANDERPLAS, D. LAXALDE, J. PERKTOLD, R. CIMRMAN, I. HENRIKSEN, E. A. QUINTERO, C. R. HARRIS, A. M. ARCHIBALD, A. H. RIBEIRO, F. PEDREGOSA, P. VAN MULBREGT, AND SCIPY 1.0 CONTRIBUTORS, *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*, Nature Methods, 17 (2020), pp. 261–272.
- [12] K. WU, P. CHEN, AND O. GHATTAS, *A fast and scalable computational framework for large-scale high-dimensional Bayesian optimal experimental design*, SIAM/ASA Journal on Uncertainty Quantification, 11 (2023), pp. 235–261.
- [13] C. ZHU, R. H. BYRD, P. LU, AND J. NOCEDAL, *Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization*, ACM Trans. Math. Softw., 23 (1997), p. 550–560.

## GROUPED SENSITIVITY ANALYSIS FOR GEOLOGIC DISPOSAL SAFETY ASSESSMENT

HALEY ROSSO\*, TERESA PORTONE†, AND REBEKAH WHITE‡

**Abstract.** The Geologic Disposal Safety Assessment (GDSA) Framework is a software toolkit developed to provide a modeling capability for geologic nuclear waste disposal systems (nuclear waste repositories) and capabilities to probabilistically assess the performance of disposal options and generic sites. A key activity in such probabilistic assessments is sensitivity analysis, which can help guide future model development and uncertainty reduction efforts by identifying the most important sources of uncertainty in the modeled systems. This work considers a generic repository reference case known as the Crystalline Reference Case. We use variance-based sensitivity analysis (VBSA) to quantitatively identify the most important sources of uncertainty to repository performance quantities of interest. Novel in this work is the application of grouped global sensitivity analysis, with sources of uncertainty grouped phenomenologically. We demonstrate how grouped sensitivity analysis can increase interpretability of results and provide computational advantages relative to standard practice in VBSA.

**1. Introduction.** Nuclear waste, such as spent nuclear fuel, is disposed of in deep geologic repositories. Ensuring performance of these repositories is critical for preventing aquifer contamination and human exposure to hazardous radionuclides. The Geologic Disposal Safety Assessment (GDSA) Framework is a software toolkit developed to provide a modeling capability for geologic nuclear waste disposal systems (nuclear waste repositories) and capabilities to probabilistically assess the performance of disposal options and generic sites [26]. Our work focuses on a generic repository reference case, referred to as the Crystalline Reference Case (CRC) within the GDSA framework. The CRC models a generic repository for commercial spent nuclear fuel in fractured crystalline rock with a spatial domain of  $3000 \times 2000 \times 1260$  meters and 4.8 million grid cells. Figure 4.1 illustrates the crystalline rock along with parameters such as repository depth and physical features of the repository. A key capability in evaluating the CRC is sensitivity analysis (SA) which identifies the most influential model components, helping direct model development and the investment of resources to better resolve model uncertainties [35].

Variance-based sensitivity analysis (VBSA) is widely used to measure the impact of model input uncertainties on predictions through the computation of Sobol' indices [17, 27, 29]. Sobol' indices represent how much each model input contributes to the total variance in the model prediction. The larger a Sobol' index, the more influential a model input is deemed. Grouped Sobol' indices reflect the influence of a *group* of model inputs rather than individual inputs and are defined analogously to the standard indices, see [28]. This reduces what could be tens to hundreds of sources of uncertainty down to a handful of phenomenologically meaningful groups, increasing interpretability of the results. Monte Carlo (MC) estimators for grouped indices are given in [17] and are clear extensions of the classical approaches presented in [20]. When the number of model inputs far exceeds the number of relevant groups, computing grouped indices is computationally more efficient than individual Sobol' indices. However, the computational burden can still be prohibitive for expensive forward models as it requires  $N(g + 2)$  model evaluations, where  $N$  is the number of MC samples and  $g$  the number of groups. To overcome this, a common approach is to utilize surrogate models for the computation of the indices [6, 32, 31].

In this work we apply grouped sensitivity analysis to repository performance QoIs (e.g., radionuclide concentration as a function of time) for the CRC. We consider seven uncertain

---

\*Emory University, haley.rosso@emory.edu

†Sandia National Laboratories, tporton@sandia.gov

‡Sandia National Laboratories, rebwhit@sandia.gov

parameters grouped phenomenologically into three categories: glacial aquifer properties, repository region properties, and radionuclide source parameters. To alleviate the computational costs associated with estimating Sobol' indices, we use Gaussian process (GP) and polynomial chaos expansion (PCE) surrogate models to represent the QoIs as functions of the uncertain parameters.

The key contributions of this work include

- Demonstration of grouped sensitivity indices for a sensitivity analysis of a nuclear waste repository application. This has not been previously demonstrated to our knowledge.
- Establishing accurate and efficient GP and PCE surrogate models of the repository performance QoIs.
- Determining through the computation of main effect individual Sobol' indices that all QoIs, apart from peak  $^{129}\text{I}$ , have sensitivities dominated by only one model input.
- Identifying that the peak iodine concentration in the glacial aquifer region (peak  $^{129}\text{I}$ ) is sensitive to aquifer properties and the radionuclide source and that interaction effects among model inputs make a small contribution to uncertainty in the prediction of peak  $^{129}\text{I}$ .

The paper is outlined as follows. Section 2 provides a review of Sobol' indices, presenting the grouped main and total effect grouped index formulations. The Monte Carlo estimators for the grouped indices are given in 3. Section 4 provides background on the PFLOTRAN model and the CRC. Here, relevant inputs and QoIs considered in the sensitivity study are provided. A discussion of surrogate models trained using high-fidelity PFLOTRAN is presented in Section 5. Computational results are provided in Section 6, with concluding remarks given in Section 7.

**2. Grouped Sobol' Indices.** We first review classical Sobol' index computation. Let  $\mathbf{X} = (X_1, X_2, \dots, X_n)$  be the model inputs represented as random variables and assumed to be independent. Here,  $X_i$  has support  $\Omega_i \subset \mathbb{R}$ , for  $i = 1, \dots, n$ . Then, consider  $f \in \mathcal{L}^2$  where  $f : \Omega \rightarrow \mathbb{R}$  for  $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_n$ . Let  $\mathbf{u} = \{i_1, i_2, \dots, i_k\} \subset \{1, 2, \dots, n\}$  and  $\mathbf{X}_{\mathbf{u}} = (X_{i_1}, X_{i_2}, \dots, X_{i_k})$ , representing a subset of input variables. Then,  $f(\mathbf{X})$  admits the following analysis of variance (ANOVA) decomposition [17, 27, 29]

$$f(\mathbf{X}) = f_{\emptyset} + \sum_{i=1}^n f_i(X_i) + \sum_{i=1}^n \sum_{j=i+1}^n f_{i,j}(X_i, X_j) + \dots + f_{1,2,\dots,n}(\mathbf{X}), \quad (2.1)$$

where

$$\begin{aligned} f_{\emptyset} &= \mathbb{E}[f(\mathbf{X})] \\ f_i(X_i) &= \mathbb{E}_{\mathbf{X}_{\sim i}}[f(\mathbf{X})|X_i] - f_{\emptyset} \\ f_{i,j}(X_i, X_j) &= \mathbb{E}_{\mathbf{X}_{\sim i,j}}[f(\mathbf{X})|X_i, X_j] - f_i(X_i) - f_j(X_j) - f_{\emptyset} \\ &\vdots \\ f_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}) &= \mathbb{E}_{\mathbf{X}_{\sim \mathbf{u}}}[f(\mathbf{X})|\mathbf{X}_{\mathbf{u}}] - \sum_{\mathbf{v} \in \mathcal{P}(\mathbf{u})} f_{\mathbf{v}}(\mathbf{X}_{\mathbf{v}}), \end{aligned}$$

where  $\mathbf{X}_{i,j} = \{X_i, X_j\}$ , and  $\mathbf{X}_{\sim k}$  denotes  $\mathbf{X}_{\{1,2,\dots,n\} \setminus \{k\}} = (X_1, X_2, \dots, X_{k-1}, X_{k+1}, \dots, X_n)$ , and  $\mathcal{P}(\mathbf{u})$  refers to the powerseries of  $\mathbf{u}$ . Similarly,  $\mathbf{X}_{\sim i,j} = \{X_{\sim i}, X_{\sim j}\}$ , and the definition for  $\mathbf{X}_{\sim \mathbf{u}}$  is analogously defined.

Due to independence of the model inputs, elements of the decomposition are orthogonal, e.g.,

$$\mathbb{E}[f_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}) f_{\mathbf{v}}(\mathbf{X}_{\mathbf{v}})] = 0, \quad \text{for all } \mathbf{u} \neq \mathbf{v}. \quad (2.2)$$

Therefore, we can decompose the variance in  $f$  into contributions from the subsets of the inputs by computing the variance of (2.1) and applying (2.2). Thus,

$$\text{Var}(f(\mathbf{X})) = \sum_{i=1}^n \text{Var}_{X_i}(f_i(X_i)) + \sum_{i=1}^n \sum_{j=i+1}^n \text{Var}_{\mathbf{X}_{i,j}}(f_{i,j}(X_i, X_j)) + \cdots + \text{Var}_{\mathbf{X}}(f_{1,2,\dots,n}(\mathbf{X})).$$

The  $\iota$ -th order Sobol' index for the variable  $\mathbf{X}_u$  is defined as

$$S_u = \frac{\text{Var}_{\mathbf{X}_u}(f_u(\mathbf{X}_u))}{\text{Var}(f(\mathbf{X}))} = \frac{\text{Var}_{\mathbf{X}_u}\left(\mathbb{E}_{\mathbf{X}_{\sim u}}[f(\mathbf{X})|\mathbf{X}_u] - \sum_{v \in \mathcal{P}(u)} f_v(\mathbf{X}_v)\right)}{\text{Var}(f(\mathbf{X}))}, \quad (2.3)$$

where  $\iota = |\mathbf{u}|$ . This index can be interpreted as the relative contribution of  $\mathbf{X}_u$  to the variance of  $f(\mathbf{X})$ . For  $k \in \{1, 2, \dots, n\}$ , we can write the first-order Sobol' index probabilistically as

$$S_k = \frac{\text{Var}_{X_k}(\mathbb{E}_{\mathbf{X}_{\sim k}}[f(\mathbf{X})|X_k])}{\text{Var}(f(\mathbf{X}))}, \quad (2.4)$$

since the  $\text{Var}(f_0) = 0$ .

The *first-order Sobol' indices* do not account for interactions with other inputs. In contrast, the *total Sobol' indices* are defined by [17] as

$$T_k = \sum_{\substack{\mathbf{u} \subset \{1, \dots, n\} \\ k \in \mathbf{u}}} S_{\mathbf{u}}, \quad k = 1, 2, \dots, n, \quad (2.5)$$

which accounts for the contributions of  $X_i$  through itself and interactions with other variables. As with first-order indices, we can write the total Sobol' index probabilistically as

$$T_k = \frac{\mathbb{E}_{\mathbf{X}_{\sim k}}[\text{Var}_{X_k}(f(\mathbf{X})|\mathbf{X}_{\sim k})]}{\text{Var}(f(\mathbf{X}))}. \quad (2.6)$$

We can apply straightforward extensions of first-order and total Sobol' indices to groups of inputs. Recall,  $\mathbf{u} = \{i_1, i_2, \dots, i_k\} \subset \{1, \dots, n\}$ . Inputs  $\{X_k\}_{k=i_1}^{i_k}$  are not required to be independent, and therefore we cannot determine the contribution of an input  $X_k$  to the variance of  $f$ . However, one can always compute the contribution of the group  $\mathbf{X}_u$ . The *grouped first-order Sobol' index* is defined as

$$S_u^g = \frac{\text{Var}_{\mathbf{X}_u}(\mathbb{E}_{\mathbf{X}_{\sim u}}[f(\mathbf{X})|\mathbf{X}_u])}{\text{Var}(f(\mathbf{X}))}, \quad \mathbf{u} \subset \{1, 2, \dots, n\}. \quad (2.7)$$

Note, the grouped Sobol' index formulation given in (2.7) also referred to as the closed Sobol' index in literature [17].

Analogous to the total Sobol' index for an individual input, we can define a *grouped total Sobol' index* as

$$T_u = \frac{\mathbb{E}_{\mathbf{X}_{\sim u}}[\text{Var}_{\mathbf{X}_u}(f(\mathbf{X})|\mathbf{X}_{\sim u})]}{\text{Var}(f(\mathbf{X}))}. \quad (2.8)$$

**3. Computing Grouped Sobol' Indices.** The probabilistic representations of the grouped Sobol' indices given in (2.7) and (2.8) allow one to derive Monte Carlo (MC) estimators for practical computation. There exist a variety of ways to compute Sobol' indices, but we focus on a Monte Carlo (MC) integration scheme presented in [17].

We utilize the *pick-and-freeze* method to estimate the Sobol' indices. To do so,  $N$  samples of the model inputs are generated through randomly sampling of their assumed distribution. One then defines sample matrices  $\mathbf{A}$  and  $\mathbf{B}$ , where each column represents one sample of model inputs  $X \in \mathbb{R}^n$ ; the columns represent the  $N$  input samples. Let  $\mathbf{A}_B^{(u)}$  be an exact copy of  $\mathbf{A}$  except the  $j$ -th column (for all  $j \in \mathbf{u}$ ) is drawn from  $\mathbf{B}$ . Analogously,  $\mathbf{B}_A^{(u)}$  is an exact copy of  $\mathbf{B}$  except the  $j$ -th column, for all  $j \in \mathbf{u}$ , is drawn from  $\mathbf{A}$ . Note that the construction of relevant matrices (e.g.  $\mathbf{A}$  and  $\mathbf{A}_B^{(u)}$ ) differs little for grouped versus individual inputs; grouped computation requires tracking of  $j$  varied columns rather than one varied column. Figure 3.1 demonstrates the construction of matrices  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{B}_A^{(u)}$  for an  $n = 3$  case. Here, model parameters are given by  $\theta = [a, b, c] \in \mathbb{R}^3$  and  $\mathbf{u} = \{1, 2\}$ .

With the matrices  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{A}_B^{(u)}$ , and  $\mathbf{B}_A^{(u)}$  we can estimate the numerator of (2.7) as

$$\text{Var}_{\mathbf{X}_{\mathbf{u}}} (\mathbb{E}_{\mathbf{X}_{\sim \mathbf{u}}} [f(\mathbf{X}) | \mathbf{X}_{\mathbf{u}}]) \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{B})_i \left( f(\mathbf{A}_B^{(u)})_i - f(\mathbf{A})_i \right). \quad (3.1)$$

Similarly, one can define MC estimators for the grouped total Sobol' index given in (2.8) as

$$\mathbb{E}_{\mathbf{X}_{\sim \mathbf{u}}} [\text{Var}_{\mathbf{X}_{\mathbf{u}}} (f(\mathbf{X}) | \mathbf{X}_{\sim \mathbf{u}})] \approx \frac{1}{2N} \sum_{i=1}^N \left( f(\mathbf{B})_i - f(\mathbf{B}_A^{(u)})_i \right)^2. \quad (3.2)$$

Finally, we can compute the denominator of the first-order and total Sobol' indices by estimating the total variance of  $f(\mathbf{X})$  as

$$\text{Var}(f(\mathbf{X})) \approx \frac{1}{2N} \sum_{i=1}^N \left[ \left( f(\mathbf{A})^{(i)} - \mu(\mathbf{A}) \right)^2 + \left( f(\mathbf{B})^{(i)} - \mu(\mathbf{B}) \right)^2 \right], \quad (3.3)$$

where  $\mu(\cdot) = \frac{1}{N} \sum_{i=1}^N f(\cdot)^{(i)}$ .

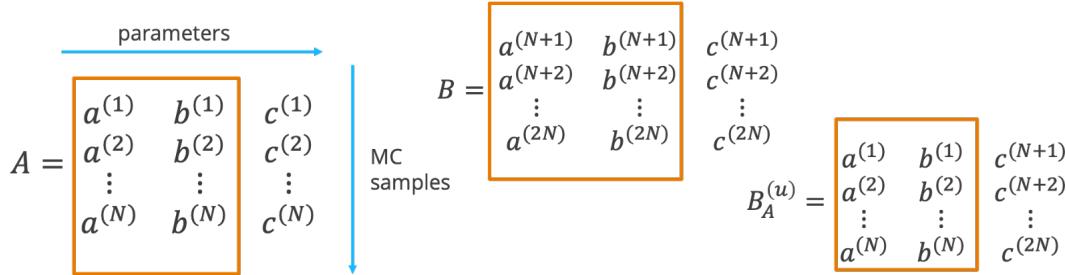


Fig. 3.1: Visual demonstration of pick and freeze method for  $n = 3$ . Notice that matrix  $\mathbf{B}_A^{(u)}$  contains  $\mathbf{u} = \{1, 2\}$  columns from matrix  $\mathbf{A}$ , while the remaining columns are drawn from matrix  $\mathbf{B}$ .

**4. PLOTTRAN Model Background.** PFLOTRAN, the modeling capability in the GDSA framework, is an open source, state-of-the-art, massively parallel subsurface flow and reactive transport simulator written in object-oriented Fortran 2003 [11, 12, 4, 5]. PFLOTRAN models subsurface flow using a porous medium continuum approach, which includes

capabilities for multicomponent systems, multiphase flow and transport, heat conduction and convection, biogeochemical reactions, geomechanics, and radionuclide decay and ingrowth. The focus of this study is the Crystalline Reference Case (CRC), a generic model of a repository for commercial spent nuclear fuel in fractured crystalline rock. The CRC is described in detail in [30, 14, 24, 34]. Studies utilizing the CRC for uncertainty and sensitivity analysis research and development have been presented in [38, 37, 36, 34, 1].

The Fuel Matrix Degradation (FMD) model, an advanced fuel degradation model, was recently implemented in PFLOTTRAN, as detailed in [2]. In [34], the FMD model was compared to another fuel degradation model, but it was not incorporated into a sensitivity analysis. In this work, we extend the previous sensitivity analyses performed for the CRC to include uncertainties from the FMD model. We consider eight repository performance assessment (PA) QoIs, defined in Table 4.1. A diagram of the CRC with regions of interest to the QoIs is presented in Figure 4.1.

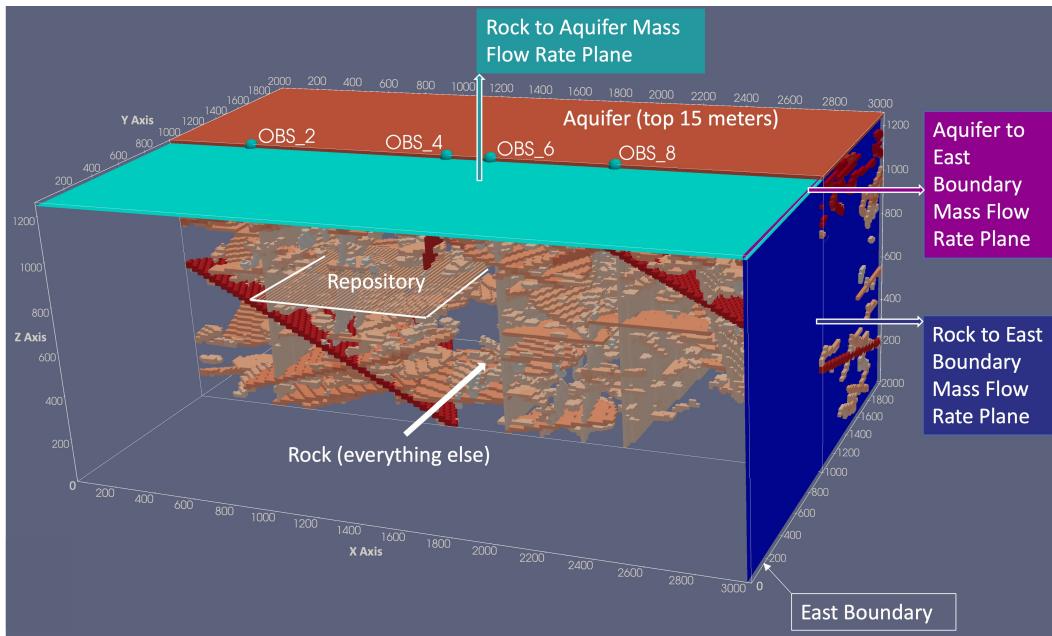


Fig. 4.1: A diagram of the PFLOTTRAN crystalline reference case (CRC) with integral flux planes and depth zones visualized within the model domain. The CRC models a generic repository for commercial spent nuclear fuel in fractured crystalline rock with a spatial domain of  $3000 \times 2000 \times 1260$  meters and 4.8 million grid cells.

In this problem, seven uncertain parameters are considered, whose names, descriptions, and sampling distributions can be found in Table 4.2. The parameters *kGlacial*, *pBuffer*, *meanWPrate*, and *IRF* are parameters that have been considered in previous sensitivity analyses for the CRC, and their distributions are identical to those used in [35]. The parameters *environmentalCO*, *environmentalHtwo*, and *burnup* are uncertain inputs to the FMD model. Their distributions are drawn from [2].

In this work, we will compare the individual parameter GSA results to a grouped GSA, where parameters associated with the radionuclide source are grouped together phenomenologically. The repository region and aquifer region groups only have one parameter each, so

Quantity of Interest (QoI)	Description
Peak $^{129}\text{I}$ concentration in aquifer in mol/L (M)	Scalar with (X,Y,Z) location of peak. This is an indicator of repository performance.
Median Residence Time MdRT of Spike in Repository	Median residence of a tracer in the repository. This is the time when half the tracer remains in the repository. It is an indicator of repository retention
Fractional Mass Flux of Tracer from Repository at 3000 yr (1/yr)	The instantaneous fractional loss rate of tracer remaining in repository at 3000 years. It is an indicator of repository retention.
Fractional Mass Flux of Tracer from Repository at 1 million yr (1/yr)	The fraction of a tracer remaining in repository at 1 million years. It is an indicator of repository retention.
Aquifer to East / Rock to East at 3000 years	This is the ratio of two water fluxes: the flux from the aquifer to the east boundary normalized by the flux from the rock to the east boundary. It indicates the multiplication factor on aquifer dominance of East side effluent at 3000yr during thermal pulse.
Aquifer to East / Rock to East at 1 million years	This is the ratio of two water fluxes: the flux from the aquifer to the east boundary normalized by the flux from the rock to the east boundary. It indicates the multiplication factor on aquifer dominance of East side effluent at 1 Myr near undisturbed conditions.
Rock to Aquifer / Rock to East at 3000 years	This is the ratio of two water fluxes: the rock to the aquifer vs. the rock to the east boundary at 3000 years. It indicates a multiplication factor on upward vs horizontal flow at 3000 years during the thermal pulse.
Rock to Aquifer / Rock to East at 1 million years	This is the ratio of two water fluxes: the rock to the aquifer vs. the rock to the east boundary at 1 million years. It indicates a multiplication factor on upward vs horizontal flow at 1 M years in near undisturbed conditions.

Table 4.1: The eight repository performance QoIs and their corresponding descriptions taken from [35].

their sensitivity measures will be identical to the individual parameter results.

**5. Surrogate Models.** The number of forward model evaluations necessary to compute Sobol' indices is often greater than  $\mathcal{O}(10^3)$ , making computation prohibitive for large-scale multi-physics codes such as PFLOTTRAN. To overcome this, we construct efficient

Parameter Group	Parameter	Description	Distribution
Glacial aquifer properties	$k_{Glacial}$	Glacial aquifer permeability [ $m^2$ ]	$\log \mathcal{U}[10^{-15}, 10^{-13}]$
Repository region properties	$pBuffer$	Buffer permeability [ $m^2$ ]	$\log \mathcal{U}[10^{-20}, 10^{-17}]$
Radionuclide source	$IRF$ $meanWPrate$ $environmentalCO$ $environmentalHtwo$ $burnup$	buffer porosity [-] General corrosion rate truncated log-Normal distribution mean [ $\log(yr^{-1})$ ] Environmental concentration of $\text{CO}_3^{2-}$ [mol/liter] Environmental concentration of $\text{H}^2$ [mol/liter] Fuel burnup [GWd/MTHM]	$\mathcal{U}[0.038, 0.156]$ $\mathcal{U}[-5.5, -4.5]$ $\log \mathcal{U}[10^{-6}, 2 * 10^{-5}]$ $\log \mathcal{U}[10^{-10}, 10^{-8}]$ $\mathcal{U}[40, 65]$

Table 4.2: The seven uncertain parameters representing the FMD model.

Gaussian process (GP) and polynomial chaos expansion (PCE) surrogate model representations of the eight QoIs. While many possible surrogate model representations exist, GPs and PCEs were chosen as they are widely used in Sobol' index computation [32, 3, 25, 15] and have properties conducive to modeling PFLOTTRAN QoIs. Training these surrogate models requires significantly fewer expensive high-fidelity model evaluations than those required to compute Sobol' index estimators. With these surrogates, we can sample the approximated QoIs at a negligible cost to compute the Sobol' indices [19].

First, let's review Gaussian process models. GPs are commonly used for regression in lower dimensions ( $d < 3$ ) and are popular in fields such as geostatistics and meteorology [8, 16, 33, 10]. Let  $f(\mathbf{x}) : \Omega \rightarrow \mathbb{R}$ , where  $\Omega$  is the model input space. We then assume that the function is distributed as a GP. That is

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) ,$$

where  $m : \mathbb{R}^d \rightarrow \mathbb{R}$  represents the *mean* function and  $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  the *covariance* function [22]. In essence, GPs are distributions over functions. Gaussian processes make convenient surrogate models as they are interpolatory, provide smooth approximations of functions, and naturally yield uncertainty estimations for model predictions.

In practice there exist many potential covariance functions one can choose, where the choice is guided by properties of the underlying function one is trying to estimate. Here, we consider the squared exponential (i.e., Gaussian) correlation function. Thus,

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp \left( \sum_{m=1}^d \frac{(x_m - x'_m)^2}{l_m^2} \right) , \quad (5.1)$$

where  $d$  is the dimension of the input space,  $l_m$  is the correlation lengths for input dimension  $m$ , and  $\sigma^2$  is a scaling term that affects the range of the output space [21]; such hyperparameters are estimated as part of the fitting procedure. This covariance function is chosen as it is infinitely differentiable, meaning square derivatives of all orders exist, and the surrogate is smooth [18]. This choice aligns well with the properties of the high-fidelity PFLOTTRAN QoIs we are modeling.

Next, let's consider polynomial chaos expansions (PCE). PCEs are a widely used type of polynomial approximation method [9, 39, 32]. They allow one to represent a model parameterized by random variables through an easy-to-evaluate polynomial function. PCE is considered a non-sampling method. In other words, PCE utilizes polynomials to represent parameter uncertainty, rather than randomly sampling the distributions, which is particularly advantageous in cases of large models where sampling the entire input space can be computationally intractable. Hence, uncertainties are represented as an expansion of random orthogonal basis polynomials and their coefficients, which are computed deterministically [23].

Polynomial chaos expansions represent the model inputs  $\mathbf{X}$  as random variables. Thus, the model output,  $f(\mathbf{X})$ , is also a random variable with assumed finite variance. The PCE represents the model response as

$$f(\mathbf{x}) = \sum_{n=1}^{\infty} \beta_n \psi_n(\mathbf{x}),$$

where  $\mathbf{x}$  represents realizations of the random variable  $\mathbf{X}$ ,  $\beta_n$  are the PCE coefficients, and  $\psi_n$  are the polynomial basis functions. The basis functions are required to be pairwise orthonormal, i.e.,

$$\begin{aligned} \langle \psi_i, \psi_j \rangle_w &= 0, \quad \text{for } i \neq j, \\ \langle \psi_i, \psi_j \rangle_w &= 1, \quad \text{for } i = j, \end{aligned}$$

where  $w$  weights the inner product based on the probability density of  $\mathbf{X}$ . The expansion coefficients  $\beta_n$  are computed via least-squares regression. [13].

**6. Computational Results.** This section details the computational results of analyzing the sensitivity of repository performance QoIs to the model components. This application requires  $\mathcal{O}(10^6)$  model evaluations (MC samples) to see convergence in the MC estimators of the indices. This is intractable for the high-fidelity multiphysics reactive transport code PFLOTRAN [11]. Thus, we instead generate 400 high-fidelity samples of the eight QoIs presented in Table 4.1 and use the samples to build surrogate model approximations of each of the QoIs. For the PCE surrogate model, a polynomial degree of  $N = 2$  is used. When constructing the Gaussian process surrogate, the covariance function given in (5.1) is augmented with white noise term often referred to as “jitter,” which serves to stabilize the Cholesky factorization used in the fitting procedure. To generate the high-fidelity data, the PFLOTRAN inputs (see Table 4.2) are sampled using a Latin Hypercube Sampling strategy [7].

Section 6.1 presents a comparison of surrogate modeling approaches. Here, we compute the individual main and total effect Sobol' indices for both the PCE and GP surrogate models. We then compare the results. In Section 6.2 we investigate the accuracy of the GP surrogate model through  $k$ -fold cross validation tests and comparisons of surrogate predictions to high-fidelity data for not included as training data. Finally, Section 6.3 presents the results of performing grouped GSA for the QoI peak  $^{129}\text{I}$ . Here, we compare the main and total effect grouped indices.

**6.1. Comparison of Gaussian Process versus Polynomial Chaos Expansions for computing Sobol' Indices.** To understand how the choice of surrogate model affects Sobol' index computation for this application, we use Dakota [21] to build GP and PCE surrogates for each of the eight QoIs using 400 high-fidelity PFLOTRAN samples. We then compute the individual main and total effect Sobol' indices for each of the QoIs and verify if

the results align. Note that when analyzing the input-output relationships of the high-fidelity PFLOTTRAN data, non-differentiable behavior is observed due the relative magnitude of model inputs and QoIs. For this reason, we constructed surrogate models and performed sensitivity analysis with log-transformed inputs *environmentalHtwo*, *environmentalCO*, and *kGlacial*, and QoI peak  $^{129}\text{I}$ . This results in more accurate surrogate models. Figure 6.1 provides representative results of the Sobol' index computation for peak  $^{129}\text{I}$ .

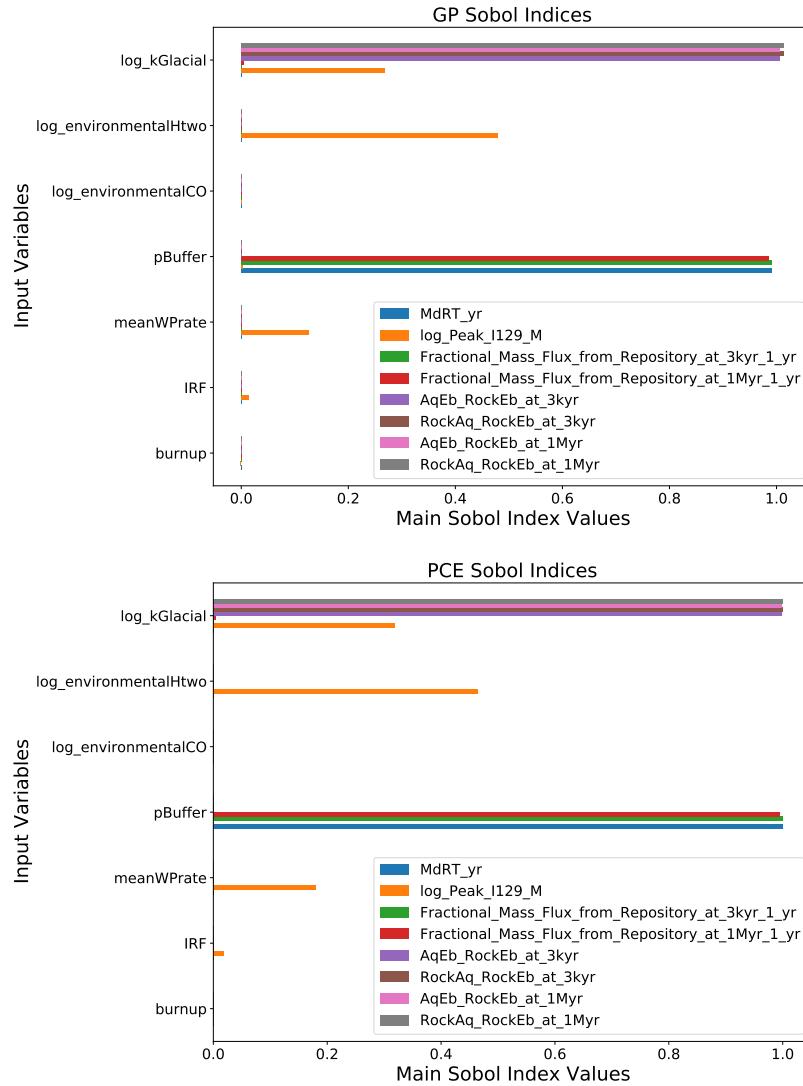


Fig. 6.1: Main effect individual Sobol' index values for the eight QoIs using Gaussian process surrogates (top) versus polynomial chaos expansion surrogates (bottom).

From Figures 6.1 and 6.2 we see similar estimates of the main and total effect indices for both the GP and PCE surrogate models. Furthermore, we see that overall the most influential inputs are *kGlacial* and *pBuffer*, while *environmentalCO*, *IRF*, and *burnup* have

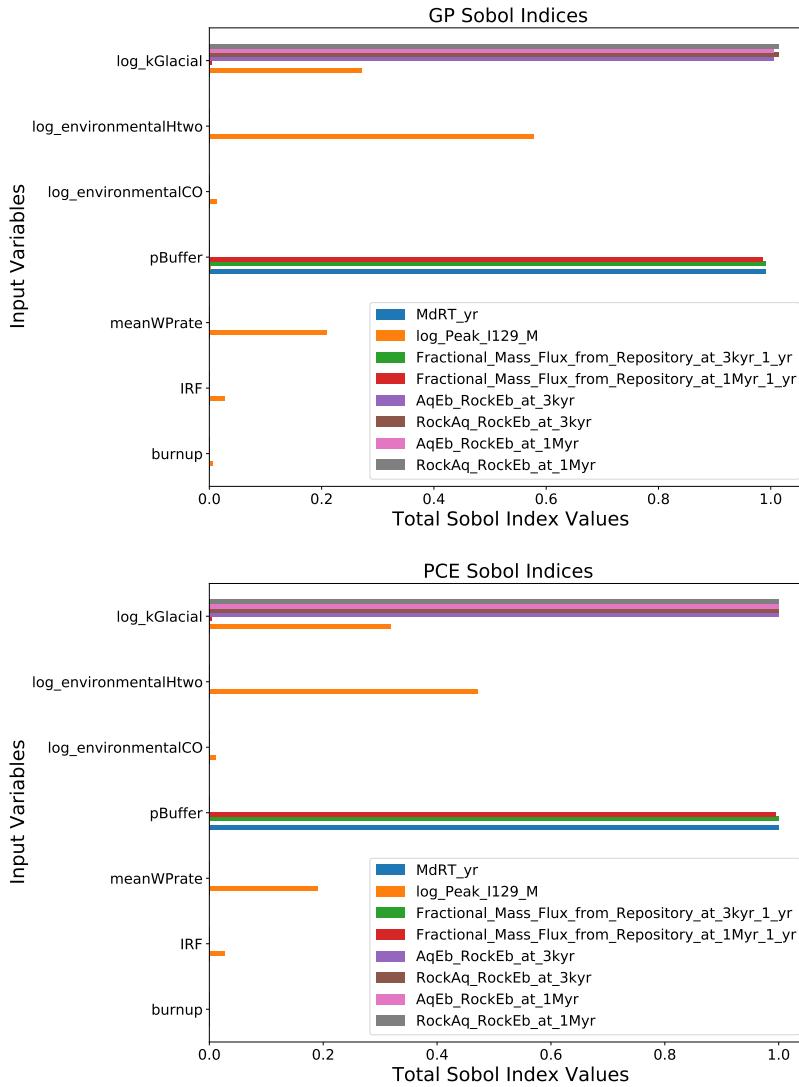


Fig. 6.2: Individual total effect Sobol' index values for the eight QoIs using Gaussian process surrogates (top) versus polynomial chaos expansion surrogates (bottom).

negligible effect on QoI predictions. Additionally, Figures 6.1 and 6.2 illustrate that peak  $^{129}\text{I}$  is sensitive to various model inputs, while the other QoIs are only sensitive to one input.

The plots in Figure 6.3 allow one to better see the differences for both the main and total effect individual indices for peak  $^{129}\text{I}$ .

Figure 6.3 depicts close correspondence between the surrogate modeling approaches. Furthermore, we see that peak  $^{129}\text{I}$  is most sensitive to  $\log(\text{environmentalHtwo})$ ,  $\log(k\text{Glacial})$ , and  $\text{meanWPrate}$ . These results indicate that the Sobol' index computation is not highly sensitive to the choice of GP or PCE surrogate model, and henceforth we provide computational results utilizing the GP surrogate model.

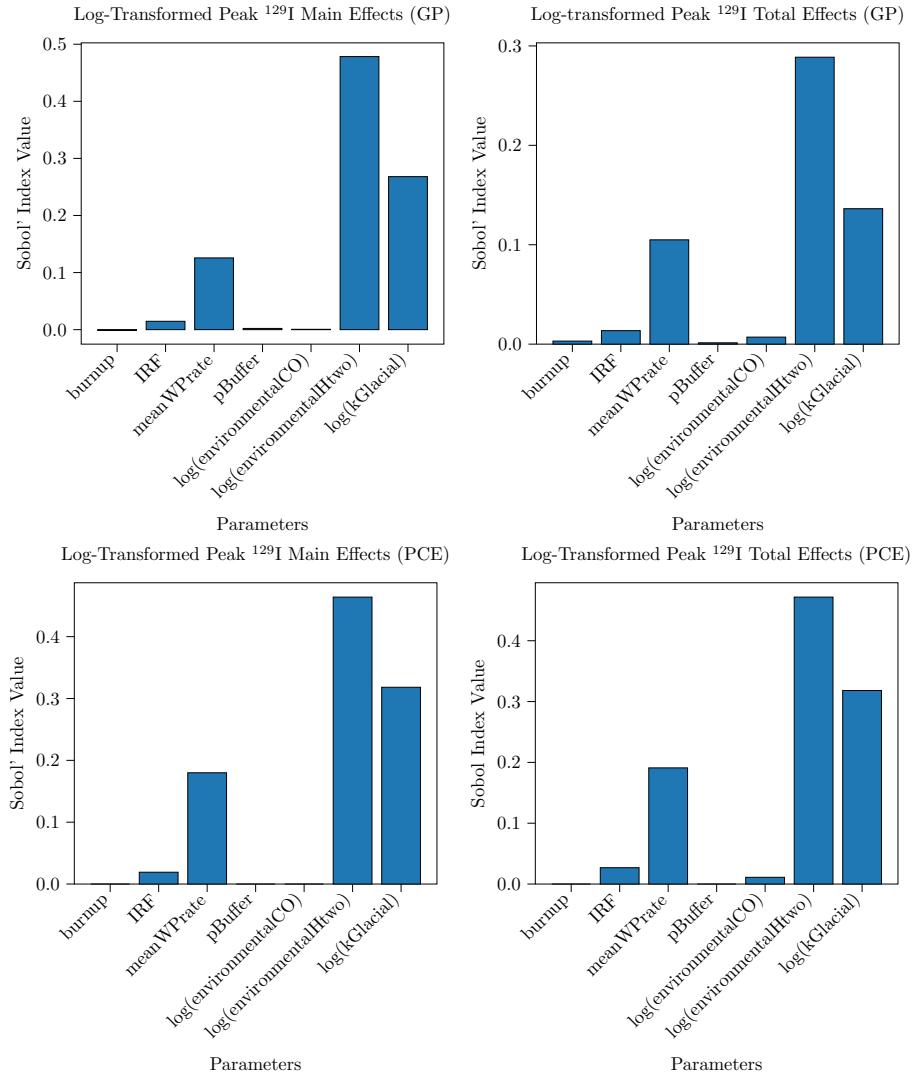


Fig. 6.3: The main and total individual Sobol' index values for peak  $^{129}\text{I}$  computed using the GP surrogate (top row) versus the PCE surrogate (bottom row).

## 6.2. Comparison of Gaussian process Surrogate versus PFLOTRAN Data.

To further validate the use of GP surrogates, we compare the GP predictions to the high-fidelity data. We do so in two ways. The first is to divide the 400 high-fidelity samples into training data (80%) and test data (20%). We then compare the GP predictions for each QoI to the high-fidelity test data. The plots in Figure 6.4 depict the comparison of surrogate model evaluations to test data ( $N = 40$  model evaluations) for peak  $^{129}\text{I}$ . From Figure 6.4 we see the GP predictions correspond well with the high-fidelity test data. Similar results were seen across all seven QoIs considered.

The second approach for evaluating the accuracy of the GP model is to compute  $k$ -fold cross validation metrics when training the GP model. Here, we again utilize Dakota.  $k$ -fold

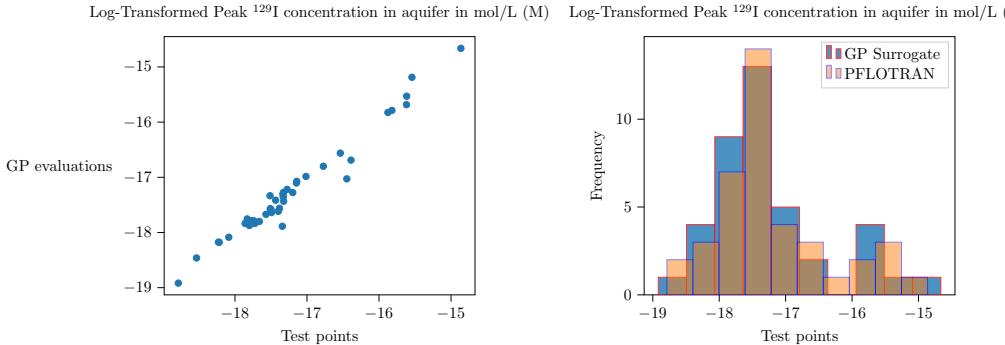


Fig. 6.4: A plot (left) and histogram with 10 bins (right) representing the 80 high-fidelity test points versus Gaussian process surrogate predictions on the test data of peak  $^{129}\text{I}$ .

cross validation is a hold-out method where the training data is subdivided into  $k$  folds. Then, all but one fold is used for training a GP; the remaining fold is used for testing. This process is repeated over all  $k$  folds, providing a measure of confidence in the use of surrogate model given a set of training data. The metrics considered are the root mean-squared error, the  $R$ -squared value (describing the proportion of the variability in the response that can be accounted for by the surrogate model), and the mean of the absolute value of the residuals. Computational results yielded mean-squared errors within acceptable tolerances relative to the magnitude of the QoIs being modeled, and the  $R$ -squared value no lower than 0.999734. These tests indicate that the GP surrogate models are adequately accurate for the purpose of Sobol' index computation.

**6.3. Evaluation of Grouped Sobol' indices for peak  $^{129}\text{I}$ .** We use the GP surrogate model to compute grouped indices for the parameter groups given in Table 4.2 representing  $k\text{Glacial}$ ,  $p\text{Buffer}$ , and the radionuclide source. Here, we present results for peak  $^{129}\text{I}$  as this QoI is sensitive to the largest number of model inputs (see Figure 6.1). Using the MC estimators (see Section 2), we compute the main and total effect grouped indices with  $(2 + g)N = 5e^7$  MC samples. This number is chosen as it provides convergence in Sobol' estimators over the random samples of model inputs. However, if one were primarily interested in screening the inputs for importance, significantly fewer MC samples could be used. Figure 6.5 shows the main and total effect grouped Sobol' indices for peak  $^{129}\text{I}$ .

From Figure 6.5 we see that the most influential model components are the radionuclide source and the glacial aquifer ( $k\text{Glacial}$ ). However, we see little impact from uncertainties associated with  $p\text{Buffer}$ . Additionally, the main and total effect indices are approximately the same, implying that interactions between these grouped inputs have a negligible effect on the prediction of peak  $^{129}\text{I}$  concentration. For this application, computing grouped Sobol' indices requires approximately half as many forward model evaluations as individual Sobol' indices, highlighting the computational advantages of grouped analysis. Furthermore, this grouped analysis' identification of aquifer properties and radionuclide source as the most important model components proves more interpretable than the individual analysis presented in Section 6.1 as analysts can increase the fidelity of these model components or reduce component uncertainty through experimentation, but may not be able to directly isolate the individual parameters.

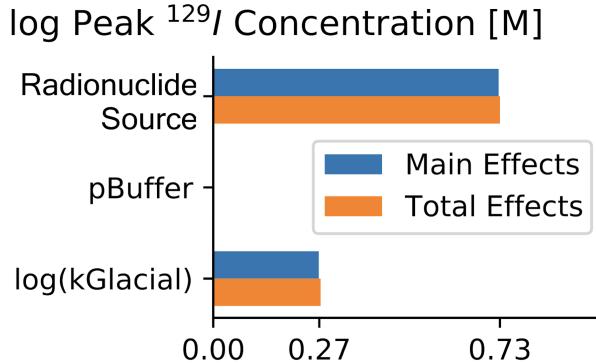


Fig. 6.5: Grouped main and total effect Sobol' index values for peak  $^{129}\text{I}$  computed using the GP surrogate.

**7. Conclusions.** This work was in support of the Geologic Disposal Safety Assessment (GDSA), a framework for modeling and analyzing potential future nuclear waste disposal sites. Our goal was to understand how uncertainties affect model predictions of nuclear waste repository performance. Understanding the most influential components when modeling nuclear waste degradation can guide future efforts to drive down uncertainty associated with highly-influential model parameters, components, or subsystems.

We considered a generic repository reference case known as the Crystalline Reference Case. Here, we focused on eight QoIs used to assess repository performance and seven model inputs grouped into representations of the glacial aquifer properties, the repository region properties, and the radionuclide source. To measure the impacts of these uncertain inputs on predictions of the performance QoIs, we used variance-based sensitivity analysis to compute main and total effect Sobol' indices. Since we are more interested in phenomenology, we computed grouped Sobol' indices providing the effects of model components rather than individual parameters. We found that the grouped SA approach increased interpretability of the results and offered computational advantages.

However, the high computational cost associated with running the PFLOTRAN model necessitated the use of surrogate models. We built Gaussian process (GP) and polynomial chaos expansion (PCE) surrogate models of each of the eight QoIs as a function of the seven model inputs. We trained these surrogates using 400 high-fidelity model evaluations and used  $k$ -fold cross validation testing to ensure accuracy of the resulting approximations. We first estimated individual main and total effect Sobol' indices using the PCE and GP surrogates in a Monte Carlo estimation of the indices. We found that both the PCE and GP surrogates produced similar estimates of the main and total effects, indicating our results are not strongly dependent upon the choice of surrogate model utilized. Thus, we considered only the GP surrogates for grouped sensitivity analysis.

In performing standard sensitivity analysis, we found that for all QoIs except the log-transformed peak iodine concentration ( $\log(\text{peak } ^{129}\text{I})$ ), sensitivities were dominated by a single model parameter. Thus, we focused the grouped analysis on  $\log(\text{peak } ^{129}\text{I})$ . Ultimately, we found that  $\log(\text{peak } ^{129}\text{I})$  was particularly sensitive to the radionuclide concentration and glacial aquifer permeability ( $k_{\text{Glacial}}$ ), indicating these as target areas for future work that will better resolve uncertainties in these parameter groups. Furthermore, we found that the main and total effects for the grouped indices are approximately equivalent, indicating that

interactions between the radionuclide source and *kGlacial* did not have a significant effect on the QoI. In future work we will consider multifidelity approaches to computing the grouped indices that will leverage combinations of the high-fidelity data and surrogate predictions, to reduce computational cost while maintaining accuracy.

## REFERENCES

- [1] D. BROOKS, L. SWILER, E. STEIN, P. MARINER, E. BASURTO, T. PORTONE, A. ECKERT, AND R. LEONE, *Sensitivity analysis of generic deep geologic repository with focus on spatial heterogeneity induced by stochastic fracture network generation*, Advances in Water Resources, 169 (2022), p. 104310.
- [2] B. J. DEBUSSCHERE, D. T. SEIDL, T. M. BERG, K. W. CHANG, R. C. LEONE, L. P. SWILER, AND P. E. MARINER, *Machine Learning Surrogates of a Fuel Matrix Degradation Process Model for Performance Assessment of a Nuclear Waste Repository*, Nuclear Technology, 209 (2023), pp. 1295–1318. Publisher: Taylor & Francis.
- [3] M. EHRE, I. PAPAIOANNOU, AND D. STRAUB, *Global sensitivity analysis in high dimensions with pls-pce*, Reliability Engineering & System Safety, 198 (2020), p. 106861.
- [4] G. HAMMOND, P. LICHTNER, R. MILLS, AND C. LU, *Toward petascale computing in geosciences: application to the hanford 300 area*, in Journal of Physics, Conference Series 125 012051, SciDAC 2008: Scientific Discovery through Advanced Computing, 2008.
- [5] G. E. HAMMOND, P. C. LICHTNER, AND R. MILLS, *Evaluating the performance of parallel subsurface simulators: An illustrative example with PFLOTRAN*, Water resources research, 50 (2014), pp. 208–228.
- [6] J. L. HART, A. ALEXANDERIAN, AND P. A. GREMAUD, *Efficient computation of sobol' indices for stochastic models*, SIAM Journal on Scientific Computing, 39 (2017), pp. A1514–A1530.
- [7] J. HELTON, *Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems*, tech. rep., Sandia National Laboratories, 2002.
- [8] R. T. W. L. HURKMAN, J. L. BAMBER, C. H. DAVIS, I. R. JOUGHIN, K. S. KHVOROSTOVSKY, B. S. SMITH, AND N. SCHOEN, *Time-evolving mass loss of the greenland ice sheet from satellite altimetry*, The Cryosphere, 8 (2014), pp. 1725–1740.
- [9] 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.
- [10] A. P. KYPRIOTI, A. A. TAFLANIDIS, N. C. NADAL-CARABALLO, AND M. O. CAMPBELL, *Incorporation of sea level rise in storm surge surrogate modeling*, Natural Hazards, 105 (2021), pp. 531–563.
- [11] P. LICHTNER AND G. HAMMOND, *Quick reference guide: PFLOTRAN 2.0 (la-cc-09-047) multiphase-multicomponent-massively parallel reactive transport code*, Tech. Rep. LA-UR-06-7048, Los Alamos National Laboratory, Los Alamos, New Mexico, 2012.
- [12] P. C. LICHTNER, G. E. HAMMOND, C. LU, S. KARRA, G. BISHT, B. ANDRE, R. MILLS, AND J. KUMAR, *PFLOTRAN user manual: A massively parallel reactive flow and transport model for describing surface and subsurface processes*, Tech. Rep. LA-UR-15-20403, Los Alamos National Laboratory (LANL), Los Alamos, NM; Sandia National Laboratories (SNL), Albuquerque, NM, 2015.
- [13] Z. LIU, D. LESSELIER, B. SUDRET, AND J. WIART, *Surrogate modeling based on resampled polynomial chaos expansions*, Reliability Engineering & System Safety, 202 (2020), p. 107008.
- [14] P. MARINER, E. STEIN, J. FREDERICK, S. SEVOUGIAN, G. HAMMOND, AND D. FASCITELLI, *Advances in geologic disposal system modeling and application to crystalline rock*, Tech. Rep. FCRD-UFD-2016-000440/SAND2016-9610 R, Sandia National Laboratories, Albuquerque, New Mexico, 2016.
- [15] A. MARREL, N. MARIE, AND M. DE LOZZO, *Advanced surrogate model and sensitivity analysis methods for sodium fast reactor accident assessment*, Reliability Engineering & System Safety, 138 (2015), pp. 232–241.
- [16] F. S. L. NG AND A. L. C. HUGHES, *Reconstructing ice-flow fields from streamlined subglacial bedforms: A kriging approach*, Earth Surface Processes and Landforms, 44 (2019), pp. 861–876.
- [17] C. PRIEUR AND S. TARANTOLA, *Variance-Based Sensitivity Analysis: Theory and Estimation Algorithms*, in Handbook of Uncertainty Quantification, R. Ghanem, D. Higdon, and H. Owhadi, eds., Springer International Publishing, Cham, 2017, pp. 1217–1239.
- [18] C. RASMUSSEN AND C. WILLIAMS, *Gaussian Processes for Machine Learning*, Massachusetts Institute of Technology, 2006.
- [19] S. RAZAVI, A. JAKEMAN, A. SALTELLI, C. PRIEUR, B. IOSS, E. BORGONOVO, E. PLISCHKE, S. LO PIANO, T. IWANAGA, W. BECKER, S. TARANTOLA, J. H. GUILLAUME, J. JAKEMAN, H. GUPTA, N. MELILLO, G. RABITTI, V. CHABRIDON, Q. DUAN, X. SUN, S. SMITH, R. SHEIKHOLESLAMI, N. HOSSEINI, M. ASADZADEH, A. PUY, S. KUCHERENKO, AND H. R. MAIER, *The Future of Sensitivity Analysis: An essential discipline for systems modeling and policy support*, Environmental

- Modelling & Software, 137 (2021), p. 104954.
- [20] A. SALTELLI, P. ANNONI, I. AZZINI, F. CAMPOLONGO, M. RATTO, AND S. TARANTOLA, *Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index*, Computer Physics Communications, 181 (2010), pp. 259–270. Number: 2.
- [21] SANDIA NATIONAL LABORATORIES, *Dakota*. <https://snl-dakota.github.io/docs/6.18.0/users/index.html>. Accessed: 2023-08-25.
- [22] E. SCHULZ, M. SPEEKENBRINK, AND A. KRAUSE, *A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions*, Journal of Mathematical Psychology, 85 (2018), pp. 1–16.
- [23] K. SEPAHVAND, S. MARBURG, AND H.-J. HARDTKE, *Uncertainty quantification in stochastic systems using polynomial chaos expansion*, International Journal of Applied Mechanics, 2 (2010), pp. 305 – 353. Cited by: 129.
- [24] S. D. SEVOUGIAN, E. R. STEIN, G. E. HAMMOND, P. E. MARINER, J. M. FREDERICK, AND E. BASURTO, *Simulating the effect of fracture connectivity on repository performance with gdsa framework – 18589.*, in Proceedings of WM2018 Conference, Phoenix, Arizona, 2018.
- [25] X. SHANG, L. SU, H. FANG, B. ZENG, AND Z. ZHANG, *An efficient multi-fidelity kriging surrogate model-based method for global sensitivity analysis*, Reliability Engineering & System Safety, 229 (2023), p. 108858.
- [26] SNL, *GDSA Framework: A Geologic Disposal Safety Assessment Modeling Capability.*, 2017.
- [27] I. SOBOL', *Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates*, The Second IMACS Seminar on Monte Carlo Methods, 55 (2001), pp. 271–280. Number: 1.
- [28] I. SOBOL' AND S. KUCHERENKO, *A new derivative based importance criterion for groups of variables and its link with the global sensitivity indices*, Computer Physics Communications, 181 (2010), pp. 1212–1217.
- [29] I. M. SOBOL', *Sensitivity Estimates for Nonlinear Mathematical Models*, Mathematical Modelling and Computational Experiments, 4 (1993), pp. 407–413.
- [30] E. STEIN, J. M. FREDERICK, G. E. HAMMOND, K. L. KUHLMAN, P. MARINER, AND S. D. SEVOUGIAN, *Modeling Coupled Reactive Flow Processes in Fractured Crystalline Rock.*, in Proceedings of the 16th International High-Level Radioactive Waste Management Conference, Charlotte, North Carolina, 2017.
- [31] C. STORLIE, L. SWILER, J. HELTON, AND C. SALLABERRY, *Implementation and evaluation of non-parametric regression procedures for sensitivity analysis of computationally demanding models*, Reliability Engineering & System Safety, 94 (2009), pp. 1735–1763.
- [32] B. SUDRET, *Global sensitivity analysis using polynomial chaos expansions*, Bayesian Networks in Dependability, 93 (2008), pp. 964–979. Number: 7.
- [33] J. SUSILUOTO, A. SPANTINI, H. HAARIO, T. HÄRKÖNEN, AND Y. MARZOUK, *Efficient multi-scale gaussian process regression for massive remote sensing data with satgp v0.1.2*, Geoscientific Model Development, 13 (2020), pp. 3439–3463.
- [34] L. SWILER, E. BASURTO, D. BROOKS, A. ECKERT, R. LEONE, P. MARINER, T. PORTONE, AND M. SMITH, *Uncertainty and sensitivity analysis methods and applications in the GDSA framework (fy2022)*, Tech. Rep. M3SF-23SN01030408, SAND2022-11220 R, Sandia National Laboratories, Albuquerque, New Mexico, 2022.
- [35] L. SWILER, E. BASURTO, D. BROOKS, A. ECKERT, R. LEONE, P. MARINER, T. PORTONE, M. SMITH, AND E. STEIN, *Uncertainty and Sensitivity Analysis Methods and Applications in the GDSA Framework (FY2021)*, Tech. Rep. M3SF- 21SN010304042, Sandia National Laboratories, 2021.
- [36] L. SWILER, E. BASURTO, D. BROOKS, A. ECKERT, R. LEONE, P. MARINER, T. PORTONE, M. SMITH, AND E. STEIN, *Uncertainty and sensitivity analysis methods and applications in the GDSA framework (fy2021)*, Tech. Rep. M3SF-21SN010304042, SAND2021-9903 R, Sandia National Laboratories, Albuquerque, New Mexico, 2021.
- [37] L. SWILER, E. BASURTO, D. BROOKS, A. ECKERT, P. MARINER, T. PORTONE, AND E. STEIN, *Advances in uncertainty and sensitivity analysis methods and applications in GDSA framework*, Tech. Rep. M3SF-20SN010304032, SAND2020-10802 R, Sandia National Laboratories, Albuquerque, New Mexico, 2020.
- [38] L. SWILER, J. HELTON, E. BASURTO, D. BROOKS, P. MARINER, L. MOORE, S. MOHANTY, S. SEVOUGIAN, AND E. STEIN, *Status report on uncertainty quantification and sensitivity analysis tools in the geologic disposal safety assessment (GDSA) framework*, Tech. Rep. M2SF-19SN010304031, SAND2019-13835R, Sandia National Laboratories, Albuquerque, New Mexico, 2019.
- [39] Y. ZHANG AND N. V. SAHINIDIS, *Uncertainty quantification in co2 sequestration using surrogate models from polynomial chaos expansion*, Industrial & Engineering Chemistry Research, 52 (2013), pp. 3121–3132.

## MAGNETOHYDRODYNAMICS: FORMULATION, OPTIMIZATION, AND UNCERTAINTY QUANTIFICATION

DANIEL SHARP\* AND BART G. VAN BLOEMEN WAANDERS†

### Abstract.

Magnetohydrodynamics (MHD) is generally understood as the behavior of magnetic fluids, such as liquid metals or dense plasmas, at an observable scale. The equations governing MHD are particularly difficult for simulations, as they are a generalization and extension of the well-known Navier-Stokes and Maxwell equations. This paper discusses certain advances in simulating magnetic fluids for the purpose of outer-loop analysis, such as inference, uncertainty quantification, and optimal control. We review several different formulations of the MHD problem, with some successful emulation and experimentation on a particular simplification of the physics.

**1. Introduction.** Science has historically wrestled with understanding the behavior of magnetic materials, with most progress coming from the past two centuries; magnetic fluids have proved all the more difficult. One of the first experiments testing properties later associated with magnetohydrodynamics (MHD), also known as magneto fluid dynamics, was by Michael Faraday in 1831 [6], who unsuccessfully tried to generate electricity using the flow of the river Thames. Since then, there have been experiments showing properties of magnetic fluids, and the properties today are applied to problems including casting molten metal [5], fusion energy generation, and stellar astrophysics [9]. However, many important questions that arise in physical simulation, such as optimal control and uncertainty quantification (UQ), often go unanswered in the realm of MHD as a result of insufficient instrumentation of optimization and UQ techniques for existing numerical models. Consequently, we are motivated to explore simulating MHD with the outlook of performing these “outer-loop” applications. At the heart of this goal lies the challenge of dealing with the highly nonlinear aspects of MHD, large number of optimization variables and the many sources of uncertainties. In this paper we explore a range of MHD formulations, evaluate the implementation in an optimization-enabled framework and make progress toward performing PDE-constrained optimization for realistic MHD problems.

### 2. Background.

**2.1. Physics.** A modern understanding of magnetic fluids starts with Maxwell’s equations, which were only formalized decades after Faraday’s experiments. Assuming the magnetic field, electric field, and electric current density vectors  $\mathbf{B}$ ,  $\mathbf{E}$ ,  $\mathbf{J}$ , respectively, the differential version of Maxwell’s equations can be stated as

$$\begin{aligned} \mu_0 \mathbf{J} &= \nabla \times \mathbf{B} - \frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t} && (\text{Amperè}) \\ \nabla \cdot \mathbf{B} &= 0 && (\text{Gauss}) \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} && (\text{Faraday}) \end{aligned}$$

where we have scalar parameters  $c$ , the speed of light, and  $\mu_0$ , free space permeability. This omits Gauss’s electric field law. We also recall two important relationships; the first, Ohm’s law, can be written as

$$\eta \mathbf{J} = \mathbf{E} + \mathbf{u} \times \mathbf{B},$$

assuming we do not require relativistic effects, where  $\eta$  represents the resistivity of the fluid. This may look different from what appears in the solid electromechanics case, where we

---

\*Massachusetts Institute of Technology, dannys4@mit.edu

†Sandia National Laboratory, bartv@sandia.gov

simply have  $\eta\mathbf{J} = \mathbf{E}'$ . However, note that  $\mathbf{E}'$  is the energy in a *static* frame of reference. Using an inertial frame of reference for a fluid at a point which has velocity  $\mathbf{u}$  and magnetic field  $\mathbf{B}$  would indeed give that  $\mathbf{E}' = \mathbf{E} + \mathbf{u} \times \mathbf{B}$ . This can be verified via unit analysis. In the case of an inertial frame of reference for the magnetic field,  $\mathbf{B}_I$ ,

$$\mathbf{B}_I = \mathbf{B} - \mathbf{u} \times \mathbf{E}/c^2,$$

so that we can approximate  $\mathbf{B} \approx \mathbf{B}_I$ .  $\mathbf{E}$  must then be of the order  $c^2/(\|\mathbf{u}\| \|\mathbf{B}\|)$  before it becomes worth considering the exact formulation  $\mathbf{B}_I$  [2], i.e., the geometric mean of  $\|\mathbf{u}\|$  and  $\|\mathbf{B}\|$  needs to be on the order of  $c$ , the speed of light. The Lorentz' law relates the primitive vector fields to the charge density  $\rho_E$  and the *force density*  $\mathbf{f}$  and dictates how much force a control volume of fluid experiences, as well as the direction of the force. This second law can be written as

$$\mathbf{f} = \rho_E \mathbf{E} + \mathbf{J} \times \mathbf{B}.$$

However, in consideration of the previous remarks about the magnitude of  $\mathbf{E}$  as well as an assumption of sufficiently low electrical resistance, we can approximate  $\rho_E \mathbf{E} \approx \mathbf{0}$  [9].

In the nonrelativistic case, we can combine our inertial version of Ohm's law with a nonrelativistic approximation of Amperè's and observe

$$\mathbf{E} = -\mathbf{u} \times \mathbf{B} + \frac{\eta}{\mu_0} \nabla \times \mathbf{B}.$$

Finally, the  $\mathbf{E}$  component can be completely eliminated via Faraday's law by taking the curl of both sides, as follows:

$$\begin{aligned} \nabla \times \mathbf{E} &= -\nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{\eta}{\mu_0} \nabla \times \nabla \times \mathbf{B} \\ \frac{\partial \mathbf{B}}{\partial t} &= \nabla \times (\mathbf{u} \times \mathbf{B}) - \frac{\eta}{\mu_0} \nabla \times \nabla \times \mathbf{B} && \text{Faraday} \\ &= \nabla \times (\mathbf{u} \times \mathbf{B}) - \frac{\eta}{\mu_0} (\nabla(\nabla \cdot \mathbf{B}) - \nabla^2 \mathbf{B}) && (*) \\ &= \nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{\eta}{\mu_0} \nabla^2 \mathbf{B} && \text{Gauss} \end{aligned}$$

Step (\*) comes from a known vector calculus identity, where  $[\nabla^2 \mathbf{B}]_j = (\partial_x^2 + \partial_y^2 + \partial_z^2) B_j$ . Then, dynamics defining the magnetic field in the fluid can be written as the “induction equation”,

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \frac{\eta}{\mu_0} \nabla^2 \mathbf{B}. \quad (2.1)$$

It is worth revisiting that the only dependence from the physical dynamics of the fluid come from the inertial frame of the electrical field,  $\mathbf{E}' = \mathbf{E} + \mathbf{u} \times \mathbf{B}$ . Some physical intuition for this might be that there are two components contributing to the evolution of  $\mathbf{B}$ : how rotational the flow of the fluid is when orthogonal to the magnetic field, and how quickly the magnetic field diffuses in space proportional to the resistivity of the fluid. One level of fidelity of the induction equation is “ideal” MHD, which approximates the induction equation by estimating the electrical conductivity of the fluid as infinity, i.e.,  $\eta = 0$ . This can be a reasonable approximation, as the conductivity of a multi-species fluid (e.g., plasma) can be high compared to the diffusion time of the magnetic field [2]. In practice, we use the vector calculus identity

$$\nabla \times (\mathbf{u} \times \mathbf{B}) = \nabla \cdot (\mathbf{B} \mathbf{u}^\top - \mathbf{u} \mathbf{B}^\top),$$

to take advantage of nice computational properties.

Moreover, since we are interested in simulating magnetic *fluids*, we must assume that a control volume of fluid not only follows the physics from electromagnetics, but also satisfies the physics for fluids. In particular, the homogeneous transport equation describes that the mass of the fluid must be conserved,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0.$$

Here  $\rho$  is used to describe the physical density of *fluid* in the control volume (not to be confused with  $\rho_E$ , density of the electric field). Similarly, to conserve all energy in a closed system, we must impose that momentum is conserved, which can be enforced by the Cauchy momentum equations

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \right) (\rho \mathbf{u}) = -\nabla p - \nabla \cdot \tau + \rho \mathbf{g},$$

where  $\rho \mathbf{g}$  describes any external force density that a particle might face in the system and  $\tau$  is the stress tensor. We recall the Lorentz force density and assume no other external force densities, so that

$$\rho \mathbf{g} = \mathbf{f} = \rho_E \mathbf{E} + \mathbf{J} \times \mathbf{B} \approx \frac{1}{\mu_0} (\nabla \times \mathbf{B}) \times \mathbf{B}.$$

Then, assuming incompressible flow and a viscous stress tensor, our system is governed by the momentum equation

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \right) (\rho \mathbf{u}) = -\nabla \cdot (p \mathbf{I} - \nu \nabla \mathbf{u}) + \frac{1}{\mu_0} (\nabla \times \mathbf{B}) \times \mathbf{B}, \quad (2.2)$$

where  $\nu$  is the fluid viscosity. One can use vector calculus to verify an equivalent formulation:

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \cdot (\mathbf{B} \mathbf{u}^\top - \mathbf{u} \mathbf{B}^\top) + \frac{\eta}{\mu_0} \nabla^2 \mathbf{B} = 0 \quad (2.3a)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) (\rho \mathbf{u}) + \nabla \cdot (p \mathbf{I} - \nu \nabla \mathbf{u}) - \frac{1}{\mu_0} \nabla \cdot \left( \mathbf{B} \mathbf{B}^\top - \frac{1}{2} \|\mathbf{B}\|^2 \mathbf{I} \right) = 0 \quad (2.3b)$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (2.3c)$$

$$\nabla \cdot \mathbf{B} = 0. \quad (2.3d)$$

Here, the primitive variables are  $\mathbf{B}$ ,  $\mathbf{u}$ ,  $p$ , and  $\rho$ . This formulation is *resistive* as opposed to ideal due to  $\eta > 0$  adding a magnetic diffusion term, *compressible* since  $\rho$  is enabled to change in time, and *viscous* due to the viscosity term  $\nu$ . There are extended formulations of MHD which add in relativistic dynamics, temperature simulation, the ability for multiple fluids, and many other capabilities. For further details, a broad overview of how these physics can integrate with energy is included in [14]; a more thorough yet concise derivation is included in [2]; a complete derivation from first physical principles in a magnetic fluids context can be found in [5, 8, 12].

**2.2. Inference.** The goal of inference is to reconstruct some unknown condition or model parameter by comparing numerical predictions to observations of magnetic fluid data using sparse, potentially noisy sensors. We first construct the solution operator  $\mathbf{v} : D \times \Omega \rightarrow$

$\mathbb{R}^m$  satisfying  $\mathbf{v}(\cdot; \theta) : x \mapsto [\mathbf{u}(x; \theta), \mathbf{B}(x; \theta), p(x; \theta), \rho(x; \theta), \dots]$ , where  $x \in D$  in our domain of interest (e.g., space, time) and  $\theta$  is the inference parameter (e.g., source term, resistivity, boundary condition). A *deterministic* and *exact* observation operator  $H : \mathbb{R}^m \rightarrow \mathbb{R}^n$  maps  $\mathbf{v}$  to an arbitrary function of the state variables or only a select set of variables, such as velocity  $\mathbf{u}$ . If we have a sensor on the domain at location  $x$ , we assume  $\mathbf{y}(x) = H(\mathbf{v}(x; \theta)) + \xi$ , where  $\xi$  is isotropic Gaussian white noise, so that the sensor only observes a slightly noisier  $H$ . If  $\xi$  has variance  $\sigma^2 \mathbf{I}_n$  and observations  $\{\hat{\mathbf{y}}(x_j)\}_{j=1}^S$ , we can calculate the log-likelihood for some particular parameter  $\theta$  via

$$\log p(\{\hat{\mathbf{y}}(x_j)\}_{j=1}^S; \theta) = -\frac{1}{2} \log(2\pi\sigma^2) - \sum_{j=1}^S \frac{1}{2\sigma^2} \|H(\mathbf{v}(x_j; \theta)) - \hat{\mathbf{y}}(x_j)\|^2.$$

**2.2.1. Maximum likelihood estimation.** One of the most common ways of estimating parameters is creating a maximum likelihood estimate (MLE), which is a frequentist method of creating a point-estimate of  $\theta$ . This philosophy assumes that there is some true  $\theta$  that provides  $\hat{\mathbf{y}}(x_j)$  and our only uncertainty is on the model itself. Then, we can see in our scenario that

$$\begin{aligned}\hat{\theta}_{MLE} &= \arg \max_{\theta} \log p(\{\hat{\mathbf{y}}(x_j)\}_{j=1}^S; \theta) = \arg \min_{\theta} -\log p(\{\hat{\mathbf{y}}(x_j)\}_{j=1}^S; \theta) \\ &= \arg \min_{\theta} \sum_{j=1}^S \|H(\mathbf{v}(x_j; \theta)) - \hat{\mathbf{y}}(x_j)\|^2,\end{aligned}$$

so in this case, we solve a nonlinear least-squares problem. The MLE is known to be consistent in the limit of infinite samples (i.e.,  $\hat{\theta}_{MLE}$  converges to the true  $\theta$  asymptotically). Refer to [7] for more information on the MLE,

**2.2.2. Bayesian parameter estimation.** An alternative approach is to suppose uncertainty on the parameter of interest itself, i.e. assume that  $\theta$  has some *prior* distribution  $p(\theta)$  (which describes our uncertainty on  $\theta$  before seeing any observation) and consider the problem of estimating the probability distribution  $p(\theta | \{\hat{\mathbf{y}}(x_j)\}_j)$ , often called the “posterior”. Due to Bayes’ rule, we know

$$\log p(\theta | \{\hat{\mathbf{y}}(x_j)\}_j) = \log p(\{\hat{\mathbf{y}}(x_j)\}_j | \theta) + \log p(\theta) - C_B$$

for some  $C_B$  dependent only on  $\{\hat{\mathbf{y}}(x_j)\}_j$ . Note here that we want a *distribution* over  $\theta$  in this method. Since the parameter is itself uncertain, we want a larger amount of information before making any downstream decisions. For example, if there are multiple likely values of  $\theta$  (i.e., we have multimodality), just seeing the most likely doesn’t necessarily help us; similarly, taking the most likely choice  $\theta$  can be a problem if we have a very large domain of likely values (high spread), or our true distribution is somehow degenerate. However, it is not immediately clear how to sample from this posterior distribution— while  $\mathbf{y}$  is Gaussian conditioned on  $\mathbf{v}$ , it is *not* Gaussian conditioned on  $\theta$  itself. Therefore, we turn to Markov chain Monte Carlo (MCMC) sampling to approximately sample the posterior distribution of  $\theta$ . While one can refer to literature for more information [3], we can outline the algorithm broadly in Algorithm 1.

The sample  $p(\theta' | \theta_n)$  is often called the “proposal distribution”, and  $\alpha$  is often called the “transition probability”. An example of such a step is the traditional random walk Metropolis-Hastings (RWMH) algorithm, where

$$\begin{aligned}p(\theta' | \theta_n) &= \mathcal{N}(\theta'; \theta_n, \gamma^2 \mathbf{I}_k), \\ \log g(\theta', \theta_n) &= \log p(\{\hat{\mathbf{y}}(x_j)\}_j | \theta') + \log p(\theta') - (\log p(\{\hat{\mathbf{y}}(x_j)\}_j | \theta_n) + \log p(\theta_n)).\end{aligned}$$

**Algorithm 1:** Generic MCMC Metropolis Step

---

```

input: Previous MCMC sample  $\theta_n$ 
output: Next MCMC sample  $\theta_{n+1}$ 
Draw  $\theta' \sim p(\theta'|\theta_n)$ 
Calculate acceptance  $\alpha = \log(g(\theta', \theta_n))$ 
Draw  $a \sim U(0, 1)$ 
if  $\log(a) < \alpha$  then
| Set  $\theta_{n+1} := \theta'$ 
else
| Set  $\theta_{n+1} := \theta_n$ 
end
```

---

Another example is Hamiltonian Monte Carlo (HMC), which proposes  $\theta' = h(\theta_n, L\Delta t)$ , an integration of a Hamiltonian system over  $L$  timesteps of size  $\Delta t$ . While the exact form of  $h$  (and the subsequent transition probability) can be found in [11], it notably requires access to the *gradient* of the log-likelihood,  $\nabla \log p(\{\hat{\mathbf{y}}(x_j)\}|\theta)$ . Since  $\theta$  does in fact parameterize the PDE, we can easily see via the chain rule that

$$\nabla_\theta \log p(\{\hat{\mathbf{y}}(x_j)\}_j|\theta) = -\frac{1}{2\sigma^2} \sum_{j=1}^n 2\|H(\mathbf{v}(x_j; \theta)) - \hat{\mathbf{y}}(x_j)\| \nabla_\theta H(\mathbf{v}(x_j; \theta)),$$

which requires a derivative of the PDE solution  $\mathbf{v}$  with respect to the input parameter  $\theta$ . This exemplifies the need for adjoint solvers when performing UQ and optimization.

**3. Formulation and implementation.** We have implemented the following MHD formulations using Sandia's MrHyDE framework, which uses different Trilinos libraries to provide a range of technologies, including mesh generation, Jacobians of flux terms with corresponding adjoints generated from automatic differentiation, parallel assembly and linear solvers, optimization libraries, and other convenient utilities. An important goal has been to investigate a range of MHD formulations with a line of sight to outer-loop analysis. Given the challenges of simulating a compressible and viscous fluid flow 2.3b we have developed a hierarchy of formulations due to either approximations or other techniques.

**3.1. Three-dimensional formulations.** First, we can generally approach the problem exactly as stated in 2.3, which can be arbitrarily difficult to solve. Some difficulties might be ensuring correct or consistent state equations. To avoid some numerical difficulties or ascribing any structure to the total energy of the system (needed to develop said state equations), we first examine when the fluid is incompressible, i.e.,  $\frac{\partial \rho}{\partial t} \equiv 0$ . This allows us to set  $\rho$  as a fixed parameter and arrive at the following expression:

$$\left( \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \right) (\rho \mathbf{u}) = \rho \frac{\partial \mathbf{u}}{\partial t} + \rho (\mathbf{u} \cdot \nabla) \mathbf{u}.$$

The equation 2.3c is replaced with  $\nabla \cdot \mathbf{u} = 0$ . This brings up a new problem, pervasive in MHD literature—if  $\rho$  is no longer a primitive variable, we now have eight physical constraints with only seven variables, and the system is overdetermined. Generally, the solution to this problem is to enforce Gauss' magnetic law,  $\nabla \cdot \mathbf{B} = 0$ , either implicitly or by construction. Without enforcing this constraint, one will arrive at nonphysical solutions.

A second, more generalized, three-dimensional formulation could resolve such an issue by reintroducing the electric field,  $\mathbf{E}$ , as an auxiliary variable; while it is a primary variable

in the original formulation of Maxwell's equations, we assume (by way of our nonrelativistic Amperè's law) it contains redundant information when combined with physical behavior of a fluid, as seen in 2.1. Then, we can formalize the coupling without any simplification,

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla p - \mathbf{f}_{NS}(\mathbf{u}, \mathbf{B}) = 0 \quad (3.1a)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (3.1b)$$

$$\frac{1}{c^2} \frac{\partial \mathbf{E}}{\partial t} + \sigma_c \mathbf{E} + \mathbf{f}_M(\mathbf{u}, \mathbf{B}) - \frac{1}{\mu_0} \nabla \times \mathbf{B} = 0 \quad (3.1c)$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} = 0 \quad (3.1d)$$

$$\nabla \cdot \mathbf{B} = 0 \quad (3.1e)$$

using the conductivity  $\sigma_c = 1/\eta$ . We can interpret  $\mathbf{f}_M$  as an ambient current passing through the medium, where  $\mathbf{f}_{NS}$  is an additional force density that may be acting on the fluid. In this setting we know the form of these source terms explicitly as

$$\mathbf{f}_{NS}(\mathbf{u}, \mathbf{B}) = \rho_e \mathbf{E} + \mathbf{J} \times \mathbf{B} = \rho_E \mathbf{E} + \sigma_c (\mathbf{E} + \mathbf{u} \times \mathbf{B}) \times \mathbf{B} \approx \sigma_c (\mathbf{E} + \mathbf{u} \times \mathbf{B}) \times \mathbf{B}$$

$$\mathbf{f}_M(\mathbf{u}, \mathbf{B}) = \sigma_c \mathbf{u} \times \mathbf{B}.$$

This simulation is more flexible than 2.3 when considering incompressible flows, as we partially take relativity into account. While this is *still* an overdetermined system, it is overdetermined purely due to the Maxwell terms; any strategies to alleviate this concern when solving pure Maxwell equations can be directly translated to this MHD form, such as using an Hdiv discretization of the magnetic field. Further, this form is powerful for established flexible PDE solvers, as the Navier-Stokes and Maxwell's equations are ubiquitous. Using 3.1 "only" requires interfacing between two solution capabilities via coupling terms expressed directly as primitive variables (with no derivatives required). Many modern tools, such as MrHyDE, have options to solve similar problems, with expressive coupling capabilities. Various numerical complications still need to be addressed.

**3.2. Two-dimensional formulation.** Simplifying the problem via lowering the dimension is complicated because of the inherent three dimensional nature of MHD. However, we take the formulation introduced in [15] via an auxiliary variable  $\mathbf{A}$  as the "vector magnetic potential", which is defined such that  $\nabla \times \mathbf{A} \equiv \mathbf{B}$ . Note that this doesn't fully define  $\mathbf{A}$ , but it does necessarily satisfy  $\nabla \cdot \mathbf{B} = \nabla \cdot \nabla \times \mathbf{A} \equiv 0$ . We then approximate the magnetic field only exist within a two-dimensional space, assuming  $\mathbf{B} = (B_x, B_y, 0)$ , and impose the form that  $\mathbf{A} = (0, 0, A_z)$ , so  $\mathbf{B}$  is entirely determined by real-valued function  $A_z$ . We can observe that this formulation satisfies the solenoidal constraint implicitly, as

$$\mathbf{B} \equiv \nabla \times \mathbf{A} = (\partial_y A_z, -\partial_x A_z, 0) \Rightarrow \nabla \cdot \mathbf{B} = \partial_{xy} A_z - \partial_{xy} A_z = 0.$$

This model is very flexible, as it is expressive enough to reasonably model problems in two-dimensional channels and domains with axisymmetric properties, but it reduces simulation cost significantly. Then we can replace the induction equation 2.3a with a much simpler advection-diffusion problem,

$$\frac{\partial A_z}{\partial t} = -\mathbf{u} \cdot \nabla A_z + \frac{\eta}{\mu_0} \nabla^2 A_z,$$

which is a straightforward simplification of 2.1 when assuming  $\partial_z A_z \equiv 0$  with sufficient regularity of the solution. This implementation suffers oscillatory and discontinuous behavior in pressure likely requiring some formulation modifications in the boundary conditions which are not specified in [15].

**3.3. One-dimensional formulations.** We first assume that the fluid travels along a tube of very small diameter and view the dynamics where a control volume along the tube can have magnetic field or velocity in any direction, but can only move in the  $x$  direction (e.g.,  $\partial_y \mathbf{u} \equiv \mathbf{0}$  and  $\partial_z \mathbf{B} \equiv \mathbf{0}$ ). Then, if we assume  $B_x$  is some constant, we know by definition that  $\nabla \cdot \mathbf{B} \equiv 0$  and thus we need not worry about satisfying the solenoidal constraint. The formulation derived from [1], accounting for compressibility and total mechanical energy  $E$ , can be given by

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \partial_x(\rho u_x) &= 0 \\ \frac{\partial \rho \mathbf{u}}{\partial t} + \partial_x \left[ \rho u_x \mathbf{u} - \frac{1}{\mu_0} B_x \mathbf{B} + \left( p + \frac{1}{2\mu_0} \|\mathbf{B}\|^2 \right) \mathbf{e}_x \right] &= \mathbf{0} \\ \frac{\partial \mathbf{B}}{\partial t} + \partial_x(u_x \mathbf{B} - B_x \mathbf{u}) &= \mathbf{0} \\ E + \partial_x \left[ \left( E + p + \frac{1}{2\mu_0} \|\mathbf{B}\|^2 \right) u_x - \frac{1}{\mu_0} B_x \mathbf{B}^\top \mathbf{u} \right] &= 0 \end{aligned}$$

where  $\mathbf{e}_x$  is just the first elementary vector  $(1, 0, 0)$ ; this formulation employs an “ideal” MHD approximation with  $\eta = 0$ . To close the system, we use a state equation

$$p = (\gamma - 1) \left( E - \frac{1}{2} \rho \|\mathbf{u}\|^2 - \frac{1}{2\mu_0} \|\mathbf{B}\|^2 \right),$$

with  $\gamma$  as the adiabatic index, usually estimated in this context as  $5/3$  or  $2$  [1]. In practice, this formulation creates promising results. Unfortunately, the one-dimensional full MHD tests involves shocks, requiring either the implementation of flux limiters in MrHyDE’s discontinuous Galerkin scheme or a finite element variational multiscale approach with specialized solver and preconditioner strategies.

However, while the above form only simulates in one dimension to allow more complex physics, we can simplify the formulation further. We define the Hartmann number as  $Ha = \|\mathbf{B}\|L/\sqrt{\eta\nu}$ , where  $L$  is the “characteristic length scale” of the domain, which in one dimension is just the length of the domain. The well-established Hartmann channel flow problem can be described as a steady-state flow of a magnetic fluid through a thin channel with a small  $Ha$  value. The geometry of the problem is described in Figure 3.1, where the magnetic field is the focus in the  $y$  direction across the channel and the fluid flow in the  $x$  direction through the channel.

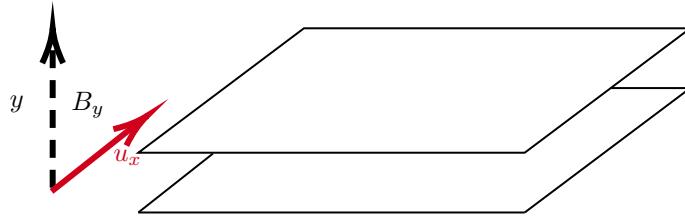


FIG. 3.1. Diagram of the Hartmann flow

The equations associated with the one-dimensional Hartmann flow are then

$$Ha \frac{\partial B_y}{\partial y} + \frac{\partial^2 u_x}{\partial y^2} = f, \quad Ha \frac{\partial u_x}{\partial y} + \frac{\partial^2 B_y}{\partial y^2} = 0,$$

where  $f$  is defined as a forcing function. This system is formalized in [4, 13] and originates from [10].

**4. Hartmann channel flow parameter estimation.** For the following numerical experiments, we impose boundary conditions found in [13],

$$u_x = 0, \quad \pm \frac{\partial B_y}{\partial y} + \eta B_y = 0 \text{ for } y = \pm 1,$$

with forcing term  $f \equiv -1$  and Hartmann number  $Ha \equiv 1$ . These conditions give an analytical solution of

$$u_x(y) = \hat{u} \left[ 1 - \frac{\cosh(Ha y)}{\cosh(Ha)} \right], \quad B_y(y) = -\frac{y}{Ha} + \hat{u} \frac{\sinh(Ha y)}{\cosh(Ha)}, \quad \hat{u} \equiv \frac{\eta + 1}{Ha(\eta + 1 + \tanh(Ha))}.$$

The comparison of the numerical solution to the analytical solution is given in Figure 4.1 and as expected our numerical simulation matches the analytical solutions. Given a validated solution, we can consider the implementation of an inference problem in which we estimate the forcing function parameter  $f$ . Physically,  $f$  represents the force that pushes the fluid through the channel, which might be of interest in an application where we can only observe the fluid flow and the magnetic field.

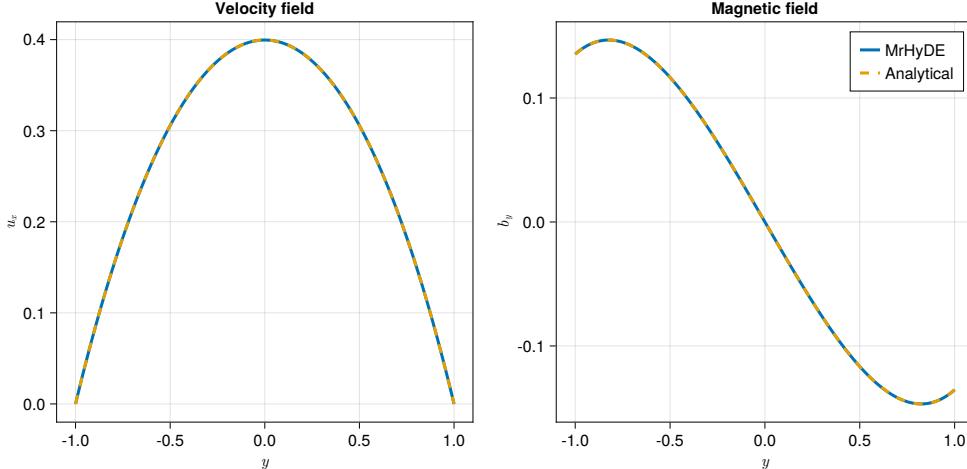


FIG. 4.1. Comparison of *MrHyDE* solution to analytical solution

**4.1. Maximum likelihood estimation.** We parameterize the PDE with  $\kappa$ , the log-negative forcing, via  $f = -\exp(\kappa)$  (i.e.,  $\kappa := \log(-f)$ ) and attempt to estimate  $\kappa$  in a scenario where we have limited spatial observations. We assume exact observations of the velocity with no measurement noise, i.e.,  $H(\mathbf{v}(x; \kappa)) = u(x; \kappa)$  and  $\sigma^2 \equiv 0$  as discussed in Section 2.2. For the following experiments, we assume a true value of  $\kappa = 0.5$ , a value which does *not* give the analytical solution displayed in Figure 4.1. We observe the velocity for  $x_{obs} \in \{-0.95, -0.72, -0.25, 0.01, 0.09, 0.21, 0.89, 0.91\}$ . In a PDE-constrained optimization setting, we would like to minimize the function

$$L(\kappa) = \sum_{j=1}^{N_{sens}} (u(x_j; \kappa) - u_j^{(obs)})^2 + \mathcal{R}(\kappa),$$

where  $u$  is our velocity for a given  $\kappa$ ,  $u_j^{(obs)}$  is a sensor's observation of the velocity at  $x_j$  and  $\mathcal{R}(\kappa)$  is a regularization term defined on  $\Omega$ , the domain of our parameter of interest. This

regularized problem in fact corresponds with estimating a posterior mode, where the prior is determined by the type of regularization [7]; this smooths out solutions in the presence of sparse data or otherwise ill-conditioned likelihood functions. However, due to the smooth nature of our problem, we present results omitting the regularization term. Figure 4.2 gives the convergence behavior of this MLE estimate when starting the guess at  $\hat{\kappa}_0 = 1.0$ , which shows (in the absence of observation noise) that we can converge to the correct value of  $\kappa = 0.5$ . This exemplifies the smoothness of the Hartmann system in the parameter space, and shows the superb conditioning of the system for reasonably small  $Ha$ .

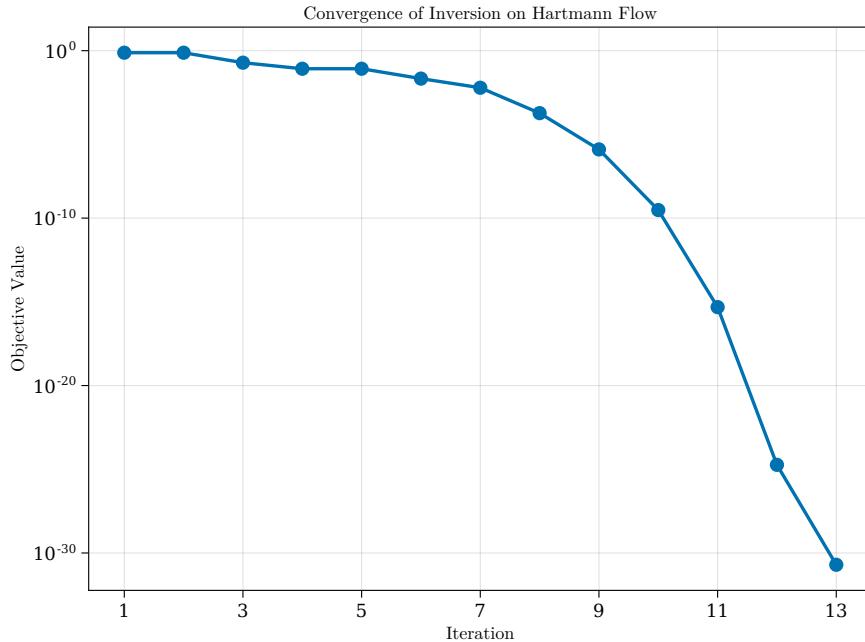


FIG. 4.2. *Convergence of estimate of  $\kappa$ , true value 0.5, initial guess 1.0*

Of course, a major drawback to such a point estimate is when we *do* have uncertainty due to measurement noise. We assume our observations  $u^{(obs)}$  instead come from a model  $u_j^{(obs)} = u(x_j; \kappa) + \sigma \varepsilon$ , where  $\varepsilon \sim \mathcal{N}(0, I)$ . In Figure 4.3, we visualize the error in our estimate  $\hat{\kappa}$  across increasing  $\sigma$ , and over  $N = 50$  trials of optimization (where a new measurement error is drawn for each trial).

**4.2. Bayesian inference.** We perform a Bayesian inversion on  $\kappa$  and assume observation noise  $\sigma = 0.0025$ . Then, we can impose a prior distribution over  $\kappa$  to sample from the posterior  $p(\kappa|y_{obs})$ . We choose a prior of  $p(\kappa)$  to be a normal distribution with mean 1 and variance 4. While it has a mean identical to the initial guess in the MLE inference, this prior suggests a scenario in which our original guess is extraordinarily uninformed due to the very large variance. Here we compare the performance of the RWMH MCMC algorithm to an implementation of HMC provided by [16]. The RWMH sampler uses a normal proposal density with variance 0.01, and HMC uses an adaptive No U-Turn Sampling algorithm [11]. The diagnostics from creating 25,000 samples from each algorithm (the first 20% of which are “burnin”, i.e. discarded after creation) are displayed in Figure 4.4. Both the auto-correlation and trace plot suggest proper convergence of the chain. While more complex diagnostics exist, these suffice for a one-dimensional sampling procedure.

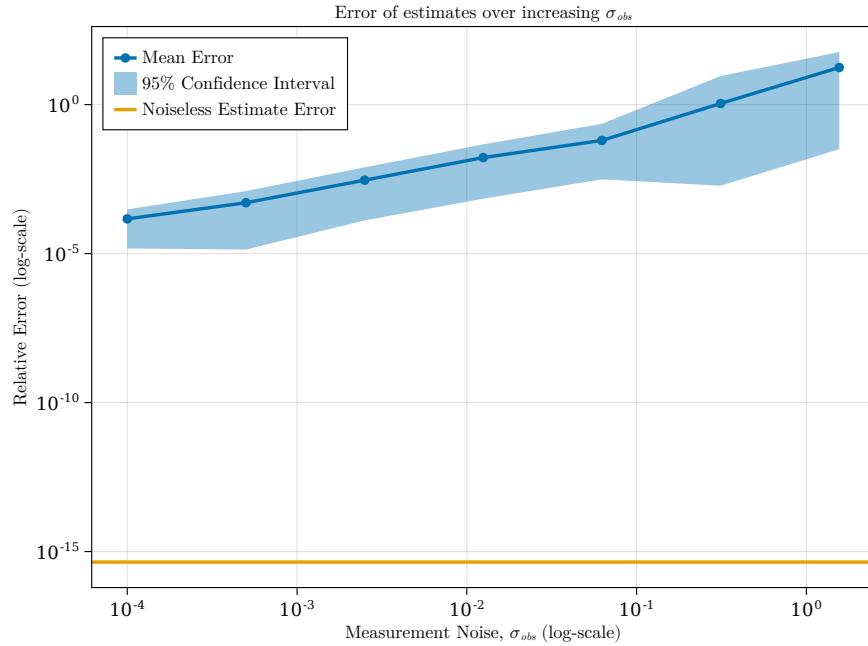


FIG. 4.3. *Relative error of point estimate  $\hat{\kappa}$  over size of observation noise with a 95% confidence interval*

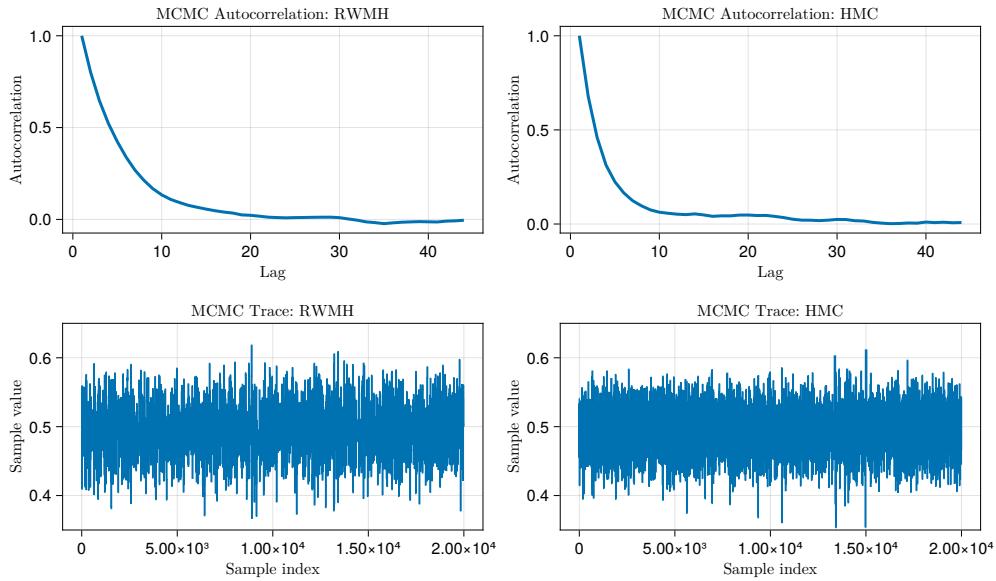


FIG. 4.4. *(Top): Autocorrelation in the samples, (Bottom): Traceplot of the samples*

We can visualize the density of the posterior samples for each algorithm, and compare these to the prior distribution as well as the true  $\kappa = 0.5$ , seen in Figure 4.5. This shows a very distinct shift from prior to posterior, with significant concentration around the true

value. Since the Hartmann system of differential equations is linear, and the observation noise is small, we see that the MLE point from the frequentist inference asymptotically matches posterior mode (i.e., maximum a posteriori point).

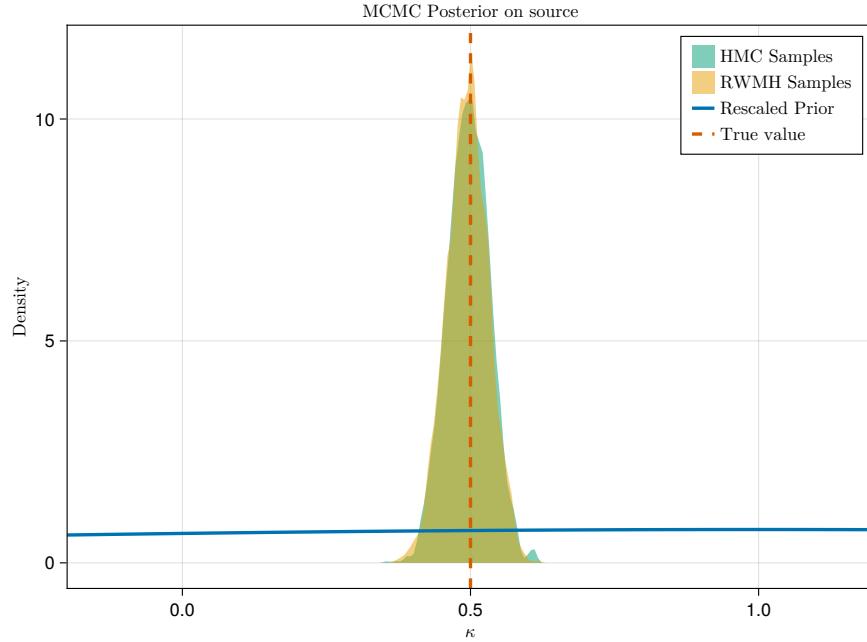


FIG. 4.5. Plot of estimated density of posterior samples from MCMC procedure

While the MLE and posterior mode are not expected to be exactly the same due to the effect of the prior, it is no surprise that the MLE and true value are near that mode. Generally, though, the MHD system is extremely nonlinear containing many more variables in three dimensions. As such, much of this inference is open to extension to more complicated, higher dimensional systems, and thus employing techniques like regularization that were only briefly touched on here. Further, with such flexibility, we open the opportunity to creating gradient-based surrogates for PDE solutions via this framework.

**5. Conclusion.** Via examining several different approaches to the MHD problem, not all of which are included here, we glean insight into the difficulties of simulating MHD. In particular, many test problems in the literature were found to either have complex numerical properties (e.g., shocks), lack analytical solutions, or omit important experimental setup details (e.g., initial or boundary conditions). It was found in implementation that the solenoidal constraint was indeed a difficulty when attempting to consider an incompressible formulation; in contrast, compressible formulations faced difficulties in simulation similar to direct numerical simulation of nonmagnetic fluids. Finally, it was found that understanding the hierarchy of physical models and simplifications is not necessarily direct. For example, we include a compressible one-dimensional formulation with energy preservation, which can be contrasted with the two-dimensional incompressible formulation. Further, these models are not a complete description of all simulation possibilities of magnetic fluids.

When applying inference to the Hartmann channel problem, we significantly simplify the true MHD problem. In the Bayesian estimation procedure, since the system is linear, we see extremely tight convergence to the true parameter. The problem itself is only very

weakly convective for small Hartmann number, exists in one dimension, is only a steady-state equation, and has a smooth solution [4, 13]. This makes the inference problem significantly simplified, and thus we caution any extensive conclusions from this implementation towards more complicated MHD regimes.

**6. Acknowledgments.** The authors thank Eric Cyr, Kris Beckwith, and Allen Robinson for valuable insight into MHD. 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. 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. This work was supported by the US Department of Energy, Office of Advanced Scientific Computing Research, Field Work Proposal 20-023231.

#### REFERENCES

- [1] J. BALBÁS, E. TADMOR, AND C.-C. WU, *Non-oscillatory central schemes for one- and two-dimensional MHD equations: I*, Journal of Computational Physics, 201 (2004), pp. 261–285.
- [2] R. D. BLANDFORD AND K. S. THORNE, *Applications of classical physics*, lecture notes, California Institute of Technology, (2008), p. 12.
- [3] S. BROOKS, A. GELMAN, G. JONES, AND X.-L. MENG, *Handbook of markov chain monte carlo*, CRC press, 2011.
- [4] C. C. CHANG AND T. S. LUNDGREN, *Duct flow in magnetohydrodynamics*, Zeitschrift für angewandte Mathematik und Physik ZAMP, 12 (1961), pp. 100–114.
- [5] P. A. DAVIDSON, *An introduction to magnetohydrodynamics*, 2002.
- [6] M. FARADAY, *Experimental researches in electricity: reprinted from the Philosophical Transactions of 1831-1843, 1846-1852*, vol. 1, R. and JE Taylor, 1839.
- [7] A. GELMAN, J. B. CARLIN, H. S. STERN, D. B. DUNSON, A. VEHTARI, AND D. B. RUBIN, *Bayesian data analysis*, CRC press, 2013.
- [8] J. P. GOEDBLOED AND S. POEDTS, *Principles of magnetohydrodynamics: with applications to laboratory and astrophysical plasmas*, Cambridge university press, 2004.
- [9] J. GOODMAN, *Stellar Magnetohydrodynamics*, 2006.
- [10] J. HARTMANN AND F. LAZARUS, *Hg-dynamics ii*, Theory of laminar flow of electrically conductive Liquids in a Homogeneous Magnetic Field, 15 (1937).
- [11] M. D. HOFFMAN, A. GELMAN, ET AL., *The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo.*, J. Mach. Learn. Res., 15 (2014), pp. 1593–1623.
- [12] R. J. MOREAU, *Magnetohydrodynamics*, vol. 3, Springer Science & Business Media, 1990.
- [13] U. MÜLLER AND L. BÜHLER, *Magnetofluidynamics in channels and containers*, Springer Science & Business Media, 2001.
- [14] K. G. POWELL, P. L. ROE, T. J. LINDE, T. I. GOMBOSI, AND D. L. DE ZEEUW, *A solution-adaptive upwind scheme for ideal magnetohydrodynamics*, Journal of Computational Physics, 154 (1999), pp. 284–309.
- [15] J. N. SHADID, R. P. PAWLOWSKI, J. W. BANKS, L. CHACÓN, P. T. LIN, AND R. S. TUMINARO, *Towards a scalable fully-implicit fully-coupled resistive mhd formulation with stabilized fe methods*, Journal of Computational Physics, 229 (2010), pp. 7649–7671.
- [16] K. XU, H. GE, W. TEBBUTT, M. TAREK, M. TRAPP, AND Z. GHAHRAMANI, *Advancedhmc.jl: A robust, modular and efficient implementation of advanced hmc algorithms*, in Symposium on Advances in Approximate Bayesian Inference, PMLR, 2020, pp. 1–10.

## IMPLEMENTING A RANDOMIZED EIGENSOLVER IN ANASAZI AND ITS APPLICATION TO SPECTRAL GRAPH PARTITIONING

HEATHER SWITZER\*, JENNIFER LOE†, AND ERIK G. BOMAN‡

**Abstract.**

The Anasazi software package in Trilinos [2, 17] offers a range of eigensolvers designed to generate accurate estimations of eigenpairs. A few notable solvers include LOBPCG, Block Davidson, and Block Krylov Schur. Achieving high accuracy with these solvers often requires frequent reorthogonalization of the Krylov basis during its construction to enable proper information extraction via the Rayleigh-Ritz technique. However, many applications may not necessitate high-precision eigenpairs, and quicker, lower-accuracy approximations could be satisfactory.

In this study, we introduce a new eigensolver to Anasazi, utilizing the subspace iteration method with a block of randomly generated initial starting vectors. A comparative analysis of convergence and time is conducted against LOBPCG and Block Krylov Schur, showing that the randomized method requires less computation and time per iteration to execute, but exhibits slower convergence when compared to the other methods. Additionally, we investigate the application of the randomized solver for obtaining eigenvectors used in spectral graph partitioning. Our results show that the randomized eigensolver is comparable to the LOBPCG in this context, providing a speedup in the eigensolver with lightly inferior results.

**1. Introduction.** One of the most fundamental problems in numerical linear algebra revolves around the eigenvalue problem of the form:

$$Ax = x\lambda, \quad (1.1)$$

where  $A \in \mathbb{C}^{n \times n}$ . This problem is prevalent across various fields of the natural sciences, including vibrational analysis, geological body localization, and structural domain analysis [8, 3, 19]. In Equation 1.1, any combination of  $(x, \lambda)$  that satisfies the equation is considered an eigenpair of  $A$ . “ $x$ ” in this context is a column vector of length  $n$  containing an eigenvector of  $A$ , while the associated scalar  $\lambda$  contains the eigenvalue. When seeking multiple eigenpairs  $x$  becomes a matrix of column vectors and  $\lambda$  transforms into a diagonal matrix with the eigenvalues as entries.

While Equation 1.1 appears deceptively simplistic, solving it is a computationally intensive and challenging task. Direct eigensolvers take  $O(n^3)$  operations to compute all eigenpairs. To illustrate this, finding all eigenpairs of a matrix  $A \in \mathbb{C}^{10^5 \times 10^5}$  would take  $10^{15} +$  operations. As the size of matrices being used by researchers today increasingly grows larger, direct methods become impractical, if not impossible.

Many applications do not require all of the eigenpairs of a matrix; only a small subset from a specific portion of the spectrum. This consideration motivates the adoption of iterative methods. These methods start with an initial guess of the eigenvectors before progressively refining them to achieve increasingly accurate approximations of a specified number of eigenpairs.

Anasazi, a software package inside the Trilinos framework, implements efficient and scalable eigensolvers [2, 17]. A few examples of iterative solvers that can be run by Anasazi include Block Krylov Schur, Davidson, and LOBPCG. These Krylov-based methods construct a Krylov basis of the form

$$V = \text{span}\{y, Ay, A^2y, A^3y, \dots\} \quad (1.2)$$

where  $y$  serves as an initial vector (either a guess or a random vector). Utilizing this Krylov basis, eigenpair approximations are derived through the Rayleigh-Ritz technique [11]. While

---

\*The College of William & Mary, hmswitzer@wm.edu,

†Sandia National Lab, jloe@sandia.gov

‡Sandia National Lab, egboman@sandia.gov

these methods in Anasazi work well to find accurate approximations, it does come with a price in performance due to the necessity of frequent basis reorthogonalization. In practice, as an iterative method continues to build up a Krylov basis, the condition number of this basis tends to escalate. This higher condition number results in a slowdown, stagnation, or even destruction of the eigensolver's convergence. Reorthogonalizing the basis can temporarily fix the condition number, but the continuous need for this comes with an additional computational cost, potentially causing a bottleneck in the solver. As stated previously, several applications do not necessitate the need for more accuracy in their approximated eigenpairs. This results in eigensolvers, such as LOBPCG, being inefficient and computationally wasteful. Instead, we could use a more economical solver that prioritizes speed over the accuracy of the solution. The subspace iteration method, which was previously not integrated into Anasazi, shows to be a potential solution to this. In this work, we:

1. **Introduce a new eigensolver:** We implement the subspace iteration method with initial starting vectors in the Anasazi package.
2. **Compare the performance of different eigensolvers:** We provide timing and convergence assessments that compare the performance between the randomized solver, LOBPCG, and Block Krylov Schur methods in Anasazi.
3. **Examine the use of the randomized solver in applications:** We use the randomized solver to compute the eigenvectors for spectral graph partitioning. In addition, we also provide comparative analysis using different eigensolvers for this application.

### 1.1. Roadmap.

The remainder of this report is as follows:

- Section 2 describes the background of the different eigenvalue methods tested. These methods include LOBPCG, Block Krylov Schur, and the randomized solver.
- Section 3 will discuss the implementation details of the randomized solver in Anasazi.
- 4 provides an analysis of the time and convergence comparisons for solvers in Anasazi.
- Sections 5 6 explore the application of the randomized solver in the context of spectral graph partitioning, followed by numerical results.
- Sections 7 and 8 conclude this report by briefly discussing the achieved results before giving a concise summary.

**2. Background.** In this section, we provide a brief overview of Krylov methods along three eigensolvers. First, the LOBPCG eigensolver discussed, followed by the Block Krylov Schur method—both of which have already been integrated into Anasazi. Finally, the newly implemented randomized eigensolver will be explained.

**2.1. LOBPCG.** The Locally Optimal Block Preconditioned Conjugate Gradient method (LOBPCG) is a well-established iterative method used to approximate the eigenpairs of a matrix pertaining to the extreme parts of the spectrum, i.e., the eigenpairs corresponding to the smallest or largest eigenvalues [9]. By constructing a Krylov subspace,  $V = \{y, Ay, A^2y, \dots, A^{k-1}y\}$  with one or more initial vectors  $y$ , the Rayleigh-Ritz method can be applied to approximate the eigenpairs of the matrix  $A$ . This involves minimizing or maximizing the Rayleigh quotient  $\frac{x^T Ax}{x^T x}$ , where  $x$  is an eigenvector. The advantage of LOBPCG is its ability to use a preconditioner, a feature that can not be done with other methods such as Lanczos or Arnoldi. The process for LOBPCG is outlined in Algorithm 1. The Rayleigh-Ritz method, utilized in both LOBPCG and the randomized solver, can be seen on Algorithm 2.

**Algorithm 1** LOBPCG

---

**Require:** Matrix  $A \in \mathbb{R}^{n \times n}$ , the number of requested eigenpairs  $nev$

- 1: Generate initial random starting vectors  $X^{(0)} \in \mathbb{C}^{n \times nev}$ . These can be random.
- 2:  $[Y^{(1)}, \Theta^{(1)}] = \text{Rayleigh-Ritz}(X^{(0)})$
- 3:  $X^{(1)} = X^{(0)}Y^{(1)}$
- 4: Compute residual  $R^{(1)} = AX^{(1)} - X^{(1)}\Theta^{(1)}$
- 5:  $P^{(1)} = []$
- 6: **for**  $j = 1, 2, \dots$  **do**
- 7:    $W^{(j)} = K^{-1}R^{(j)}$  ▷ Preconditioning
- 8:    $S^{(j)} = [X^{(j)}, W^{(j)}, P^{(j)}]$
- 9:    $[Y^{(j+1)}, \Theta^{(j+1)}] = \text{Rayleigh-Ritz}(S^{(j)})$
- 10:    $X^{(j+1)} = S^{(j)}Y^{(j+1)}(:, 1 : nev)$
- 11:    $R^{(j+1)} = AX^{(j+1)} - X^{(j+1)}\Theta^{(j+1)}$
- 12:    $P^{(j+1)} = S^{(j)}(:, nev + 1 : \text{end})Y^{(j+1)}(nev + 1 : \text{end}, 1 : nev)$
- 13:   If  $nev$  eigenpairs have converged, break out of the **for** loop
- 14: **end for**
- 15: **Return** converged eigenpairs  $X$  and  $\Theta$

---

**Algorithm 2** Rayleigh-Ritz

---

**Require:** Matrix  $A \in \mathbb{R}^{n \times n}$ , orthonormal basis matrix  $X \in \mathbb{C}^{n \times d}$

- 1: Solve the eigenproblem  $(X^TAX)Y = \Theta Y$
- 2: **Return** eigenvectors  $Y$  and their corresponding matrix of eigenvalues  $\Theta$

---

**2.2. Block Krylov Schur.** The Krylov Schur eigensolver, introduced by Stewart in 2001 [15], extends the implicitly restarted Arnoldi algorithm by replacing the Arnoldi decomposition with a Krylov decomposition. This substitution allows more stability and makes it easier to deflate converged Ritz pairs. In Anasazi, a block version of this algorithm has been incorporated.

**2.3. The Randomized Eigensolver.** The primary limitation of LOBPCG and Block Krylov Schur lies in their computational cost attributed to orthogonalizing the basis at every iteration. While this orthogonalization increases the solver's ability to create highly accurate eigenpair approximations for a matrix  $A$ , it introduces a significant amount of computational overhead. When used in tandem with preconditioning, convergence of the solver accelerates, making it well suited for applications in which these high accuracy vectors are needed. However, for certain applications such as graph partitioning, high-accuracy eigenvectors may not be needed. In these cases, the use of a cost-effective, randomized solver that sacrifices accuracy for reduced computational power may provide a better alternative.

Initially, an orthonormal matrix of random vectors  $V \in \mathbb{C}^{n \times nev}$  where  $nev$  represents the number of eigenpairs sought, is generated for use by the solver. After constructing these initial vectors, the algorithm left-multiplies the matrix  $A$  a specified number of times,  $q$ , before solving the Rayleigh-Ritz problem for  $A^q X$ . It is necessary to ensure  $A^q X$  is orthonormal before executing Rayleigh-Ritz procedure. Note that this randomized subspace iteration method converges to the eigenpairs of  $A$  corresponding to eigenvalues of largest magnitude [12]. Further details regarding the parameters `Res.Freq` and `Ortho.Freq` will be discussed in Section 3.

**3. Randomized Solver in Anasazi.** All prior work regarding the randomized eigensolver was located in a test directory under the Tpetra subfolder of Anasazi. Tpetra serves

as the package that provides linear algebra objects and their associated operations [16]. Although a basic script implementing the randomized eigensolver existed in Anasazi's source directory, it had yet to be fully integrated in. This issue has since been addressed and the randomized solver can now be found under the file `anasazi/src/AnasaziRandomizedSolMgr.hpp`. Aside from the standard parameters such as the tolerance, maximum number of iterations, block size, etc., two additional options were added, and are used as shown in Algorithm 3.

1. **Orthogonalization Frequency:** This parameter allows users to designate how many iterations should pass before reorthogonalizing the basis.  
**Default:** 0. No additional orthogonalization is applied. Orthogonalization of the basis occurs twice: initially to orthogonalize the random vectors before the subspace iteration begins, and subsequently to orthogonalize the basis immediately before solving the Rayleigh-Ritz problem.
2. **Residual Frequency:** This parameter allows users to designate how many iterations should pass before checking for the convergence of the requested eigenvectors. This involves solving the Rayleigh-Ritz problem, which necessitates the reorthogonalization of the basis. If all requested eigenpairs are converged, the subspace iteration ends and the eigenpairs are returned.  
**Default:** 0. Residuals are not computed until the randomized method performs the user-specified number of iterations.

---

**Algorithm 3** Randomized Eigensolver

---

**Require:** Matrix  $A \in \mathbb{R}^{n \times n}$ , number of requested eigenpair  $nev$ , number of iterations  $q$ , frequency of orthogonalization `Ortho_Freq`, Number of iterations between checking residuals `Res_Freq`, convergence tolerance `tol`

- 1: Generate a random matrix  $\hat{V} \in \mathbb{R}^{n \times nev}$
- 2:  $[V, \sim] = \text{qr}(\hat{V}, 0)$  ▷ Ensure the random starting vectors are orthonormal
- 3: **for**  $j = 1, 2, \dots, q$  **do**
- 4:    $V = AV$
- 5:   **if**  $\text{mod}(j, \text{Ortho\_Freq}) == 0$  or  $\text{mod}(j, \text{Res\_Freq}) == 0$  **then**
- 6:      $[V, \sim] = \text{qr}(V, 0)$  ▷ Orthonormalize basis vectors
- 7:   **end if**
- 8:   **if**  $\text{mod}(j, \text{Res\_Freq}) == 0$  **then** ▷ Check convergence
- 9:     Compute the Ritz vectors  $X = VY$  and Ritz values  $\lambda$
- 10:   Compute the residual norms  $\|AX - X\lambda\|$
- 11:   **if** All residual norms  $< \text{tol}$  **then**
- 12:     **Return**
- 13:   **end if**
- 14:   **end if**
- 15: **end for**
- 16:  $[V, \sim] = \text{qr}(V, 0)$  ▷ Ensure basis vectors are orthonormal
- 17:  $[Y, \Theta] = \text{Rayleigh-Ritz}(V)$
- 18: Compute the Ritz vectors  $X = VY$
- 19: **Return** Ritz vectors  $X$  and their corresponding matrix of eigenvalues  $\Theta$

---

**4. Comparisons With Other Eigensolvers In Anasazi.** To test the efficacy of the randomized solver, we conduct a convergence comparison using various matrices obtained from SuiteSparse, as detailed in Table 4.1. The comparison includes the pre-existing Anasazi solvers Block Krylov Schur and LOBPCG. While the Generalized Davidson and Block Davidson solvers were also tested, their increased run times did not justify an dis-

TABLE 4.1  
*Information about the matrices being tested from SuiteSparse.*

Matrix Name	# Rows	Nnz	Problem Type	Cond #
c-33	6,317	56,123	Optimization	1.12E+09
c-40	9,941	81,501	Optimization	1.11E+06
cryg10000	10,000	49,699	Materials	2.26E+19
delaunay_n13	8,192	49,094	Undirected Graph	1.03e+04
flowmeter0	9,669	67,391	Model Reduction	1.59E+07
fv3	9,801	87,025	2D/3D	1.03E+03
G67	10,000	40,000	Undirected Weighted Random Graph	8.10E+04
kmnist_norm_10NN	10,000	156,932	Undirected Weighted Graph	-
mark3jac020	9,129	52,883	Economic	3.89E+13
vsp_data_and_seymourl	9,167	111,732	Random Unweighted Graph	3.90E+18

cernible performance gains, and were therefore excluded from the results. Additionally, residual results for the randomized solver with **Orthogonalization Frequency** set to 0 or 100 were omitted, but their timing results are still included. Due to the reorthogonalizing occurring so infrequently, rank deficiency occurred in the bases for 9 out of 10, resulted in a breakdown of convergence and unstable results after some iterations had passes. The exception to this was the “cryg10000” matrix. Instead, residual results for the randomized solver with orthogonalization frequencies of 1 and 10 are shown.

All tests were run on an Intel(R) Xeon(R) W-2155 CPU @ 3.30GHz with four MPI processes. The solver parameters were set as follows:

- **Number of requested eigenvalues:** 10
- **Block size:** 10
- **Convergence Tolerance:** 0
- **Part of the eigenspectrum sought:** Largest magnitude
- **Orthogonalization method:** SVQB (detailed in [14])
- **Preconditioning:** None
- **Restarting:** None

All other solver-specific parameters were left as their default values.

All matrices tested, with the exception of “fv3”, are not symmetric positive definite. This posed a challenge in LOBPCG, as the default setting is to fully orthogonalize the basis using the Cholesky decomposition—a process made for Hermitian positive definite matrices. Surprisingly, only two matrices, “mark3jac020” and “c-40”, exhibited failure in the Cholesky decomposition, while the others did not encounter this issue. This failure resulted in the solver returning an error after a certain number of iterations. When full orthogonalization in LOBPCG was disabled for these two matrices, the residual norms quickly spiked and stagnated between 1E+5 and 1E+7, rendering the results of this eigensolver unreliable. For this reason, we remained using full orthogonalization.

In Figure 4.1, convergence comparisons are presented for 10 different vectors using LOBPCG, Block Krylov Schur, and the randomized solver with orthogonalization frequencies 1 and 10. To the right of each convergence plot, the time taken by each solver to execute a specified number of iterations is displayed. These timings were averaged over 4 MPI processes. Four different matrices out of the 10 tested are shown: “cryg10000”, “delaunay\_n13”, “fv3”, and “mark3jac020”. In each of these convergence plots data is missing for the Block Krylov Schur solver when run to a large number of iterations due to the substantial about of time required for the solver to run. Results for LOBPCG are excluded

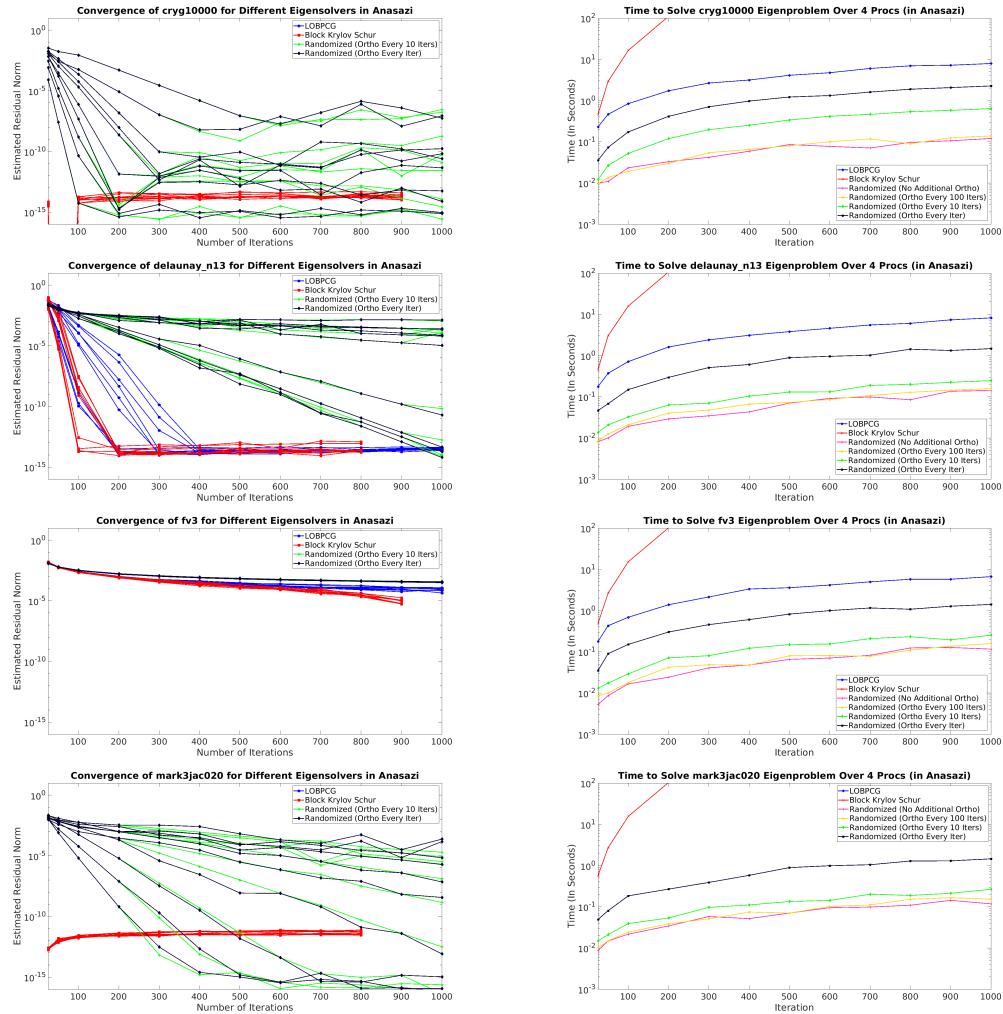


FIG. 4.1. The left side of this figure depicts the convergence of 10 residual vectors for the largest magnitude eigenpairs. Each line corresponds to a different eigensolver: LOBPCG (blue), Block Krylov Schur (red), and the randomized eigensolver with two distinct orthogonalization frequencies (black and green). On the right, timing comparisons between the different solvers are shown in seconds.

from the ‘‘mark3jac020’’ convergence plot, as it was one of the two matrices that returned a failure in the Cholesky decomposition. LOBPCG results are also absent for ‘‘cryg10000’’ because, while the LOBPCG method did not return a Cholesky failure, the residual norms of the returned solutions consistently remained in the thousands, regardless of the number of iterations. In general, LOBPCG and Block Krylov Schur demonstrate relatively fast convergence, while the randomized eigensolver may necessitate more iterations.

In the right column of Figure 4.1, timing results corresponding to the convergence plots on the left are displayed. These timings include results for the randomized solver with orthogonalization frequencies of 0 and 100. Interestingly, even if the randomized method undergoes orthogonalization at every iteration, it still runs approximately an order of magnitude faster than LOBPCG for the same number of iterations. On the other hand, the Block Krylov Schur method substantially increases in time as the number of iterations grows,

making it impractical for use with a large number of iterations or without restarting.

The following conclusions can be drawn from the presented results:

1. The randomized eigensolver, while sacrificing some accuracy, significantly reduces the time spent on each iteration when compared to LOBPCG and Block Krylov Schur, often by at least an order of magnitude.
2. Regularly orthogonalizing the basis of the randomized solver enables the method to achieve eigenpair estimates and their residual norms in cases where LOBPCG fails.
3. While the Block Krylov Schur method exhibits faster converge by using a fewer number of iterations to converge, the time it takes for the computation of the eigenpair estimations may be impractical; especially without restarting and when using a large basis size.

**5. Applications of the Randomized Solver: Spectral Graph Partitioning.** In this section, we discuss one of the potential applications of the randomized solver: spectral graph partitioning. Before presenting results and comparisons to other eigensolvers for this particular application, we will first provide a brief background as to what spectral graph partitioning entails.

**5.1. Spectral Graph Partitioning.** Graph partitioning is a well-studied problem in the field of Computer Science, particularly in the subfields of distributed systems and high-performance computing [13, 20]. The concept is a relatively straightforward one. Consider a graph  $G = (V, E)$ , where  $V$  is the set of all vertices, and  $E$  is the set of all edges connecting these vertices. The fundamental question is how to divide the vertices into  $k$  partitions of relatively equal size while reducing the amount of edges that connect vertices from different partitions. These cross-partitional edges are referred to as *edgecuts*.

Various techniques are used to compute the partitioning of a graph, but the two primary approaches are multilevel methods[7] and spectral methods[6]. In this context, we will focus on spectral graph partitioning, which uses the eigenvectors of the graph Laplacian to determine how to divide the graph. While the multilevel method is seemingly more prevalent, spectral graph partitioning is particularly well-suited for GPUs due to its reliance on linear algebra operations, allowing for a high level of parallelization.

The Sphynx software, located in the Zoltan2 package of Trilinos, implements a spectral graph partitioning method designed for use with multiple GPUs and takes advantage of distributed-memory systems [1, 18, 17]. At present, the default approach to computing the needed eigenvectors for partitioning is through the use of the LOBPCG iterative method located in Anasazi.

The randomized solver can be used to expedite the eigensolve. However, as previously mentioned, the speedup comes with the cost of decreased accuracy in the eigenvectors, which may lead to an increase in edgecuts when partitioning the matrix. Previous research suggests that the lower-quality eigenvectors generated by the randomized solver may actually provide a fewer of edgecuts for irregular graphs when compared to LOBPCG [5]. We define a graph to be irregular if the maximum degree of a node in the graph is significantly larger than the average degree across all nodes. For the purpose of this work, we label a graph as irregular if  $\frac{\text{Max Deg}}{\text{Avg Deg}} \geq 10$ .

**5.2. The Graph Laplacian.** The process of finding the graph Laplacian for spectral graph partitioning is outlined as follows:

1. Set the matrix  $A$  equal to the adjacency matrix of the graph  $G$  (ensuring no vertex has a self-loop, i.e., the diagonal of  $A$  is all zeros).
2. Let  $D$  be a diagonal matrix whose entries correspond to the row sums of  $A$ .
3. Obtain the graph Laplacian  $L = D - A$ .

For a general matrix  $M$ , some preprocessing is required to convert  $M$  to the adjacency matrix of a graph before finding its graph Laplacian. The details of this preprocessing is detailed in in Algorithm 4.

---

**Algorithm 4** Finding the Graph Laplacian of a General Matrix

---

**Require:** Matrix  $M \in \mathbb{C}^{n \times n}$

- |  |   |
|--|---|
| 1: $A = M + M^T$                                 | ▷ Symmetrize $M$ and assign to $A$            |
| 2: $A \leftarrow A(A \sim= 0) = 1$               | ▷ Set all non-zeros in $A$ to be 1            |
| 3: $\text{diag}(A) \leftarrow 0$                 | ▷ Make sure the diagonal entries of $A$ are 0 |
| 4: $D = \text{diag}(\text{rowsum}(A))$           | ▷ Set $D$ to be the degree matrix of $A$      |
| 5: <b>Return</b> the graph Laplacian $L = D - A$ |   |
- 

In the above algorithm, we set  $A$  to be the adjacency matrix of the graph formed from  $M$ . Another way to view an adjacency matrix is given by

$$A_{i,j} = \begin{cases} 0 & \text{if } i = j \text{ or if } M_{i,j} = 0 \\ 1 & \text{otherwise.} \end{cases} \quad (5.1)$$

$D$  is referred to as the degree matrix of  $A$ . The degree of a vertex in a graph is the number of edges attached to that vertex in an undirected graph. Since  $A$  is the adjacency matrix and we assume all graph edges to have a weight of one,  $D$  can be viewed as

$$D_{i,j} = \begin{cases} 0 & \text{if } i \neq j \\ \text{sum}(A_{i,:}) & \text{otherwise.} \end{cases} \quad (5.2)$$

From these two matrices, we form the *combinatorial* graph Laplacian

$$L = D - A. \quad (5.3)$$

For the randomized eigensolver, we seek the eigenvectors corresponding to the smallest eigenvalues of the *normalized* graph Laplacian denoted  $L_N$ .  $L_N$  is a scaled version of  $L$  such that its diagonal entries are equal to one, and the sum of all elements in each row is equal to zero.

$$L_N = I - D^{-\frac{1}{2}} L D^{-\frac{1}{2}}. \quad (5.4)$$

By definition, the eigenvalues of  $L_N$  are bounded between 0 and 2, with the first eigenvalue guaranteed to be 0. If an eigenvalue is 0, its associated eigenvector is trivial. Spectral graph partitioning computes  $d + 1$  eigenvectors to compute find  $2^d$  partitions of the input graph or matrix. The first trivial eigenpair is discarded, leaving the  $d$  remaining eigenvectors.

The  $d$  eigenvectors retained are used to map points onto a  $d$ -dimensional space. Subsequently, the multi-jagged method is employed to partition the points, each of which represents a distinct vertex in the graph  $G$  [4]. In using the multi-jagged method, the number of required eigenvectors is reliant on the number of partitions, denoted as  $p$ . If  $p$  is the number of partitions requested, the the number of required eigenvectors needed  $d$ , not including the trivial vector, is given by  $d = \lceil \log_2 p \rceil$ .

**5.3. Sphynx Graph Partitioner.** The Sphynx code within the Zoltan2 package of Trilinos serves as a graph partitioning software designed to determine the optimal way in which an input matrix can be partitioned across multiple processes and to compute the resulting number of edgecuts. It's worth noting that the current Sphynx code is not

equipped to handle matrices whose graphs have disconnected components. While addressing this issue is outside the scope of the current work, a potential solution involves applying spectral graph partitioning to each connected component separately. As of now, when the matrix is read in and converted into a graph, Sphynx identifies the largest connected component, and constructs the graph Laplacian from that. Any other connected components are disregarded. This limitation is one of the drawbacks of Sphynx. The Sphynx process is outlined in Algorithm 5.

---

**Algorithm 5** Sphynx Graph Partitioner

---

**Require:** Matrix  $M \in \mathbb{R}^{n \times n}$ , number of needed partitions  $p$ , number of iterations  $q$ , convergence tolerance  $\text{tol}$

- 1:  $nev = \lceil \log_2 p \rceil + 1$  ▷ Determine the number of needed eigenpairs
- 2:  $L_N = \text{NormalizedGraphLaplacian}(M)$
- 3:  $[X, \Theta] = \text{Eigensolver}(L_N, nev, q, \text{tol})$  ▷ Solve for the  $nev$  smallest eigenpairs using either LOBPCG or the randomized method
- 4: Remove the trivial eigenvector  $X = X(2 : \text{end})$
- 5: Partition the matrix using the multijagged method  $\Pi = \{V_1, V_2, \dots, V_p\} = \text{Multijagged}(X, p)$
- 6: **Return**  $\Pi$

---

When examining this algorithm, the immediate question may be asked: how do we find the smallest eigenpairs of  $L_N$  using the randomized solver, given that the solver was previously stated to only converge to the largest eigenpairs of  $L_N$ ? The answer is a straightforward one: we shift the eigenspectrum. All eigenvalues of  $L_N$  are bounded between 0 and 2, with the guaranteed trivial eigenvalue of 0. To shift the eigenspectrum of  $L_N$ , we flip the signs of the main diagonal before adding 2 to the diagonal

$$\hat{L}_N = 2I - L_N. \quad (5.5)$$

This shift operation causes all the eigenvalues that were close to 0 in  $L_N$  to be close to 2 in  $\hat{L}_N$ , and vice versa for the eigenvalues close to 2 in  $L_N$ . The benefit of this shift is that it only affects the eigenvalues, and does not change their corresponding eigenvectors. As a result, the randomized eigensolver can approximate the necessary eigenpairs without any complications.

**6. Partitioning Experimental Results.** Previous research on the comparison between eigenvectors obtained from LOBPCG and the eigenvectors produced by the randomized solver, and their impact on the number of edgecuts required to partition the matrix, was done by Espinoza et al[5]. In their study, edgecuts from both methods were compared on a variety of matrices. Initial results indicated that the use of LOBPCG resulted in fewer edgecuts for regular matrices. However, for irregular graphs, the randomized solver was more likely to produce fewer edgecuts. The study did not provide a definite explanation as to why the randomized eigensolver appeared to perform better for irregular matrices.

In previous experiments, LOBPCG was always run to either a tolerance of 1e-1 or a maximum of 1,000 iterations, and orthogonalization was performed at every step. The randomized solver, on the other hand, was run with a varying number of iterations: 1, 2, 4, 8, or 16. Orthogonalization was only being carried out just before the Rayleigh-Ritz problem was solved at the end. By not orthogonalizing the initial random vectors before the subspace iterations occur, the resulting eigenpairs would become highly inaccurate if too many iterations were performed. Adding that initial orthogonalization resulted in the

TABLE 6.1  
*Information about the normalized graph Laplacians for matrices obtained from SuiteSparse.*

Matrix Name	# Rows	Nnz	Max Deg	Avg Deg	Max/Avg Deg
G67	10,000	50,000	4	4	1.00
whitaker3	9,800	67,778	8	5.92	1.00
fv3	9,801	87,025	8	7.88	1.02
cryg10000	10,000	49,800	5	3.98	1.26
flowmeter0	9,669	67,391	10	5.97	1.68
delaunay_n13	8,192	57,286	12	5.99	2.00
mark3jac020	9,129	94,093	59	9.31	6.34
kmnist_norm_10NN	10,000	166,932	140	1.57	8.92
mycielskian13	6,143	1,233,885	3,071	199.86	15.37
vsp_data_and_seymourl	9,167	120,899	229	12.19	18.79
Erdos992	4,991	19,847	61	2.98	20.49
c-33	6,317	56,123	985	7.88	124.93
c-40	9,941	81,501	1,618	7.20	224.77

condition number of the basis not becoming an issue until eigenpairs begin to converge, and repeated directions start entering the basis.

Several questions were then raised:

1. How does the eigenspectrum of the graph Laplacian differ between regular and irregular graphs?
2. How does the number of edgecuts compare when running LOBPCG for the same number of iterations as the randomized method?
3. Why do the lower-accuracy eigenvectors yield a partitioning with fewer edgecuts than the higher accuracy ones obtained from LOBPCG?

Additionally, the decision was made to run Sphynx with the Block Krylov Schur eigensolver to provide a comparison against LOBPCG and the randomized solver.

**6.1. Regular vs Irregular Graph Laplacians: Eigenspectra.** The matrices described in Table 4.1, along with a few additional matrices, were used to illustrate the difference in eigenspectra between their respective graph Laplacians [10]. Further details on the graph Laplacians used can be found in Table 6.1.

In spectral graph partitioning,  $d$  eigenvectors can generate up to  $2^d$  partitions. Therefore, when comparing the eigenspectra of different graph Laplacians, only the 20 smallest eigenvalues were considered. MATLAB's `eig` function was utilized for this purpose. The smallest eigenvalue and its corresponding eigenvector, guaranteed to be trivial, are therefore excluded when analyzing the eigenspectra in Figure 6.1. From this figure, a pattern emerges: the further the smallest eigenvalues are from 0, the higher the degree of irregularity in the graph.

**6.2. Changes in the Number of Edgecuts.** The second question explored was how the number of edgecuts varied between using LOBPCG with Jacobi preconditioning, Block Krylov Schur, and the randomized solver with different orthogonalization frequencies for the same number of iterations. Similarly to Section 4, full orthogonalization was done in LOBPCG and Block Krylov Schur, and all orthogonalization done use the SVQB technique. No restarting or preconditioning was used for Block Krylov Schur or the randomized method. All tests were conducted using the Sphynx software, which called the eigensolvers from the Anasazi package in Trilinos.

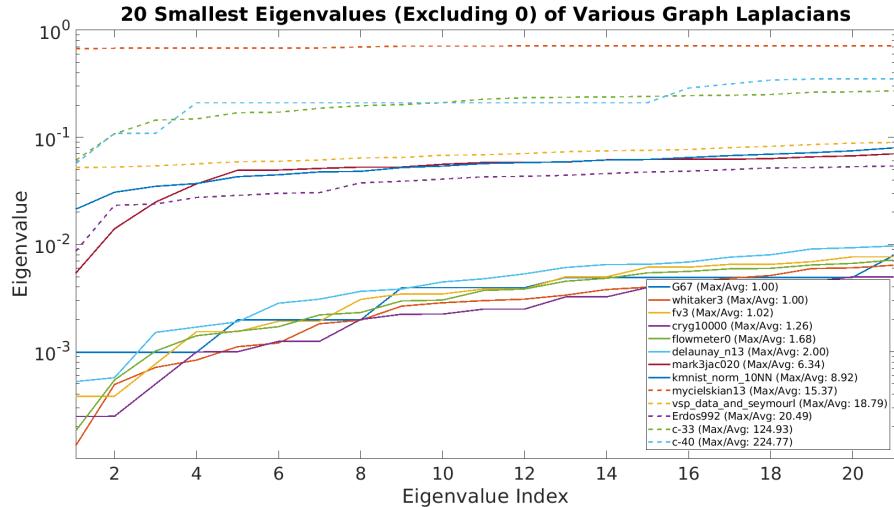


FIG. 6.1. The 19 eigenvalues closest to zero, but excluding zero, for a variety of graph Laplacians. The dashed lines indicate that the graph Laplacian being plotted is irregular.

The initial observation indicated the amount of orthogonalization done in the randomized solver had minimal impact on the number of edgecuts in most cases, nor on the convergence of the solver itself. One possible explanation for this is that the eigenpairs do not converge quickly, resulting in the repeated directions that would cause the condition number to spike have not yet been introduced into the basis. However, when no orthogonalization was added, the randomized solver crashed at  $\sim 1,000$  iterations. Consequently, we only present the convergence results for the randomized solver with **Orthogonalization Frequency**=100. Additionally, results for orthogonalization frequencies of 10 and 1 are not provided since the primary aim of this test is to optimize the speed of the eigensolver.

Figure 6.2 shows the convergence of 4 out of 13 graph Laplacians formed from the matrices “G67”, “kmnist\_norm\_10NN”, “vsp\_data\_and\_seymourl”, and “c-33” matrices. These specific matrices were chosen to represent a range of irregularities ( $\frac{\text{Max Deg}}{\text{Avg Deg}}$ ): 1, 8.92, 18.79, and 124.93, respectively. Contrary to previous work indicating that the vectors returned from LOBPCG offer better edgecuts for regular graphs, this does not appear to be the case overall. In all four examples shown, the resulting eigenvectors from LOBPCG, Block Krylov Schur, and the randomized solver for the same number of iterations seem comparable. Although not shown, this trend remains consistent with the other nine matrices as well.

As indicated in the highly irregular case of “c-33,” the randomized solver appears to produce slightly better edgecuts than the other methods the most of the time, similar to what was observed with the “c-40” matrix results. Further investigation is warranted, particularly with larger matrices that exhibit higher degrees of irregularity ( $> 100$ ), to determine if this pattern remains consistent or if the relative gap between the number of edgecuts increases.

The vectors resulting from the Block Krylov Schur solver appear to behave similarly to LOBPCG in terms of both convergence rate and the number of edgecuts required. However, when comparing the timings of the solvers in Sphynx, the results were nearly identical to the timing plots shown in Figure 4.1. This suggests that even though using the Block Krylov Schur method produces good edgecuts, the computational cost is not justified.

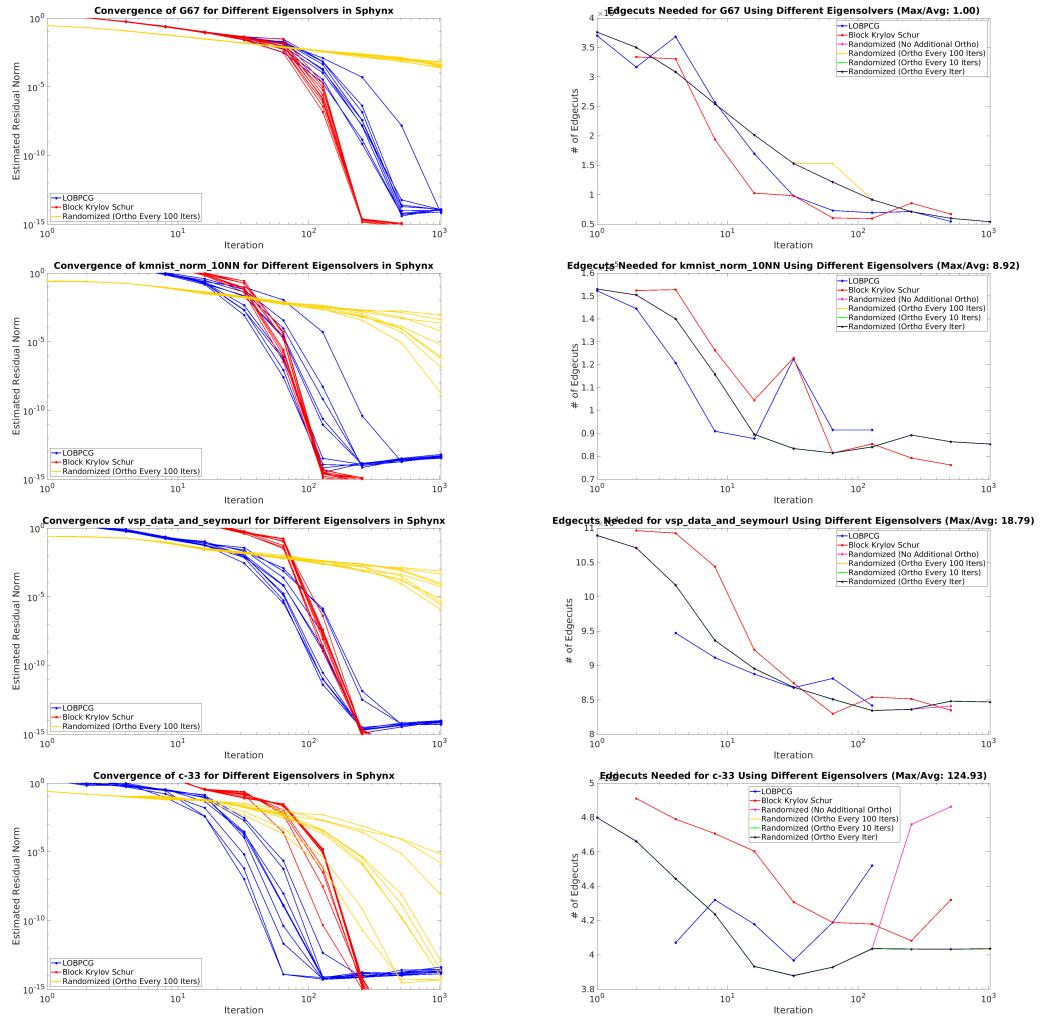


FIG. 6.2. The left column illustrates the convergence of the eigenpairs from the graph Laplacian of 4 different matrices using 3 eigensolvers from Anasazi. The right column displays the corresponding number of edgecuts required to partition the matrix into 64 parts given the returned eigenvectors. In the convergence plots, the residual norms for the first eigenpair have been omitted since it is not used in spectral graph partitioning.

**7. Discussion.** If it is indeed the case that Laplacians with higher degrees of irregularity yield better edgecut results when being partitioned with eigenvectors from the randomized solver, the next question to explore is why this phenomenon occurs. It raises the possibility that vectors with lower accuracy might work better than those with higher accuracy, essentially increasing the “flexibility” of the partition. This question remains open for future investigation. Regardless, the results suggest that the number of edgecuts is more reliant on the accuracy of the eigenvectors than on the specific eigensolver is being used.

Furthermore, scalability tests are crucial for evaluating the performance of the randomized solver with different orthogonalization frequencies. These tests should be used to determine the cost-benefit trade-offs of using it over established methods like LOBPCG, Block Krylov Schur, Davidson, and potentially other solvers. Scalability tests should not

only consider the actual size of the matrices, but also the number of processes involved as well as GPU performance. This analysis may provide further insight into the effectiveness and efficiency of the randomized solver when using it in various computational scenarios.

Additional open research directions for the application of the randomized solver in Sphynx involve investigating the potential performance gains that can be achieved by integrating the randomized eigensolver with a preconditioning method to speed up convergence. Another route of exploration is the comparison of the solver's performance with that of the established graph partitioner, ParMetis. Additionally, there is an interest in assessing how the randomized eigensolver in Sphynx performs on well-structured Galeri problems. These problems are left for future work.

**8. Conclusion.** In this project, a new eigensolver has been added to the Anasazi package in Trilinos, the changes of which have been submitted as a pull request on GitHub. Furthermore, we perform a comparison on the convergence and timings of the randomized solver with different orthogonalization frequencies against the LOBPCG and Block Krylov Schur methods. The results demonstrate that the randomized solver, even when orthogonalizing at every iteration, was still on average an order of magnitude faster than LOBPCG when stopping at the same number of iterations, even though the convergence rate suffered as a result.

The randomized solver was also evaluated using the Sphynx graph partitioner for spectral partitioning. The convergence rates were once again compared to LOBPCG and Block Krylov Schur, and the resulting edgecuts from the partitioning were plotted. These plots indicated that the 3 eigensolvers were comparable when stopping at the same number of iterations, particularly for graph Laplacians with small degrees of irregularity.

## REFERENCES

- [1] S. ACER, E. G. BOMAN, C. A. GLUSA, AND S. RAJAMANICKAM, *Sphynx: A parallel multi-gpu graph partitioner for distributed-memory systems*, Parallel Computing, 106 (2021), p. 102769.
- [2] C. G. BAKER, U. L. HETMANIUK, R. B. LEHOUCQ, AND H. K. THORNQUIST, *Anasazi software for the numerical solution of large-scale eigenvalue problems*, ACM Trans. Math. Softw., 36 (2009).
- [3] M. BEIKI AND L. B. PEDERSEN, *Eigenvector analysis of gravity gradient tensor to locate geologic bodies*, Geophysics, 75 (2010), pp. I37–I49.
- [4] M. DEVECI, S. RAJAMANICKAM, K. D. DEVINE, AND V. ÇATALYÜREK, *Multi-jagged: A scalable parallel spatial partitioning algorithm*, IEEE Transactions on Parallel and Distributed Systems, 27 (2016), pp. 803–817.
- [5] H. ESPINOZA, *Randomized methods for graph partitioning applications*, California State University, Master's Thesis (2023).
- [6] L. HAGEN AND A. KAHNG, *New spectral methods for ratio cut partitioning and clustering*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 11 (1992), pp. 1074–1085.
- [7] B. HENDRICKSON AND R. LELAND, *A multi-level algorithm for partitioning graphs*, in Supercomputing '95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, Dec 1995, pp. 28–28.
- [8] A. HILLIGES, C. MEHL, AND V. MEHRMANN, *On the solution of palindromic eigenvalue problems*, in Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS), Jyväskylä, Finland, 2004, p. 108.
- [9] A. V. KNYAZEV, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM Journal on Scientific Computing, 23 (2001), pp. 517–541.
- [10] S. P. KOLODZIEJ, M. AZNAVEH, M. BULLOCK, J. DAVID, T. A. DAVIS, M. HENDERSON, Y. HU, AND R. SANDSTROM, *The suitesparse matrix collection website interface*, Journal of Open Source Software, 4 (2019), p. 1244.
- [11] J. MACDONALD, *Successive approximations by the rayleigh-ritz variation method*, Physical Review, 43 (1933), p. 830.
- [12] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, second ed., 2003.

- [13] K. SCHLOEGEL, G. KARYPIS, AND V. KUMAR, *Parallel multilevel algorithms for multi-constraint graph partitioning*, in Euro-Par 2000 Parallel Processing, A. Bode, T. Ludwig, W. Karl, and R. Wismüller, eds., Berlin, Heidelberg, 2000, Springer Berlin Heidelberg, pp. 296–310.
- [14] A. STATHOPOULOS AND K. WU, *A block orthogonalization procedure with constant synchronization requirements*, SIAM Journal on Scientific Computing, 23 (2002), pp. 2165–2182.
- [15] G. W. STEWART, *A krylov-schur algorithm for large eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 23 (2002), pp. 601–614.
- [16] THE TPETRA PROJECT TEAM, *The Tpetra Project Website*, 2020.
- [17] THE TRILINOS PROJECT TEAM, *The Trilinos Project Website*, 2020 (accessed May 22, 2020).
- [18] THE ZOLTAN2 PROJECT TEAM, *The Zoltan2 Project Website*, 2020.
- [19] F. W. VOLLMER, *An application of eigenvalue methods to structural domain analysis*, Geological Society of America Bulletin, 102 (1990), pp. 786–791.
- [20] W. ZHANG, Y. CHEN, AND D. DAI, *Akin: A streaming graph partitioning algorithm for distributed graph storage systems*, in 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), May 2018, pp. 183–192.

## ADAPTIVE RELAXATION METHODS BASED ON PATCH REUSE FOR EVOLVING LINEAR SYSTEMS

ALEXEY VORONIN\*, RAYMOND S. TUMINARO†, LUKE N. OLSON‡, AND SCOTT  
MACLACHLAN §

**Abstract.** This report introduces a novel patch-based relaxation approach to improve multigrid setup times when addressing a series of related linear systems. These systems often stem from time-dependent partial differential equations (PDEs), nonlinear solvers, or continuous design challenges like topology optimization. Rather than bearing the computational burden of extracting and factoring patches anew, our method capitalizes on reusing patch factorizations from prior linear system solutions, where appropriate. By leveraging relaxation concepts, we introduce a metric that selectively pinpoints a limited subset of patches for updating or factoring between consecutive linear system solutions. This strategy can achieve optimal convergence rates while significantly reducing setup costs when the system operator undergoes gradual changes from one solve to another. Numerical results, grounded in variable coefficient diffusion problems, validate the efficiency and effectiveness of this approach. The report also highlights the algorithm's potential applicability to more complex problems.

**1. Introduction.** Effective relaxation methods are the cornerstone of a robust multigrid solver. For many PDE systems based on self-adjoint operators, Jacobi and Gauss-Seidel relaxation provide a good trade-off between convergence rate and cost per iteration. However, for more complex, especially multi-variable systems, more sophisticated domain-decomposition-based relaxation methods are required for robust mesh-independent convergence [2, 4, 3, 15]. While these domain decomposition methods, commonly referred to as patch-relaxation, provide superior convergence properties [7, 9, 12], they also come at a higher computational cost, requiring extraction of patch sub-matrices from the system matrix and computing their numerical inverse. In the case of a simple element-based scalar patch domain on a quadrilateral (hexahedral) mesh, the cost of inverting each patch scales as  $O((p + 1)^{3d})$ , where  $d$  is the dimension of a mesh (e.g., 2 for quadrilateral, 3 for hexahedral) and  $p$  is the order of the discretization technique. This cost can quickly become the most time-consuming part of the solution of the PDE problem, especially when the matrix coefficients continuously change from solve to solve. There have recently been several research efforts focused on reducing the computational and storage costs associated with Vanka relaxation [5, 10, 16].

Motivated by these challenges, our research introduces a novel approach to updating the preconditioner in evolving PDE problems (see for example [1, 2, 3, 6, 11, 13]) where the system operator undergoes gradual changes from solve to solve. We propose a relaxation-property-based measure that allows us to identify patches where inverses from the previous solve poorly approximate inverses associated with the current solve. Only these identified patches must then be updated/factored to recover optimal convergence rates, thereby minimizing solver setup cost between solves. The numerical results presented in the report illustrate the algorithm's convergence in different scenarios, along with a cost analysis. While this report focuses on variable coefficient diffusion problems, the same technique naturally extends to more complex heat-flow, fluid-flow, and elasticity-type problems.

The remainder of this report is structured as follows. In Section 2, we detail a standard additive patch relaxation algorithm. Section 2.2 introduces the patch-updating algorithm and analyzes its computational cost. Section 3 presents an artificial variable coefficient diffu-

---

\*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

sion problem to evaluate the robustness of the patch-selection algorithm. The critical steps of the algorithm, as applied to diffusion problems, are illustrated in Section 3.2, followed by the convergence results in Section 3.3. Finally, Section 4 concludes the report and offers insights into how these findings might be extended to address more sophisticated problems.

**2. Patch-based Relaxation.** Patch-based relaxation methods divide the matrix into smaller pieces, known as patches. Typically, these patches correspond to overlapping groups of adjacent degrees of freedom (DoFs) within the computational domain. The selection criteria for these patches often vary depending on the specific problem and the discretization approach employed. Nevertheless, a rough guideline is that more overlap between adjacent patches generally leads to faster convergence rates, though the cost to setup and apply the preconditioner also grows. Within the scope of this report, our primary focus is on cell-restricted patches for scalar problems. These patches overlap with their neighboring patches only at cell boundaries. However, our subsequent discussions on patch reuse can be extrapolated to other patch relaxation forms, including vertex-star and cell-centered patches [5, 8].

Section 2.1 details the mathematical framework for a general patch relaxation method, while Section 2.2 offers novel insights into identifying suboptimal patches for problems sharing the same grid and discretization.

**2.1. Background.** In this work, we use the additive form of Vanka relaxation relaxation [14], although the patch reuse techniques also extend to a multiplicative variant of Vanka. The mathematical representation of a single additive Vanka relaxation sweep that employs  $n_p$  patches is expressed as

$$R^{-1} = \sum_{i=1}^{n_p} (V^{(i)})^T W^{(i)} \left( V^{(i)} A (V^{(i)})^T \right)^{-1} V^{(i)},$$

where  $A$  is the full system matrix being solved,  $V_\ell^{(i)}$  is a rectangular injection matrix that transforms vectors defined over the entire domain to just a vector defined over the  $i^{th}$  patch's degrees-of-freedom, and  $W^{(i)}$  is a diagonal weight matrix where each diagonal entry is equal to the reciprocal of the number of patches that contain the associated degree of freedom (DoF).

Figure 2.1 provides an example of cell-restricted patches for orders 2 and 3 Lagrange elements on a quadrilateral mesh.

Construction and application of the inverses of patch matrices,  $(V^{(i)} A (V^{(i)})^T)^{-1}$ , often constitutes the most computationally demanding part of the multigrid method. These inverses are usually pre-computed during the initial setup phase. Even when only part of the domain and numerical operator changes, all Vanka patch inverses are typically recomputed. Obviously, this expensive redundant work is wasteful and should be avoided. While the operator may change throughout the domain in other cases, it is not uncommon that significant changes are only restricted to small sub-regions of the domain. In these scenarios, we would also hope to reuse most of the patch inverses from the previous linear solve, but this requires devising an algorithm to identify the faulty or deficient patch inverses.

**2.2. Patch Re-use Algorithm.** A patch reuse algorithm is developed to efficiently address numerical applications where the mesh and discretization are constant, but the coefficients of the operators evolve over time or iterations. This is common in both time-dependent and continuous design steady-state linear systems. For instance, consider two

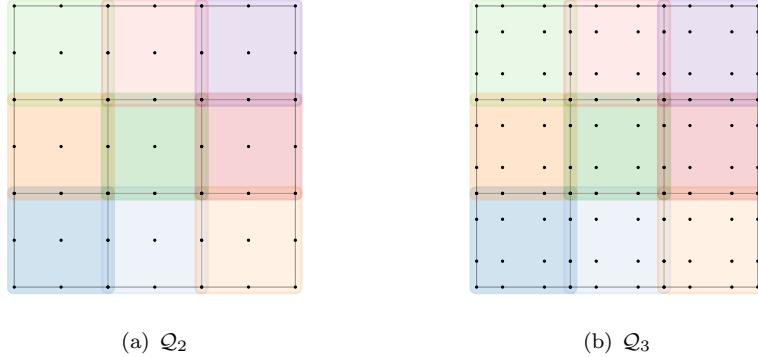


FIG. 2.1. *Cell-restricted patches on  $3 \times 3$  quadrilateral mesh for second and third order Lagrange elements. Each color corresponds to a unique patch and each black dot is a unique DoF.*

successive linear systems

$$A_0 x_0 = b_0 \quad \text{and} \quad (2.1)$$

$$A_1 x_1 = b_1, \quad (2.2)$$

where (2.1) represents the initial system being solved and (2.2) denotes the subsequent system. While  $A_0$  and  $A_1$  have the same dimensions and sparsity patterns, the actual matrix values differ. Let us denote  $R_0^{-1}$  as the Vanka relaxation operator tailored for the (2.1) system. Specifically, one sweep of relaxation improves the  $k^{th}$  approximate solution  $x_0^{(k)}$  via

$$x_0^{(k+1)} = x_0^{(k)} + R_0^{-1}(b_0 - A_0 x_0^{(k)}).$$

Our primary objective is to devise an algorithm that can adaptively modify the  $R_0^{-1}$  relaxation, ensuring that it offers smoothing properties for the new  $A_1$  system, akin to how a true Vanka relaxation operator  $R_1^{-1}$  would behave.

In Section 2.2.1, we introduce a specific measure employed to gauge the efficacy of the  $R_0^{-1}$  relaxation as applied to the  $A_1 x_1 = b_1$  system. Our end goal is to efficiently pinpoint the patches of  $R_0^{-1}$  that warrant adjustments to better fit the  $A_1 x_1 = b_1$  system. Section 2.2.2 describes an algorithm that leverages the measure defined in Section 2.2.1. Lastly, an analysis of computational costs, crucial for understanding the feasibility of our algorithm, is presented in Section 2.2.3.

**2.2.1. Approximation Property Measure.** The error propagation operator associated with solving the  $A_0$  system using  $R_0^{-1}$  is given by

$$e_0^{(k+1)} = (I - R_0^{-1} A_0) e_0^{(k)}$$

where  $e_0^{(k)} = x_0^{(k)} - A_0^{-1} b_0$ . The corresponding error propagation operators for solving the  $A_1$  system are

$$e_1^{(k+1)} = (I - R_1^{-1} A_1) e_1^{(k)} \quad \text{and} \quad \hat{e}_1^{(k+1)} = (I - \hat{R}_0^{-1} A_1) \hat{e}_1^{(k)}$$

using  $R_1^{-1}$  and  $\hat{R}_0^{-1}$  respectively. Here,  $\hat{R}_0^{-1}$  is an approximation to  $R_1^{-1}$  where only a few patch matrices have been re-computed to locally update the  $R_0^{-1}$  operator. Thus, we seek

Symbol	Description
$A_0, A_1$	Linear system operators
$R_0^{-1}$	Vanka relaxation operator for the $A_0$ system
$R_1^{-1}$	Vanka relaxation operator for the $A_1$ system
$\hat{R}_0^{-1}$	$\hat{R}_0^{-1}$ partially updated with $A_1$ patches
$y_r$	Random vector for approximation property measure
$y_0, y_1$	Approximate solutions
$\alpha$	Percentage of DoFs with biggest correction error
$\hat{d}$	Indicator vector for defective patches
$V$	Binary mapping operator
$n_p$	Number of patches
$n_{dp}$	Number of defective patches
$s$	Size of each patch
Superscript $G$	denotes vector variable in “patch-space” (e.g. $u^G = Vu$ )
$\xi$	constant related to extracting patches
$\iota$	Constant related to inverting patches
$\beta$	Constant related to patch solves (e.g., matrix-vector multiplication)
$\Omega$	Cost of sparse matrix-vector multiplications with $V$

TABLE 2.1  
Key symbols with their respective descriptions.

to evaluate how well  $I - \hat{R}_0^{-1}A_1$  mimics the behavior of  $I - R_1^{-1}A_1$ . If the approximation  $\hat{R}_0^{-1}$  is subpar, we aim to pinpoint which patches should be re-computed to refine  $\hat{R}_0^{-1}$  approximation. A potential measure to consider is:

$$\left\| (I - \hat{R}_0^{-1}A_1)(I - R_1^{-1}A_1)^{-1} - I \right\|_2, \quad (2.3)$$

which is identically zero when  $\hat{R}_0^{-1} = R_1^{-1}$ . However, this formula is not computationally attractive as it requires  $R_1^{-1}$  to be constructed and applied to  $A_1$  - precisely what we are trying to avoid with the  $\hat{R}_0^{-1}$  approximation. Moreover, it doesn't clearly show which patches need updating to optimize  $\hat{R}_0^{-1}$  and necessitates a costly eigenvalue computation. Instead, we seek a more computationally attractive alternative that can also be used to pinpoint patches that should be re-computed. One potential simplification could involve assessing the behavior of the matrix in (2.3) when applied to a single random vector,  $y_r$ . While this avoids the eigenvalue calculation, it still requires the construction of the matrix  $R_1^{-1}$ .

To devise an inexpensive estimate that altogether avoids the computation of  $R_1^{-1}$ , we consider one iteration for each relaxation method using a non-zero starting vector  $y_r$  and a zero right-hand-side. That is,

$$A_0y_r = 0 \quad \text{and} \quad A_1y_r = 0$$

where the initial guess  $y_r$  is improved by a single relaxation iteration

$$y_0 = (I - R_0^{-1}A_0)y_r \quad \text{and} \quad y_1 = (I - \hat{R}_0^{-1}A_1)y_r$$

for the two respective systems. To evaluate the properties of  $I - \hat{R}_0^{-1}A_1$ , we examine the

relative changes in the solution vectors  $y_0$  and  $y_1$  leveraging the formula

$$\begin{aligned} y_1 - y_0 &= \left[ (I - \hat{R}_0^{-1} A_1) - (I - R_0^{-1} A_0) \right] y_r \\ &= \left[ (I - \hat{R}_0^{-1} A_1)(I - R_0^{-1} A_0)^{-1} - I \right] (I - R_0^{-1} A_0) y_r \\ \|y_1 - y_0\|_\alpha &= \left\| \left[ (I - \hat{R}_0^{-1} A_1)(I - R_0^{-1} A_0)^{-1} - I \right] y_0 \right\|_\alpha. \end{aligned} \quad (2.4)$$

From Equations (2.3) and (2.4), we note that the term  $(I - R_1^{-1} A_1)^{-1}$  has been replaced by the term  $(I - R_0^{-1} A_0)^{-1}$ . This avoids the explicit computation of  $R_1^{-1}$  and yields a reasonable approximation. With an additional simplification, we arrive at a simple and inexpensive formula

$$\frac{\|y_1 - y_0\|_\alpha}{\|y_0\|_\alpha} \leq \left\| (I - \hat{R}_0^{-1} A_1)(I - R_0^{-1} A_0)^{-1} - I \right\|_\alpha, \quad (2.5)$$

which implies that a large relative difference between  $y_0$  and  $y_1$  indicates that  $\hat{R}_0^{-1}$  needs further modification. Utilizing (2.5), we can compute bounds on the smoothing property of  $\hat{R}_0^{-1}$  on  $A_1$  relative to the smoothing property of  $\hat{R}_0^{-1}$  on  $A_0$ . While not completely rigorous, we have found it useful to compute DoFs-wise quantities to identify the DoFs that have been most impacted by the change in the operator. Specifically, the magnitude of the  $j^{th}$  component of the relative difference between  $y_0$  and  $y_1$  guides what DoFs and, ultimately, what patches should be updated. Although this currently lacks robust theoretical backing, our empirical observations show that the approach is effective in practical scenarios.

More robust alternative measures, which operate on patches directly, are detailed in Section 5. However, given the space and time constraints in this report, the numerical results for these measures will be presented in future publications.

**2.2.2. Algorithm.** This section presents Algorithm 1, designed to locate and replace defective patches. This algorithm employs measure (2.5) to achieve this purpose. The algorithm operates without requiring simultaneous access to system operators  $A_0$  and  $A_1$ , instead using the relaxed vector  $y_0$  to infer the action of  $R_0^{-1}$  on the  $A_0$  operator. However, it mandates that  $A_1$  and  $R_0^{-1}$  be accessible concurrently, a requirement usually met in Vanka relaxation. Algorithm 1 consists of several steps. It computes the action of  $R_0^{-1}$  on the residuals associated with  $A_1 y_r = 0$  (line 9), then uses the measure to pinpoint the top  $\alpha\%$  of DoFs exhibiting the largest correction deviation (line 10). Identifying the patches to which these DoFs belong involves scattering the indicator vector  $\hat{d}$  using the binary mapping operator  $V$  (lines 13-17), which is already present from the  $R_0^{-1}$  setup. Next, the patches needing replacement are determined (lines 19-25) and replaced using the standard Vanka setup algorithm (line 27).

**2.2.3. Cost Analysis.** This section evaluates the computational cost in two scenarios: completely rebuilding all Vanka patches and updating selected Vanka patches using Algorithm 1. The cost of rebuilding all patches includes patch extraction and patch inversion. The complexity of extracting each patch from the system operator  $A$  depends on the discretization order (i.e., the number of DoFs in a patch). We denote this complexity per patch as  $\xi$ . The complexity of inverting each patch is proportional to the cube of its size  $s$ . This step can be approximated as  $(\iota s^3)$ , where  $\iota$  is a constant related to inverting patches. The total complexity of rebuilding all  $n_p$  Vanka patches becomes  $n_p(\xi + \iota s^3)$ .

**Algorithm 1** Defective Patch Selection and Replacement Algorithm

---

```

1: Input:  $A_1$ , updated/new linear system
2:       $R_0^{-1}$ , Vanka relaxation operator based on  $A_0$ 
3:       $y_r$ , random vector of same dimension as column space of  $A_i$ 
4:       $y_0$ , pre-computed action of  $(I - R_0^{-1}A_0)$  on  $y_r$ 
5:       $\alpha$ , percent cutoff; sets the cutoff for “unacceptable” error
6: Output:  $\hat{R}_0^{-1}$ , updated Vanka relaxation operator
7:       $y_1$ , action of  $(I - \hat{R}_0^{-1}A_1)$  on a random vector
8:
9:  $y_1 = (I - \omega R_0^{-1}A_1)y_r$                                 // Apply the  $R_0^{-1}$  relaxation to  $A_1y_r = 0$  system
10:  $d = |(y_1 - y_0)/y_0|$                                          // DoF-wise normalized difference
11:
12: # { Get IDs of “bad” patches }
13: bad_dofs = { get top  $\alpha\%$  of DoFs with the highest  $d$  }
14:  $\hat{d} = [0 \dots 0]$                                          // allocate indicator vector of same size as  $y_1$ 
15:  $\hat{d}[bad\_dofs] = 1$                                          // mark defective DoFs
16:  $V \leftarrow get\_scatter(R_0^{-1})$                            // extract binary mapping operator from  $R_0^{-1}$ 
17:  $\hat{d}^G = V\hat{d}$                                          // scatter the defect test vector to respective patches
18:
19: bad_patches = []
20: for  $i = 0, \dots, n_{patches} - 1$  do                         // mark patches for replacement
21:      $idx = get\_patch\_indc(V, i)$                              // get patch indices
22:      $\hat{d}_i^G = \hat{d}^G[idx]$                                      // extract  $y_0$  patch vector
23:     if  $\|\hat{d}_i^G\|_\infty > 0$                                  // if patch contains bad DoFs, record it
24:         bad_patches  $\leftarrow i$ 
25: end for
26:
27:  $\hat{R}_0^{-1} = update\_patches(R_0^{-1}, bad\_patches)$  // Update bad Patches

```

---

The total cost of applying Algorithm 1 includes:

- **Additional Relaxation Sweep:** This step involves patch solves and two sparse matrix-vector multiplications with the binary mapping operator  $V$ . The complexity of this operation is estimated as  $n_p(\beta s^2) + 2\Omega$ , where  $\beta$  is a constant related to patch solves, and  $\Omega$  is the cost of sparse matrix-vector multiplications with  $V$ .
- **Scattering the Indicator Vector:** The complexity of scattering the indicator vector  $\hat{d}$ , which identifies defective patches, is  $\Omega$ .
- **Updating Defective Patches:** The cost of updating defective patches, referred to as  $dp$ , is described as  $n_{dp}(\xi + \iota s^3)$ , where  $n_{dp}$  is the number of defective patches.

Hence, the total cost of the algorithm is  $n_{dp}(\xi + \iota s^3) + n_p\beta s^2 + 3\Omega$ .

The point where updating selected patches becomes beneficial can be determined by equating the total costs (assuming convergence is not adversely affected), yielding an equa-

tion for the number of defective patches.

$$\begin{aligned}
n_{dp}(\xi + \iota s^3) + n_p \beta s^2 + 3\Omega &= n_p(\xi + \iota s^3) \\
n_{dp}(\xi + \iota s^3) &= n_p(\xi + \iota s^3) - n_p \beta s^2 - 3\Omega \\
n_{dp} &= \frac{n_p(\xi + \iota s^3) - n_p \beta s^2 - 3\Omega}{\xi + \iota s^3} \\
&= n_p - \frac{n_p \beta s^2 + 3\Omega}{\xi + \iota s^3} \\
&= n_p \left( 1 - \frac{\beta s^2 + 3\Omega / n_p}{\xi + \iota s^3} \right) \\
&= n_p \left( 1 - \frac{\{\text{cost of identifying bad patches per patch}\}}{\{\text{cost of extracting and inverting a patch}\}} \right)
\end{aligned} \tag{2.6}$$

Equation (2.6) implies that updating only selected patches is advantageous if patch extraction and inversion are expensive relative to relaxation cost, which is usually the case. A thorough investigation of constants and the extraction and scatter operations' costs is necessary for sharp bounds on  $n_{dp}$ .

**2.3. Preconditioner.** Patch relaxation methods are usually used within geometric or algebraic multigrid methods (abbreviated as GMG and AMG). The standard two-level error-propagation operator for a multigrid cycle is given by

$$M_0 = (I - \omega R_0^{-1} A_0)(I - P^T A_c^{-1} P)(I - \omega R_0^{-1} A_0) \tag{2.7}$$

where  $A_0$  is the fine-level system matrix,  $R_0^{-1}$  represents relaxation operator applied to  $A_0$  with a damping parameter  $\omega$ ,  $P$  is the interpolation between levels and  $A_c$  is the coarse-grid operator.

The algorithm outlined in Section 2.2.2 easily extends to all discretization matrices within the GMG multigrid hierarchy. For AMG, the changes in the system operator can result in different graph coarsening patterns and, therefore, a different number of unknowns on the coarse grid. This can limit the patch-reuse approach to only the fine level of the AMG hierarchy. This issue could be mitigated by reusing the strength-of-connection matrix. However, this approach can lead to diminished convergence rates and needs further study. This report focuses on 2-level AMG hierarchies, thereby avoiding the described problem.

### 3. Numerical Results.

**3.1. Problem Setup.** Instead of grappling directly with time-dependent PDEs or continuous design problems, we propose starting with variable coefficient diffusion problems as a surrogate for more sophisticated problems.

We express the variable coefficient diffusion problem as:

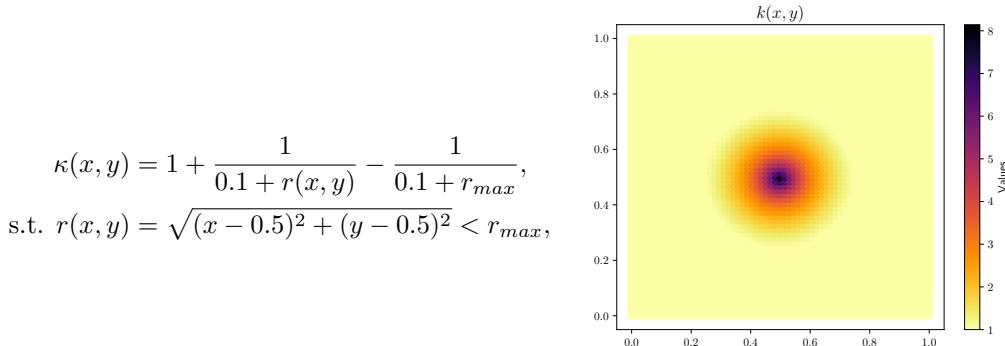
$$\nabla \cdot (\kappa(x, y) \nabla u) = 1, \quad u = 0 \text{ on } \Gamma, \tag{3.1}$$

where we define linear systems with different values of  $\kappa$  as:

$$\kappa_{const} = 1 : A_0 x_0 = b_0, \tag{3.2}$$

$$\kappa(x, y) > 1 : A_1 x_1 = b_1. \tag{3.3}$$

This report restricts the focus to diffusion problems on the 2D structured quadrilateral unit square mesh using the quadratic basis ( $Q_2$ ). The variable coefficient diffusion operator  $A_1$  is only perturbed within a disc using the formula



where  $r_{max} = 0.25$  represents the radius of perturbation and  $(x, y)$  denote the grid coordinates. The function  $\kappa(x, y)$  is visualized to the right of the formula.

**3.2. Visualizing the Measure.** We apply Algorithm 1 to the two diffusion operators defined in Equations (3.2) and (3.3) and utilize cell-restricted Vanka relaxation to solve the systems. We aim to test if the same Vanka configuration can retain analogous smoothing properties when used with the  $A_1$  operator. The problems are discretized with  $Q_2$  elements on  $80 \times 80$  quadrilateral mesh.

Initially, we visualize the normalized difference vector  $d$  (line 10 of Algorithm 1) in Figure 3.1. As anticipated, not all DoFs are influenced by the perturbation in the  $A_1$  operator. If we modify the patches corresponding to the 24% of affected DoFs, it should enable us to transform  $R_0^{-1}$  into  $\hat{R}_1^{-1}$  Vanka relaxation operator at a reduced computational expense when compared to forming  $R_1^{-1}$ .

Using Algorithm 1, we pinpoint the affected DoFs at various cutoffs. We identify the related patches by scattering the indicator vector  $\hat{d}$  and marking patches where  $\hat{d}^G$  is non-zero for replacement. Figure 3.2 showcases the selected patches for different thresholds. The patches nearest the domain's center, where the system was most perturbed, are prioritized. The selection spreads outwards from the domain's center as the cutoff expands. The marked patches that appear outside the perturbation regions are at least partially perturbed because part of the patch is inside the region touching the boundary. Increasing  $\alpha$  beyond 30% does not result in updating any other patches since all non-zero  $d^i$  values have been exhausted.

**3.3. Convergence Analysis.** This section investigates the impact of partial patch updates on the convergence behavior of a multigrid solver. Specifically, we analyze the convergence histories of FGMRES (Flexible Generalized Minimum RESidual method) preconditioned with a two-level smoothed-aggregation algebraic multigrid (AMG) method. This setup employs a single pre- and post-iteration of the cell-restricted patch relaxation on the fine grid.

For the initial system  $A_0 x_0 = b_0$ , the solver converges in 10 iterations, as depicted by the solid blue line in the chart below. However, when applying the same preconditioner to a modified system  $A_1 x_1 = b_1$ , the number of iterations needed for convergence increases to 35, as represented by a dashed red line. Next, we explore whether the patch-selection algorithm (as detailed in Algorithm 1) can be utilized to improve this convergence behavior.

When we attempt to update the  $R_0^{-1}$  using patches derived from the  $A_1$  operator without a corresponding update to the AMG hierarchy, the resulting solver fails to converge. This result is not shown in the chart but is important, as it emphasizes the need to rebuild the multigrid hierarchy using the  $A_1$  operator. This updated configuration is denoted as

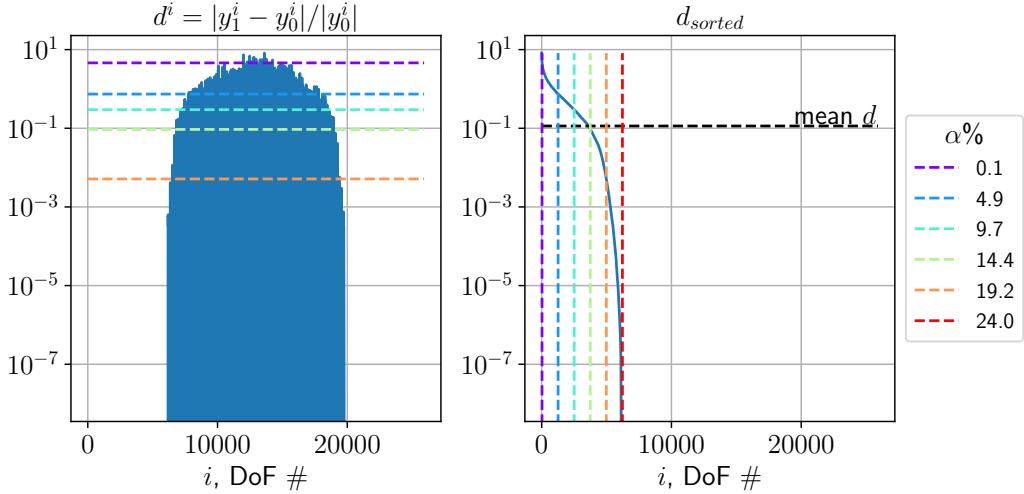


FIG. 3.1. The normalized-difference vector for the Algorithm 1 in natural ordering and sorted in descending order (right).

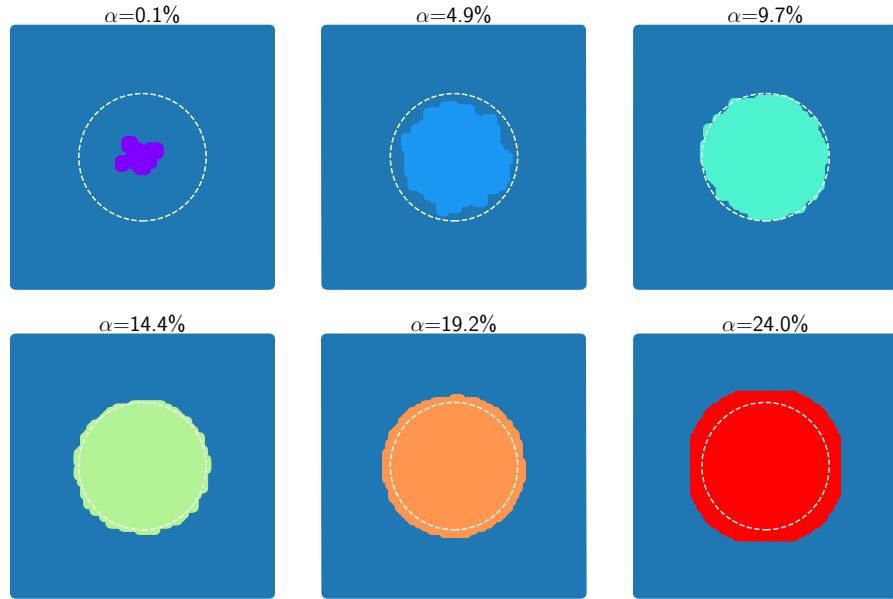


FIG. 3.2. Affected patches at various cutoffs  $\alpha$ . A dashed white circle indicates the area where  $\kappa$  values have been perturbed.

FGMRES( $A_1, MG(A_1, \hat{R}_0^{-1}(\alpha))$ ), where  $\alpha$  represents the selected patch-update cutoff parameter.

From Figure 3.3, we note that for a cutoff value of  $\alpha = 12\%$ , the convergence achieved is unsatisfactory. This is attributed to the partial inconsistency between the updated relaxation operator and the fine-grid operator  $A_1$ . As we gradually increase the cutoff parameter

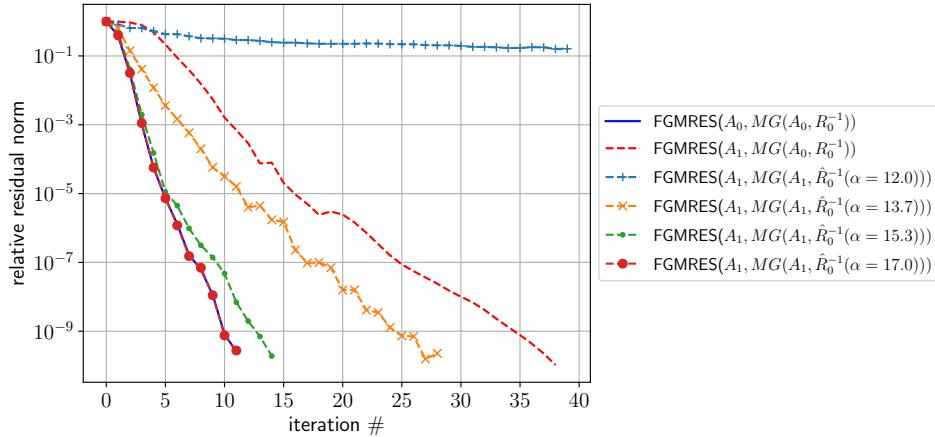


FIG. 3.3. Convergence histories for 2D diffusion problems discretized with  $Q_2$  elements on  $80 \times 80$  quadrilateral mesh. The solvers are FGMRES preconditioned with 2-level smoothed-aggregation AMG, incorporating cell-centric Vanka relaxation on the fine level.

$\alpha$ , the solver's convergence performance improves, with optimal convergence restored at  $\alpha = 17\%$ . This is the same convergence rate we would expect from  $R_1^{-1}$ .

In conclusion, by selectively updating the Vanka relaxation operator associated with the  $A_0$  operator, we can successfully recover desired convergence behavior at a significantly reduced computational cost, compared to setting up a new  $R_1^{-1}$  operator. Future work will focus on identifying an optimal cutoff threshold, which is anticipated to vary depending on the problem and application specifics.

**REMARK 3.1** (Relaxation Damping Weights). *Optimal convergence for the presented problem can also be recovered by setting the relaxation weight  $\omega_D = \text{diag}(A_0)/\text{diag}(A_1)$ , where  $\text{diag}$  denotes extraction of diagonal elements from the input operator. The resulting Vanka relaxation method  $\omega_D R_0^{-1}$  is not numerically equivalent to  $R_1^{-1}$  operator but results in similar convergence rates for relatively smooth changes in  $\kappa(x, y)$  values.*

**REMARK 3.2** (Choice of Krylov Solver). *The choice of the conjugate gradient (CG) method is also appropriate for the outer iteration of the solver since we are solving strictly symmetric positive definite systems. Instead, we use FGMRES for the outer iteration solver because the preconditioner is indefinite at lower  $\alpha$  values due to the inconsistency between the  $A_1$  operator and the un-updated Vanka patches.*

**4. Discussion.** This work uses the variable coefficient diffusion problems as a platform to investigate the intricacies and complexities of patch relaxation reuse techniques. While simpler relaxation methods suffice for these problems, the true value of this study lies in understanding the functionality of the proposed patch-replacement algorithm.

The proposed algorithm relies on approximating an upper-bound smoothing property of the reused relaxation operator to identify the patches that need to be updated. Through convergence analysis and the visualization of the replaced patches, the study has provided insights into how an approximate bound on the relaxation property can be used to guide solver updates. We have shown that for problems that exhibit localized changes in the operator, we can identify the location of the most significant changes and selectively update the relaxation operator, recovering the desired convergence behavior at a fraction of the cost required for a full setup.

We are planning to extend these techniques to more complex scenarios. These include

but are not limited to, time-dependent PDEs and continuous design problems such as topology optimization, where the system operator undergoes gradual changes from solve to solve. We encourage the reader to contact the main author if they have interesting use cases for the described patch replacement algorithm.

**5. Supplement: Alternative Measures.** This section presents alternative measures that, in contrast to (2.5), function directly on patches rather than system unknowns. Theoretically, these alternatives should provide more efficient localization of sub-optimal patches.

**5.1. Patch-Space Measure.** We offer an alternative version of the measure found in (2.5), moving the emphasis from the DoF space to patch space. Consider the relative error:

$$\begin{aligned} y_1 - y_0 &= \left[ (I - \hat{R}_0^{-1} A_1) - (I - R_0^{-1} A_0) \right] y_r \\ &= \left[ -\hat{R}_0^{-1} A_1 + R_0^{-1} A_0 \right] y_r \\ &= \left[ I - (\hat{R}_0^{-1} A_1)(A_0^{-1} R_0) \right] (R_0^{-1} A_0 y_r) \\ \|y_1 - y_0\|_\alpha &= \left\| \left[ I - (\hat{R}_0^{-1} A_1)(A_0^{-1} R_0) \right] (R_0^{-1} A_0 y_r) \right\|_\alpha \\ &\leq \left\| I - (\hat{R}_0^{-1} A_1)(A_0^{-1} R_0) \right\|_\alpha \left\| (R_0^{-1} A_0 y_r) \right\|_\alpha. \end{aligned} \quad (5.1)$$

Our focus will not be on the error in the solution vector but rather on a measure specific to each patch. We aim to remove the gather operator, denoted as  $V^T$ , from the last norm term:

$$\begin{aligned} y_0 &= R_0^{-1} A_0 y_r \\ &= \left( V^T W (K_0^G)^{-1} V \right) A_0 y_r \\ &= V^T y_0^G, \end{aligned} \quad (5.2)$$

Here,  $y_0^G$  symbolizes the correction to the solution within the patch-space, indicated by superscript  $G$ . The term  $(K_0^G)^{-1}$  represents a block-diagonal operator made up of inverses of individual patches,  $K_0^i = V^{(i)} A_0 V^{(i)T}$ , extracted from the  $A_0$  system. Using the SVD decomposition of the binary scatter operator  $V^T = U \Sigma Z^T$ , we can bound (5.2):

$$\begin{aligned} \left\| (U \Sigma Z^T) y_0^G \right\|_\alpha &= \left\| \Sigma Z^T y_0^G \right\|_\alpha \\ &\leq \left\| \sigma_{max} Z^T y_0^G \right\|_\alpha \\ &= \sigma_{max} \left\| Z^T y_0^G \right\|_\alpha \\ &= \sigma_{max} \left\| y_0^G \right\|_\alpha, \end{aligned} \quad (5.3)$$

where  $\sigma_{max} = \|\Sigma\|_\infty$ . The  $U$  and  $Z^T$  operators are omitted because inner-product-induced norms remain invariant when multiplied by orthonormal operators. It is worth noting that the singular values of  $V^T$  lie between [1, 2] (for 2D patches with 1 DoF layer overlap).

Using rationale similar to (5.3) for the left-hand side of (5.1), we derive a lower-bound:

$$\|y_1 - y_0\|_\alpha \geq \sigma_{min} \|y_1^G - y_0^G\|_\alpha \quad (5.4)$$

Combining the primary outcome from (5.1) with the norm simplifications in Equations (5.3) and (5.4), we derive a measure based on solution correction vectors in patch-

space:

$$\frac{\|y_1^G - y_0^G\|_\alpha}{\|y_0^G\|_\alpha} \leq \frac{\sigma_{max}}{\sigma_{min}} \left\| I - \left( \hat{R}_0^{-1} A_1 \right) \left( R_0^{-1} A_0^{-1} \right) \right\|_\alpha \quad (5.5)$$

This result provides a framework for assessing the relative changes in the approximation property of individual patches.

**5.2. Variance-minimizing Measure.** We also aim to explore an alternative measure that calculates:

$$\left\| \left( I - \left( V^{(i)} A_1 (V^{(i)})^T \right) \left( V^{(i)} A_0 (V^{(i)})^T \right)^{-1} \right) \tilde{v}_r^{(i)} \right\|_\alpha,$$

where  $V^{(i)} A_1 (V^{(i)})^T$  is the  $i^{th}$  patch matrix for  $A_1$ ,  $V^{(i)} A_0 (V^{(i)})^T$  is the  $i^{th}$  patch matrix for  $A_0$ , and  $\tilde{v}_r^{(i)}$  is a random vector defined over the  $i^{th}$  patch matrix with norm one. While this formulation requires extraction of  $A_1$ 's patch matrices, it does not require  $A_1$ 's patch matrices to be inverted to use the measure. An analogous measure has been shown to be robust in the spectral clustering of Vanka patches, as detailed in [10].

## REFERENCES

- [1] R. ABU-LABDEH, S. MACLACHLAN, AND P. E. FARRELL, *Monolithic multigrid for implicit runge-kutta discretizations of incompressible fluid flow*, Journal of Computational Physics, 478 (2023), p. 111961.
- [2] 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.
- [3] J. H. ADLER, T. R. BENSON, E. C. CYR, S. P. MACLACHLAN, AND R. S. TUMINARO, *Monolithic multigrid methods for two-dimensional resistive magnetohydrodynamics*, SIAM Journal on Scientific Computing, 38 (2016), pp. B1–B24.
- [4] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in  $h$  (div) and  $h$  (curl)*, Numerische Mathematik, 85 (2000), pp. 197–217.
- [5] P. D. BRUBECK AND P. E. FARRELL, *A scalable and robust vertex-star relaxation for high-order fem*, arXiv preprint arXiv:2107.14758, (2021).
- [6] T. E. BRUNS AND D. A. TORTORELLI, *Topology optimization of non-linear elastic structures and compliant mechanisms*, Computer methods in applied mechanics and engineering, 190 (2001), pp. 3443–3459.
- [7] 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.
- [8] P. E. FARRELL, M. G. KNEPLEY, L. MITCHELL, AND F. WECHSUNG, *Pcpatch: software for the topological construction of multigrid relaxation methods*, ACM Transactions on Mathematical Software (TOMS), 47 (2021), pp. 1–22.
- [9] C. GREIF AND Y. HE, *A closed-form multigrid smoothing factor for an additive vanka-type smoother applied to the poisson equation*, Numerical Linear Algebra with Applications, (2023), p. e2500.
- [10] G. HARPER AND R. TUMINARO, *Compression and reduced representation techniques for patch-based relaxation*, arXiv preprint arXiv:2306.10025, (2023).
- [11] T. Y. LIN, S. E. BAKER, E. B. DUOSS, AND V. A. BECK, *Topology optimization of 3d flow fields for flow batteries*, Journal of The Electrochemical Society, 169 (2022), p. 050540.
- [12] S. P. MACLACHLAN AND C. W. OOSTERLEE, *Local fourier analysis for multigrid with overlapping smoothers applied to systems of pdes*, Numerical Linear Algebra with Applications, 18 (2011), pp. 751–774.
- [13] M. A. SALAZAR DE TROYA, D. A. TORTORELLI, AND V. A. BECK, *Two dimensional topology optimization of heat exchangers with the density and level-set methods*, in WCCM-ECCOMAS Congress, vol. 1300, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2021.
- [14] J. SCHÖBERL AND W. ZULEHNER, *On Schwarz-type smoothers for saddle point problems*, Numerische Mathematik, 95 (2003), pp. 377–399.
- [15] S. P. VANKA, *Block-implicit multigrid solution of navier-stokes equations in primitive variables*, Journal of Computational Physics, 65 (1986), pp. 138–158.
- [16] A. VORONIN, S. MACLACHLAN, L. N. OLSON, AND R. TUMINARO, *Monolithic algebraic multigrid preconditioners for the stokes equations*, arXiv preprint arXiv:2306.06795, (2023).

## II. High Performance & Post-Moore Computing

Articles in this section discuss the implementation of software for high performance computing (HPC) or development of new types of scientific computing. In many cases, performance improvements and portability are demonstrated for many-core heterogenous architectures, such as conventional multicore CPUs, the Intel Many Integrated Core coprocessor (MIC), and graphical processing units (GPUs). Software and algorithms for non-traditional hardware such as SmartNICs or quantum computers offer other opportunities to advance scientific computing in the post-Moore era.

1. *Toribio, Milewicz, Teves, Willenbring and Frye* improve productivity tools for workflow management, reproducibility, and sustainability.
2. *Cooper and Pedretti* optimize sparse matrix-vector multiplication on Vortex, a RISC-V-based GPU.
3. *Dube and Pedretti* evaluate the RISC-V “V” Extension for vectorization in the Ocelot platform using Chipyard and FireSim.
4. *Grayson and Milewicz* investigate the feasibility of system-level retrospective provenance tools for scientific software.
5. *Holman, Kallaughher and Parekh* examine the space-constrained quantum query complexity of boolean functions.
6. *Johansson and Moore* implement rigid body dynamics in the LAMMPS molecular dynamics package to leverage the Kokkos performance portability library.
7. *Khan, Kelley, Liegeois and Rajamanickam* showcase a method to generate fast and portable Kokkos code for sparse linear algebra in Python using MLIR.
8. *Mishler, Ciesko, Olivier and Bosilca* demonstrate the advantages of device-initiated partitioned global address space programming models.
9. *Shawger and Curry* characterize the performance of offloading file operations in node-local parallel file systems.
10. *Surti and Proctor* present a statistically rigorous method to analyze randomized benchmarking data for quantum computers.
11. *Wilber-Gauthier and Seritan* use application-inspired benchmarks to compare different quantum chemistry algorithms for quantum computers.
12. *Yirka, Kallaughher and Parekh* study various problems in quantum complexity theory related to optimization.

S.K. Seritan  
B.W. Reuter

November 28, 2023

## EXPERIENCE REPORT ON ADDITIONAL FEATURES TO ECMF BACK-END PIPELINE AND ENVIRONMENT CONTAINER BUILD PROGRAM

SIMONE ANGELO TORIBIO\*, REED MILEWICZ†, JOSHUA TEVES‡, JAMES WILLENBING§,  
AND JOE FRYE¶

### **Abstract.**

Work was done on two different projects focusing on improving and adding quality of life features for both developers and users. One project worked on was the Engineering Common Model Framework project (ECMF). Work was specifically done on the back-end pipeline's interactive building feature, where users interactively build a workflow component on the command line. The main contributions include the changing of the underlying design to use Rust enums instead of Rust generic trait objects, which provides developers with more centralized/comprehensive build functions. It also provides better compatibility with match structures, as the enums support breaking them down into more specific variants. The other main contribution was the addition of a feature that allows users to restore their interactive build sessions that were interrupted, saving users time when building larger objects interactively. The other project worked on was the environment containers for development project. The goal of this project is to provide containers with development environments so developers can use the containers to build, test, and develop their software without worrying about locally downloading all dependencies. Work was done to incorporate best practices for containers into the build program for the environment containers. This included the use of multi-stage builds and consolidation of RUN commands to reduce image size, as well as better optimizing for the build cache by more optimally ordering the layers in the images. All of these help reduce download and build times, which benefit both developers and users alike.

**1. Introduction.** This paper will describe the work done this summer on two different efforts. One effort this work focused on is the Engineering Common Model Framework (ECMF) project. In this project, the interactivity feature was improved and a new restore in-place feature was added, bettering the user's experience with the interactivity feature. The other effort this work focused on is the work regarding environment containers for development, where current environment container build tools were tested with different programs. Additionally, best practices regarding containers and images were researched and incorporated.

### **2. Background.**

**2.1. ECMF and Workflows.** Much of modern research in all disciplines makes extensive use of software; per a 2014 UK survey, 92% of academics use research software, and 60% say their research would not be practical without it [2]. Often times, an experiment is done using a sequence of different software tools as opposed to a single piece of software. The sequence of software used to carry out some task/experiment will be referred to as a "Workflow". The basic template of a workflow has one software take in some input, process the input accordingly, then produce some output that gets fed into the next piece of software as input. This process repeats itself across the chain of software until the desired output is produced. The primary concern with using workflows instead of a single program is that they are more vulnerable to issues caused by software decay. Using different computer architectures than what was used originally also adversely affects multi-software workflows more than single programs. It only takes one piece of software in the workflow to fail for the whole workflow to fail. Failures may be induced in software over time due to updates

---

\*Sandia National Laboratories, sstrib@sandia.gov

†Sandia National Laboratories, rmilew@sandia.gov

‡Sandia National Laboratories, jbteves@sandia.gov

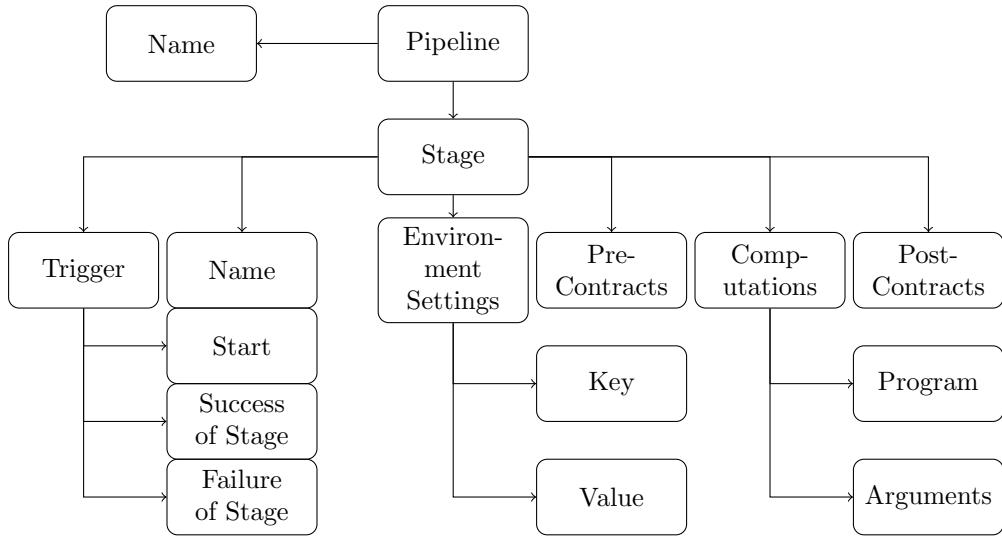
§Sandia National Laboratories, jmwille@sandia.gov

¶Sandia National Laboratories, jfrye@sandia.gov

and changes in components and/or dependencies of the software. For example, the software may use a function that becomes deprecated and eventually removed from a library, causing the program to crash when attempting to run with newer versions of the software. Another example of how dependencies can cause issues over time is if some dependencies don't port over to newer computer systems, even if the main program should be able to. Thus, attempting to run the workflow with dependencies/software that aren't compatible with the newer hardware can also induce faults with results. Faults caused by changes over time may be catastrophic, such as a total crash of the workflow during runtime, or may be subtle, such as slightly altering the results of the test despite using the same input data. Either type of failure is detrimental to experiment reproducibility, because some experiments may be impossible to run in the future or may produce inaccurate results.

Maintaining a workflow can prove to be difficult, especially for domain scientists whose expertise may not lie in computer science or software engineering. According to that same 2014 UK survey, only 21% of academics surveyed had any formal software training [2]. Not only does the workflow need to be revisited and fixed regularly to account for changes in the software, but the workflow must also be maintained for different and emerging computer architectures/systems. It must be maintained for emerging computer architectures/systems because researchers in the future should be able to run the experiment exactly as it was first run. Without maintenance, experiments can get "lost to time", where no hardware in the future can replicate the experiment. Furthermore, as some errors introduced may be subtle, one needs to be able to re-run the experiment regularly to verify that the results have not been altered. There are ways to set up workflows on a per case basis to automatically run like this, but since different workflows can work vastly different from each other, no effective common framework has been created to standardize workflows.

All of the work done on workflow management and sustainability was done in the context of the ECMF project. The goal of ECMF is to provide a way to store workflows in a standard format. ECMF also aims to provide a streamlined way to run workflows for experiment reproduction and software maintenance purposes. The work done focused specifically on the interactivity feature of the back-end pipeline of the framework. All other parts of the framework are out of the scope of this work. The purpose of the back-end pipeline is to create and manage workflow instructions through the use of JSON serializing. The idea is to create a standardized framework that all workflows can be expressed in. This helps alleviate the issue of different workflows having different structures and information flow when trying to create a system to run many different types of workflows. It is also capable of running workflows that are already standardized and serialized as JSON. The back-end pipeline uses multiple nested JSON objects to describe a workflow pipeline:

FIG. 2.1. *Pipeline Components in ECMF*

- EnvironmentSetting
  - key: Environment key to set
  - value: Value to set key to
  - description
- RequiredEnvironmentKey
  - key: Environment key to check if it exists
  - description
- FileVerificationType: Contains a verification to compute when running the workflow
- FileVerificationMethod
  - method: The method used to verify the file
  - verification: The expected output of this verification type
- RequiredFile
  - path: Path to the file to require
  - verification (FileVerificationMethod): The verification method to verify the file with
  - description
- PipelineComputation
  - program: The program to be executed
  - arguments: Arguments to run the program with
  - description
- RequiredFunctionality
  - computation (PipelineComputation): The computation to run for this check
  - expected\_substring (Optional): A substring to expect from the computation
  - description
- Contract: Contains RequiredFile, RequiredFunctionality, or RequiredEnvironmentKey to be satisfied during the execution of a workflow.
- StageTrigger: Specifies when a PipelineStage should be executed
- PipelineStage
  - name
  - on (StageTrigger): Trigger for this stage

- envs (Vector of EnvironmentSettings): A collection of environment settings to be set for this stage
- pre (Vector of Contracts): Contracts to check before computations
- computations (Vector of PipelineComputations): Computations to run for this stage
- post (Vector of Contracts): Contracts to check after computations
- description
- Pipeline: Contains a vector of PipelineStages to run in sequence.

Pipelines of varying structures can be broken down into and described in terms of these nested JSON objects. Since they are serialized as JSON, this also allows simpler transfers of instructions for the workflows, as they can be saved as JSON files and transferred in that form.

**2.2. Containers and containerized development.** A long standing common issue for software users and development teams is dependency management. Ensuring a machine has all properly configured dependencies for a piece of software costs users and development teams a significant amount of time and money. Containers have been identified as a technology that can help alleviate this problem. They are able to accomplish this through grouping all dependencies pre-configured alongside the desired piece of software in one readily downloadable file [5]. When attempting to use the contents of a container, the container software, such as Docker, takes a "blueprint" on how to create a container. This "blueprint" is called an "image". Then, the software creates a container according to the image. Then, the software can be run within that container [5]. Containers have the major benefit of being extremely portable, as containers run similarly on all hardware that can use container software like Docker. Containers are often used to package single applications for use on many different platforms. Examples of popular containerized software include Python, Wordpress, and Rust [4].

Due to a container's ability to run different operating systems and replicate an isolated file-system, containers can also be used to help manage development dependencies. This use of containers is far less explored than the traditional use of packaging a single already built program, but some official images, such as "buildpack-deps" [4] aim to provide packages and dependencies for development rather than a single application. This can bypass much of the hassle of installing and configuring dependencies on a new development system, as the software can be built and tested within a container that already contains all dependencies. This also allows easy deployment, as the container with the dependencies with the built software can be turned into an image and deployed as such.

### 3. Contributions.

**3.1. ECMF Interactive Pipeline Construction.** The main contribution of this work is the improving of the interactive building feature of the software. This feature walks through the creation of any of the JSON workflow objects within the framework using the command line. In particular, the work focused on improving the underlying structure of the code. Tests were added to better assure the functionality of the program. Additionally, the underlying implementation was changed from using trait-objects to using enums instead. In Rust, a type of polymorphism can be achieved through the use of Traits. Traits are similar to interfaces in languages like C#, where any item with that trait must implement the functions of that trait. You can then specify generic type parameters to have certain traits so the trait functions can be used in the function. Since the interactive session's goal is to build all types of ECMF Specifications, all the specifications need to have some unifying "type" that can be specified as a return type. Originally, a trait called "InteractivelyBuildable" was

used as a "return type" of sort. This worked, but is not idiomatic Rust and also produces some issues. One such issue is that the compiler cannot know the size of any struct that inherits "InteractivelyBuildable". This made it hard to recursively nest specifications within each other, which was key for the creation of the new "restore" feature. Furthermore, it is impossible to get more specific than the trait "InteractivelyBuildable". That is, if one wanted to have a function that changed its behavior based on the specification passed in, it would be impossible to identify which specification was passed in, let alone use any functions/methods specific to that specification. In order to resolve these issues, an enum containing variants for all different specifications was used instead of the generic trait type. An enum specifies all variants and what all variants hold when defined, which means the compiler knows the size of enums. Aside from being more idiomatic Rust, enums also allow breaking down into the specific variants. This means that the program can define different behavior for different variants and also use any associated methods for each ECMF specification. Furthermore, this centralized the building function from a collection of disjointed build functions to a single build function that outputs an ECMF specification enum.

The other major contribution of this work is the development of a new "restore" feature of the back-end, allowing restoring in-place of any interrupted interactive session. This ensures users using longer interactive sessions can pick up where they left off if something goes wrong partway through the process. This was accomplished through the use of a build stack that is tracked and saved throughout a build session. The build stack keeps track of all specifications that have been built and are currently being built. For an example, consider the build process for a FileVerificationMethod. One FileVerificationType must be built and held within the FileVerificationMethod. When the build process starts, an info object specifying that a FileVerificationMethod is being built is added to the stack. Once the FileVerificationType starts building, a new info object is added to the top of the stack specifying that a FileVerificationType is being built. Once the FileVerificationType is completed, the stack is popped, and the object is saved within the FileVerificationMethod info object in the build stack. Then, the program finishes building the FileVerificationMethod. The key to the restore command is that the build stack is saved periodically. If the session quits part way through filesystem. If it detects a saved session, it then restores that session's build stack. Then, the program retraces the build stack and starts more or less where the person left off. Figure 3.1 shows a completely successful build process and Figures 3.2 and 3.3 show differently interrupted then restored build process. Note that, when restoring, the program skips over any already completed steps. As an added fail-safe, the program writes the JSON created for all completed ECMF specifications immediately upon completion to disk. This means that, if while building a FileVerificationMethod the FileVerificationType is fully created but the session crashes before full completion, a file containing the built FileVerificationType in JSON will still exist on the disk. These files are deleted after the build process fully completes.

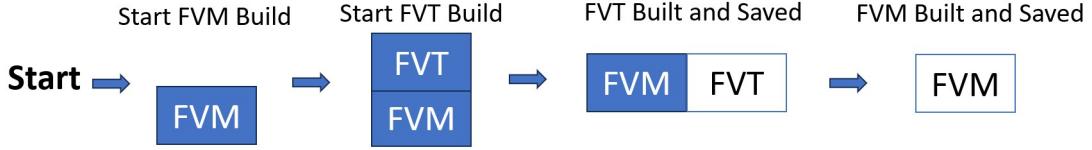


FIG. 3.1. Diagram of a build stack during a completely successful build. *FVT* = *FileVerificationType* and *FVM* = *FileVerificationMethod*. Blue rectangles are incomplete structures and white rectangles (blue outline) are complete structures.

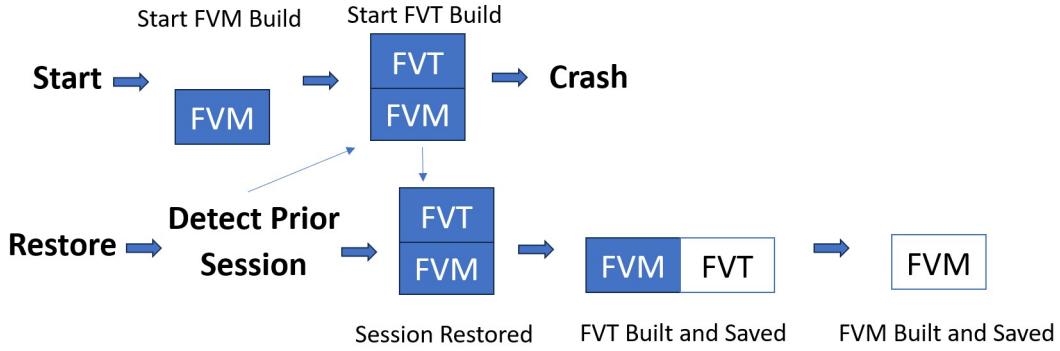


FIG. 3.2. Diagram of a build stack when failing during *FileVerificationType* build step

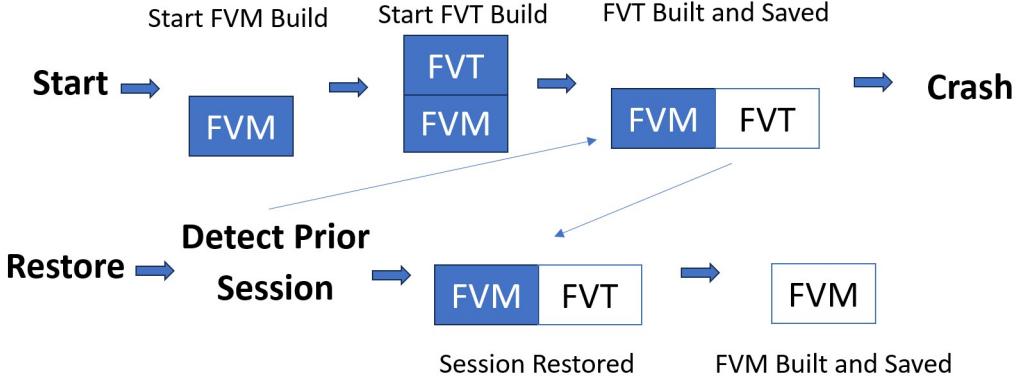


FIG. 3.3. Diagram of a build stack when failing after *FileVerificationType* build step

**3.2. Environment Containers.** All of the work done on the containerized development was done on improving the creation of environment containers. The idea behind environment containers is to provide a container with all dependencies necessary for developing and building a piece of software. One of the centerpieces for the creation of these environment containers is a creation program using Spack. Spack is a package manager that can download and manage package environments. Similar to how must containers and images are structured, the program builds the environment container in layers. First, it

creates a base image that just contains some OS. The program used in this work uses RHEL 8. Next, it creates an image with spack installed in the file-system. Then, it creates a new layer by using spack to download a specified version of gcc. Finally, it uses spack and the specified version of gcc to download and install all packages desired in the environment container. The packages desired can be easily specified by the user by using a spack environment YAML file in the standard spack convention. The resulting image can be used to develop and build the software using either an interactive session, which runs the container in the terminal as its own system, or through adding files and running commands through a new Dockerfile (the file used to create new images). Figure 3.4 shows the layer structure of an environment container and image.

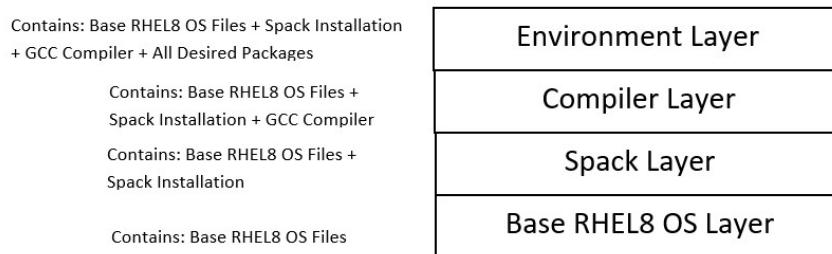


FIG. 3.4. *Diagram of an environment container. Oldest layer is on the bottom.*

The work done on the environment containers project included researching best practices to improve the creation process of environment containers, testing of the existing environment containers, testing of the building program, and documentation on how to build and use custom environment containers. In particular, best practices from Docker and Google were researched and incorporated [1] [3]. One best practice incorporated is the reduction of image size to improve build times and to also reduce download/update times. Different procedures were incorporated to achieve this. One such procedure is the use of multi-stage builds to reduce image size. A multi-stage build is accomplished by running all stages out of one Dockerfile instead of splitting up the build process into different Dockerfiles per stage. Figure 3.5 shows the difference between a multi-stage build and a segmented build. A drawback of the multi-stage build was that it was noticeably harder to find and save the intermediate stages of a build for later use. Another avenue using the label function of Dockerfiles was explored and was found to be effective at finding and saving intermediate stages. However, the process was more complicated than if no multi-stage build was used at all. This technique involved the inspect function from podman, which displays details about a local image in JSON format. The two details that are the most important to this technique are the "parent" and "labels" fields. The "parent" field helps find the image of the layer before this one. This allows the program to recursively go through the whole layer structure. The "labels" field contains all labels of the image. By explicitly labeling each stage with its own value for a common label, in this case the "stage" label, the program can detect changes in build stage by detecting a change in the "stage" label. Then, the program can save the image just before the stage change to save the intermediate image for each stage. A visualization of this is depicted in Figure 3.6.

Another technique incorporated to reduce image size is the combining of multiple sequential "RUN" commands into a single "RUN" command with multiple lines. This helps reduce image size because each "RUN" command creates a new layer (with its own cache).

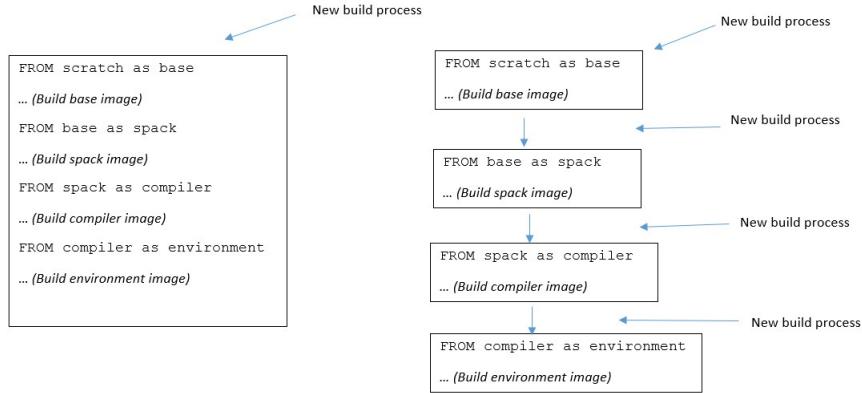


FIG. 3.5. *Multi-stage build (left) vs. segmented build (right). Each border denotes a different Dockerfile*

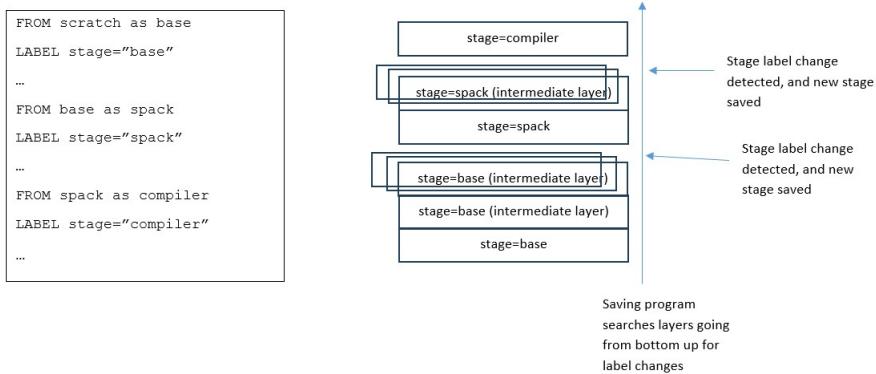


FIG. 3.6. *Visualization of label detection. Stacked boxes represent multiple intermediate layers with the same label.*

The difference between using this technique and not can be seen in Figure 3.7, which shows how using this technique reduces the number of layers.

One other best practice considered and incorporated is better use of the build cache. When building a new image, Docker/Podman searches the build cache for images with identical checksums (based on the contents of the image file) to images to be used in the build. Images are built in layers, which means that unchanged layers can be pulled from the build cache if they already exist instead of building them from scratch. Thus, optimizing for the build cache involves structuring the layering such that the layers at the bottom are the ones that change the least. This is because, if a layer changes, all layers on top of that layer must be rebuilt. A visualization of this can be seen in Figure 3.8. Thus, work was done to examine and restructure the build process of the environment containers such that the layering meets this paradigm. This ensured that the most changed layers were the last layers and the least changed layers were the first layers. The work with multi-stage builds

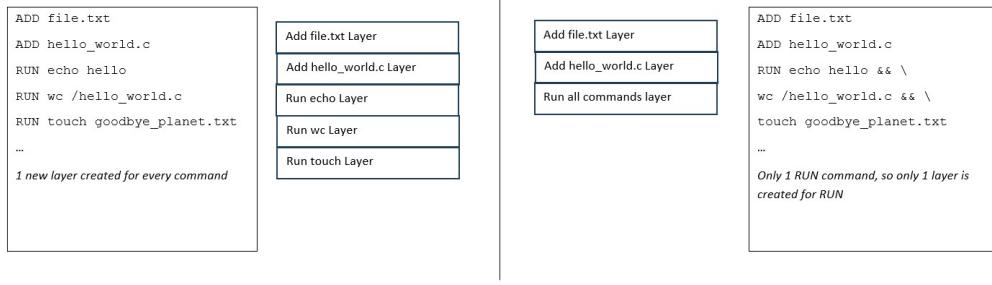


FIG. 3.7. Multiple *RUN* commands (left) vs. single *RUN* command (right).

also helped here, as multi-stage builds can also help optimize for the build cache.

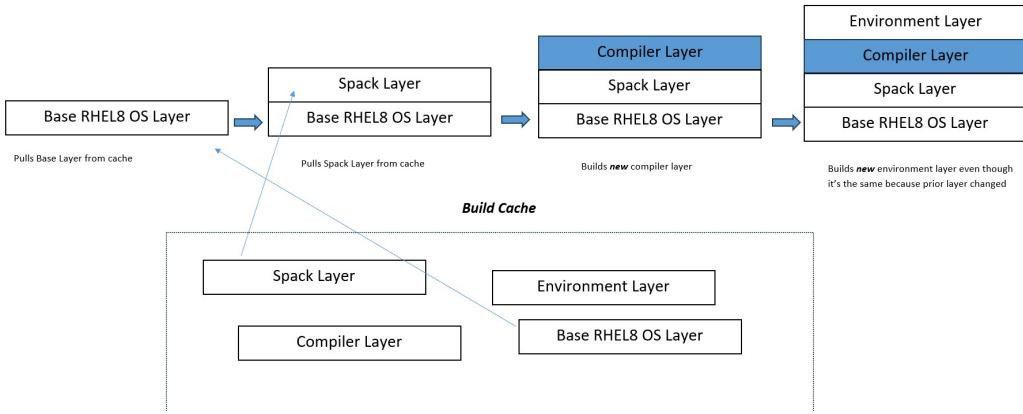


FIG. 3.8. Build cache visualization. Blue layer denotes a changed layer.

#### 4. Impact.

**4.1. ECMF Interactivity.** The efforts done to improve the interactivity function of the ECMF pipeline back-end contributed many quality of life improvements for both developers and users. The work done to change the underlying code from using generic trait objects to using an enum allows developers to more easily work with all ECMF components. This is because enums are much easier to use due to their compatibility and use with match statements. It also ensures developers of future features consider all ECMF specification variants when making functions that work with them, as the compiler will force the programmer to have an action for every variant in a match clause. This can help avoid undefined/unwanted behavior from a programmer not considering one or more ECMF specifications in their implementation. Additionally, the enum allows programmers to use each type specifically, allowing methods specific to a certain variant, such as a method that computes the checksum for a FileVerificationType, to be made and used.

The creation of the restore function will serve as a great boon to users. This is because, in real applications of ECMF, it is expected that users will be creating large pipelines with many nested components. Prior to this work, if something went wrong during the interactive session, all data of objects already built would have been lost. The new restore function

allows the user to start almost right back where they left off, saving a lot of time and trouble. Furthermore, the saving of intermediately built objects as JSON to disk allows users to more easily recover any pipeline specification manually by finding the corresponding JSON file and saving it elsewhere.

**4.2. Environment Containers.** The work done on environment containers helps optimize certain parts of the build process of environment containers. This was done through the implementation of multiple best practices. Optimizing for the build cache drastically reduces build time, as pulling from the cache is much faster than building a layer. The reduction of image sizes through the use of multi-stage builds and consolidating of RUN commands helps reduce download and re-build times, as each layer increases both of these times considerably. The work done also helped simplify the build process for future developers on the project. The use of multi-stage builds allows the program to only use one build process and one file for building one environment container, as opposed to running many separate build process from many different Dockerfiles. This means there are less files and processes to keep track of, allowing future developers to work with the build program in a more streamlined fashion. The label technique for saving intermediate images in multi-stage builds also ensures that each stage of a build has an explicitly saved image, allowing better logging of build processes. This also allows better use of the build cache in future builds, as many of the intermediate images are expected to be shared between builds.

**5. Conclusion.** The work detailed above on the ECMF project has provided quality of life improvements for both the developer and the user. Future developers on the ECMF project will benefit from the switch to enums and users will benefit from the new restore function. The work detailed above on the environment containers project improves the building of containers, as well as helps developers work with the build process of the environment containers project. As a whole, the work detailed in this paper has helped improve both projects worked on for all involved, developer and user alike.

## REFERENCES

- [1] G.-C. A. CENTER, *Best practices for building containers*. <https://cloud.google.com/architecture/best-practices-for-building-containers>, 2 2023.
- [2] S. HETTRICK, *software saved/software\_in\_research\_survey\_2014: Software in research survey*, (2018).
- [3] D. INC., *Docker development best practices*. <https://docs.docker.com/develop/dev-best-practices/>, 10 2023.
- [4] D. INC, *Docker official images*. [https://hub.docker.com/search?image\\_filter=official&q=&type=image](https://hub.docker.com/search?image_filter=official&q=&type=image), 2023.
- [5] ———, *Overview of the get started guide*. <https://docs.docker.com/get-started/#:~:text=A%20container%20is%20an%20isolated,to%20a%20base%20operating%20system.>, 2023.

## OPTIMIZATION OF SPARSE MATRIX-VECTOR MULTIPLICATION ON A RISC-V GPU

LIAM COOPER\* AND KEVIN PEDRETTI†

### **Abstract.**

While the free and open RISC-V instruction set architecture has provided researchers and developers with unprecedented freedom to design and explore hardware architectures for various applications in recent years, the role the RISC-V ecosystem plays in the field of high-performance computing is unclear. This work-in-progress investigates utilizing a RISC-V-based GPU, Vortex, to accelerate sparse matrix-vector multiplication (SpMV), a computation which is important to many scientific applications in high-performance computing. The developments made as part of this work allowed CUDA kernels, including SpMV, to run using Vortex on FPGA for the first time. Ultimately, an analysis for the optimization of SpMV for Vortex and projected future work are presented.

**1. Introduction.** The free and open RISC-V instruction set architecture (ISA) in recent years has enabled an unprecedented level of freedom in exploring hardware architectures with its extensibility and free and open-source software ecosystem [1]. In addition to the widespread adoption and commercial support, these qualities make RISC-V a possible candidate for co-design in the high-performance computing domain in the future.

This work-in-progress aims to provide an analysis of optimizing a typical high-performance computing workload, sparse matrix-vector multiplication (SpMV), on a RISC-V-based GPU. SpMV is an important computation in a variety of scientific applications in many different fields. Vortex is an open-source, RISC-V-based, soft GPU which extends the RISC-V ISA to support a single-instruction, multiple-threads (SIMT) execution model [11].

Vortex includes a full software stack which supports OpenCL and provides a cycle-level simulator. Vortex is able to be configured and synthesized for FPGAs with up to 64 cores and has a peak clock of over 250 MHz. The design with its standard RISC-V five-stage pipeline is highly configurable and extensible, which makes Vortex a fitting research platform for exploring high-performance workloads at an architectural level.

**2. RISC-V SIMT.** In a SIMT execution model, independent hardware threads execute in a group called a warp via a single instruction. Hardware warps typically consist of single-instruction, multiple-data lanes that threads are assigned to and which make use of data parallelism. At runtime, software warps are scheduled to hardware warps according to the device kernel.

**2.1. RISC-V ISA Extension.** To support a SIMT execution model, Vortex extends the RISC-V ISA with new R-type instructions. These instructions are `wspawn`, `tmc`, `split`, `join`, and `bar` [10]. The `wspawn` instruction spawns a number of hardware warps at a certain program counter (PC) value with thread 0 activated. The `tmc` instruction activates or deactivates hardware threads within an executing warp according to the bits set in the thread mask register. The `split` and `join` instructions handle control flow divergence. The `split` instruction pushes the state of the thread mask and branch prediction information for threads into a hardware immediate post-dominator (IPDOM) stack, which stores information related to the immediate post-dominator of a branch. A basic block X immediately post-dominates a distinct basic block Y if and only if X is on the path from Y to the exit and no other basic block is both post-dominated by X and dominates Y [6]. The `join` instruction pops this information off of the IPDOM stack. The `bar` instruction synchronizes warp execution at inserted barrier locations.

---

\*Georgia Institute of Technology, lcooper43@gatech.edu

†Sandia National Laboratories, ktpedre@sandia.gov

**2.2. Microarchitecture.** Vortex's five-stage in-order pipeline includes SIMT hardware components to support the five instructions outlined in section 2.1 [10]. Figure 2.1 depicts Vortex's basic microarchitecture.

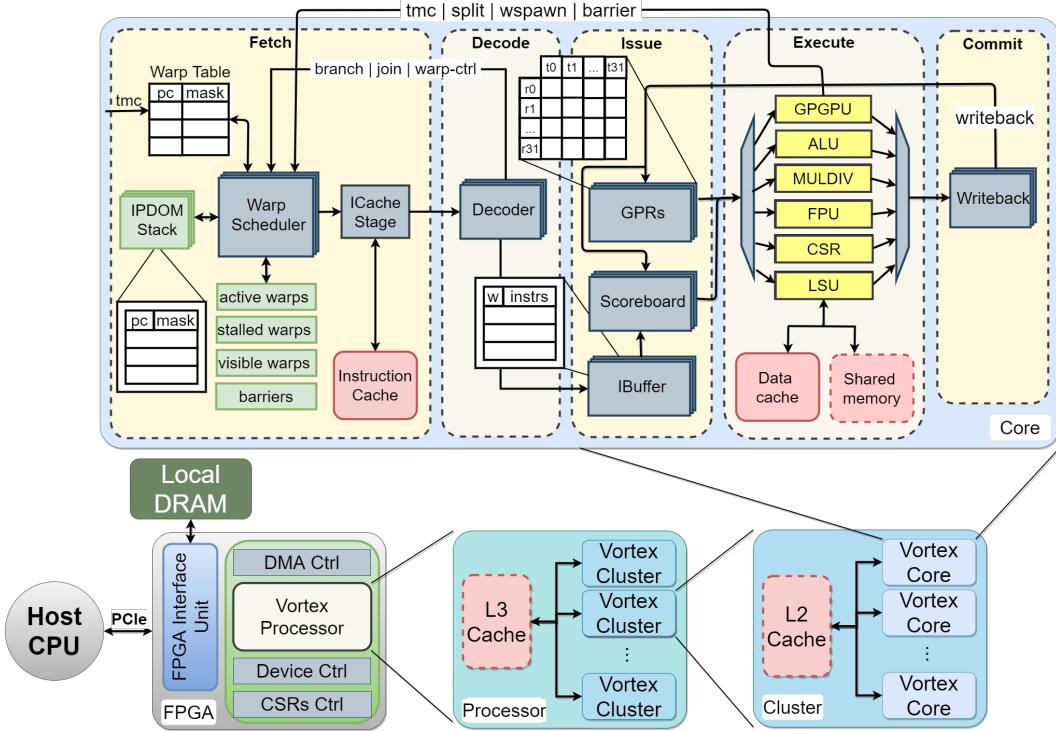


Fig. 2.1: Overview of Vortex's microarchitecture. Adapted from [11].

A warp scheduler was added in the fetch stage which contains a set of warp masks in order to select the next warp to be scheduled and a warp table which keeps track of each warp's information. An IPDOM stack and thread mask register have also been added in hardware. In the case of a `split` instruction, the predicate value for each thread is evaluated. When there is control flow divergence in the program, the following occurs: the thread mask register is pushed onto the IPDOM stack, the active false predicate threads are pushed onto the stack along with the next PC, and thread execution resumes with the thread mask register set in accordance to the active true predicate threads. Then, when a `join` instruction is executed after a `split` instruction, the IPDOM stack is popped and the thread mask register is set using the information popped off the stack. Warp barriers are also implemented in hardware where a table keeps track of the number of warps remaining that have not yet executed the barrier and a mask of barrier-stalled warps. Another similar table is used to manage global barriers.

**3. CUDA for RISC-V.** NVIDIA CUDA is a widely-used computing platform which enables GPUs to be high-performance general-purpose processors [8]. However, CUDA is a proprietary, closed-source platform that is only intended for use with NVIDIA devices. CUDA for Parallelized and Broad-range Processors (CuPBoP) is a solution to execute CUDA on a variety of non-NVIDIA devices and supports many ISAs such as X86, AArch64, and RISC-V [7]. CuPBoP also makes optimizations to support CUDA specifically

on Vortex.

**3.1. CuPBoP.** CuPBoP implements code conversion at an intermediate representation (IR) level and does not rely on source-to-source translators or portable languages. CuPBoP is the first solution for CUDA on non-NVIDIA devices to implement fully compiler-automated single program, multiple data (SPMD) to multiple program, multiple data (MPMD) code conversion at an IR level. CuPBoP has been developed to support Vortex’s RISC-V GPU architecture. The general process for how CuPBoP supports CUDA for Vortex is given below:

1. Using Clang, the source files are separately compiled to LLVM bitcode for both the host and kernel.
2. LLVM IR files are generated separately for the host and kernel: NVVM IR for the kernel code and LLVM IR for the host code.
3. Kernel bitcode is translated by CuPBoP’s kernel translator and the host bitcode is translated by the host translator.
4. The resulting translated bitcode for the kernel and host is then separately cross-compiled to RISC-V for Vortex.

The host translator mentioned in step 3 includes implicit barrier insertion and kernel parameter packing. Implicit barrier insertion is implemented to prevent race conditions associated with multiple asynchronous kernel launches. Kernel parameter packing packs kernel parameters into a single variable to provide a universal interface to the kernel launch function. The kernel translator also mentioned in step 3 implements memory mapping, variable insertion, and SPMD to MPMD translation. As for variable insertion, some CUDA intrinsic functions require use of special registers found on NVIDIA GPUs, so CuPBoP declares these variables in kernel code. During SPMD to MPMD translation, the input kernel functions are wrapped in loops to condense each CUDA block’s workload into a single function.

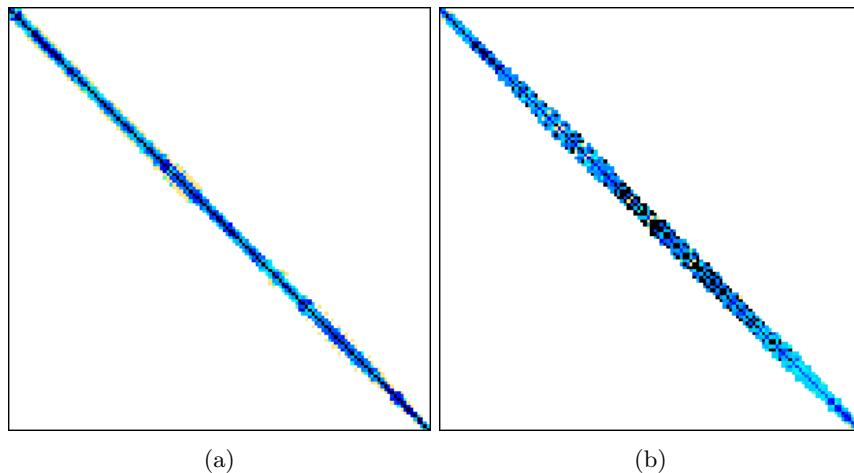


Fig. 4.1: Sparse matrix graphs showing the non-zero patterns for matrices (a) *cf2d* and (b) *ship\_003*.

**4. SpMV for RISC-V SIMT.** SpMV is a computation widely used in many large-scale scientific applications in a number of fields, particularly for problems characterized by linear systems, partial differential equations, and eigensystems [13]. An SpMV computation computes  $y = \mathbf{A}x$  where vectors  $x$  and  $y$  are dense and matrix  $\mathbf{A}$  is sparse. The matrices from The University of Florida Sparse Matrix Collection used for this work, *ship\_003* and *cf2d*, are related to finite element solid mechanics problems and finite difference fluid problems respectively [4]. These two large classes of problems are reasonably representative of the problems frequently solved at Sandia National Laboratories.

Both matrices are real, symmetric, and have relatively low row-length coefficient of variation. *ship\_003* has 121,728 rows and columns and 3,777,036 non-zeros. *cf2d* has 123,440 rows and columns and 3,085,406 non-zeros. The compressed sparse row (CSR) format is used due to its general-purpose nature, no preprocessing overhead, and is not architecture-specific.

One of the simplest implementations of SpMV on a SIMT architecture is to use one thread to multiply one row of the input matrix with the input vector. An example of such an implementation's CUDA kernel can be found in Listing 1 [2]. The CUDA kernel launch parameters for such an implementation include the grid dimension as the number of rows divided by the block dimension. The block dimension can be determined based on the device properties. This naive implementation allows for relatively fewer number of blocks, so each thread is then responsible for more than one row in which the thread would move to the next rows sequentially to accommodate larger matrices.

```

1 int thread_id = blockIdx.x * blockDim.x + threadIdx.x;
2
3 for (int row = thread_id; row < N; row += blockDim.x * gridDim.x)
4 {
5     int row_start = d_rows[row];
6     int row_end = d_rows[row+1];
7
8     T dot_product = 0;
9
10    for (int j = row_start; j < row_end; j++)
11    {
12        int col = d_cols[j];
13        dot_product += d_val[j] * d_x[col];
14    }
15    y[row] = dot_product;
16 }
```

Listing 1: Naive implementation of a SpMV CUDA kernel.

**5. Results.** CuPBoP was ported to run using Vortex's Xilinx Runtime Library (XRT) driver and ran for the first time on FPGA using Vortex synthesized for the Xilinx Alveo U280 FPGA. CUDA on Vortex had previously only ran using Vortex's cycle-level simulator. To accomplish this, CuPBoP's build script for Vortex was modified to pull in the correct dependencies and set up the environment for Vortex's XRT driver to function properly. In part of this effort, Vortex and its dependencies were rebuilt for machines with Red Hat Enterprise Linux-based distributions for ease of use with FPGAs installed in such machines.

The naive implementation described in section 4 was successful in performing SpMV on the *ship\_003* and *cf2d* matrices on Vortex by both executing via Vortex's cycle-level simulation and on Vortex synthesized for the Xilinx Alveo U280 FPGA. However, this implementation is far from the most efficient and optimized solution for either GPUs or Vortex.

**5.1. Optimization.** Because each thread is assigned to do multiply operations on all the elements in a row of the matrix, some threads assigned to shorter rows will be idle while threads with longer rows are still computing. This creates a load imbalance that gets worse when the problem is scaled up with bigger matrices and a larger number of threads. Other performance issues to consider include poor memory performance such as weak memory coalescing and locality.

A suitable solution to the load imbalance problem is to use a merge-based parallel algorithm [9]. This algorithm is able to compute on the commonly-used CSR format matrices, which avoids the overhead of preprocessing for other sparse matrix formats and does not require any architecture-specific encodings. This kind of algorithm decomposes the SpMV as a logical merger of the non-zeros and row-offsets in CSR, then assigning each thread an equitable amount of work. It aims to split up the aggregate CSR such that no thread is overwhelmed by assignment to a large row.

Since this kind of load-balancing optimization has not been run on Vortex yet, the extent to which it would benefit (or degrade) performance is unknown. Without any architecture-specific tuning, this algorithm is able to match or exceed the performance of other CSR SpMV implementations using a conventional NVIDIA GPU. However, when compared to a conventional GPU, Vortex synthesized on FPGA supports far fewer total threads and fewer threads per warp. These factors will influence the performance benefits of load-balancing optimizations.

Exploration of Vortex’s highly configurable hardware, such as number of cores, warps, threads per warp, and cluster configurations would provide insight on the impact on performance compared to a conventional GPU. Creating a specialized sparse matrix format which would better map the data to the underlying architecture of Vortex, including the exploitation of custom hardware instructions for load-balancing search, or row reordering to improve locality and cache performance, could also be explored [5].

**6. Future Work.** This work provided an analysis of optimizing SpMV on a RISC-V-based GPU and leaves room for future work. Future work could include further exploring SpMV and other high-performance computing workloads on Vortex as well as Kokkos on Vortex.

**6.1. Kokkos on Vortex.** Kokkos is an ecosystem for writing high-performance computing applications abstracted from a variety of diverse system architectures [12]. Kokkos allows for a performance-portable programming model implemented through a C++ software library [3]. Among the backends supported by Kokkos are CUDA, OpenMP, C++ Threads, Serial, HIP, HPX, and SYCL.

Vortex is potentially a platform to explore the fitness of RISC-V for high-performance computing co-design due to its highly customizable parallel architecture. Therefore, an effort to port Kokkos to Vortex could be a worthwhile endeavor as part of an analysis of the suitability of the current and future RISC-V ecosystems for high-performance computing applications.

One path to porting Kokkos to Vortex is to use the existing CUDA backend for Kokkos with CuPBoP. This would require changing Kokkos’ build system to use CuPBoP’s build scripts instead of Kokkos’ nvcc wrapper. However, CuPBoP does not yet support all CUDA code, so complicated Kokkos programs may not function correctly or will fail to build. For example, CuPBoP does not yet support CUDA textures and their associated functions, which is used in Kokkos’ `RandomAccess` trait for Views (multi-dimensional arrays). Other examples are some NVIDIA-specific intrinsic functions which NVIDIA does not provide any specifications for. Another potential method to porting Kokkos to Vortex includes using

a RISC-V backend with RISC-V Vector Extension support for use on Vortex’s work-in-progress vector extension.

**6.2. SpMV on Vortex.** As for future work on SpMV for Vortex, many of the optimizations proposed in section 5.1 could be explored. Implementing the sparse CUDA library that includes a merge-based parallel SpMV algorithm as described in section 4 for Vortex could be beneficial to facilitate writing efficient and optimized programs that deal with sparse matrices.

Hardware modifications could also be made to improve Vortex’s performance. A potential improvement would be optimizing Vortex’s memory system for the high-bandwidth memory found on Xilinx’s Alveo FPGAs. If Vortex shows performance benefits in load-balancing optimizations for SpMV, implementing the load-balancing search in hardware could also be experimented with.

## REFERENCES

- [1] 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.
- [2] N. BELL AND M. GARLAND, *Efficient sparse matrix-vector multiplication on CUDA*, NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, Dec. 2008.
- [3] H. CARTER EDWARDS, C. R. TROTT, AND D. SUNDERLAND, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202–3216. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.
- [4] T. A. DAVIS AND Y. HU, *The university of florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011).
- [5] S. FILIPPONE, V. CARDELLINI, D. BARBIERI, AND A. FANFARILLO, *Sparse matrix-vector multiplication on gpgpus*, ACM Trans. Math. Softw., 43 (2017).
- [6] W. W. FUNG, I. SHAM, G. YUAN, AND T. M. AAMODT, *Dynamic warp formation and scheduling for efficient gpu control flow*, in 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007), 2007, pp. 407–420.
- [7] R. HAN, J. CHEN, B. GARG, J. YOUNG, J. SIM, AND H. KIM, *Cupbop: Cuda for parallelized and broad-range processors*, 2022.
- [8] D. KIRK, *Nvidia cuda software and gpu parallel computing architecture*, in Proceedings of the 6th International Symposium on Memory Management, ISMM ’07, Association for Computing Machinery, 2007, p. 103–104.
- [9] D. MERRILL AND M. GARLAND, *Merge-based parallel sparse matrix-vector multiplication*, in SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2016, pp. 678–689.
- [10] B. TINE, V. SAXENA, S. SRIVATSAN, J. R. SIMPSON, F. ALZAMMAR, L. COOPER, AND H. KIM, *Skybox: Open-source graphic rendering on programmable risc-v gpus*, in Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2023, Association for Computing Machinery, 2023, p. 616–630.
- [11] B. TINE, K. P. YALAMARTHY, F. ELSABBAGH, AND K. HYESOON, *Vortex: Extending the risc-v isa for gppgpu and 3d-graphics*, in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO ’21, Association for Computing Machinery, 2021, p. 754–766.
- [12] 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.
- [13] F. VÁZQUEZ, G. ORTEGA, J. FERNÁNDEZ, AND E. GARZÓN, *Improving the performance of the sparse matrix vector product with gpus*, in 2010 10th IEEE International Conference on Computer and Information Technology, 2010, pp. 1146–1151.

## EVALUATING RISC-V SIMD VECTOR “V” EXTENSION SUPPORT WITH CHIPYARD AND FIRESIM

GRiffin Dube\* AND KEVIN PEDRETTI†

**Abstract.** Single Instruction Multiple Data (SIMD) computation via vectorization has a rich history in high-performance computing (HPC), from early Cray vector processors to modern fixed-width vector models like ARM’s scalable vector extensions used in Fugaku’s A64FX processors. This history coupled with recent developments in length-agnostic vector instruction set architectures such as the recently ratified 1.0 version of the RISC-V “V” Extension represents promising advances in the future of vectorization and its role within a high-performance ecosystem. RISC-V’s flexible and customizable instruction set architecture combined with tightly coupled accelerator interfaces provides a prime testbed for re-evaluation of the SIMD vectorization model of computation in an HPC context for increasingly specialized future systems. We evaluate an open-source implementation of the 1.0 “V” specification and its performance on several HPC benchmarks. To the best of our knowledge, we are the first to attempt an evaluation of the 1.0 RISC-V “V” extension in Linux under application-level simulation in FireSim.

**1. Introduction.** Vectorization as a means of accelerating performance in high performance computing (HPC) is a well known method of Single Instruction Multiple Data (SIMD) computation [8] that dates back to the 1970s [15]. In recent years, the popularity of x86, the AVX vector extensions [9], and its fixed-width vector registers has led to an increased rigidity in design that serves to potentially limit the abilities of new custom hardware and software stacks. The release of the RISC-V 1.0 “V” Extension [1] aims to address these challenges by allowing for length-agnostic vector instructions, meaning the ability to choose an appropriate flexible vector length based on the needs of the system. The power of vectorization coupled with the customizable, extensible RISC-V architecture creates an environment full of potential for the HPC ecosystem. Open source RISC-V Core designs such as Rocket [4] and BOOM [22] provide robust bases to build upon when implementing new extensions and features such as the aforementioned 1.0 version of the “V” vector extension.

Since the specification for RISC-V “V” Extension has reached a point of stability, allowing for development of tools, implementations, and infrastructure to advance, the need for robust software stacks and hardware implementations is at a crucial point if continued adoption of the technology is expected to progress within the HPC community. In order to evaluate the progress of the RISC-V vector (RVV) stack, we select and evaluate an open source implementation of RVV, called Ocelot [18], that builds upon the BOOM core to add RVV 1.0 support. Our work investigates the performance of the Ocelot core as well as an example software stack used for developing HPC applications for RVV-enabled processors.

Further, it is crucial to the development of software that efficiently and effectively takes full advantage of the available hardware. For RVV this means that a toolchain supports both development and execution of RISC-V vector applications. However, development of applications that utilize vectorization proves challenging in many situations. Vector code can be difficult to integrate into new and existing applications due to the need for complex intrinsics or inline assembly code [11]. For this reason, compiler autovectorization has also become an important part of the software toolchain and should be considered in a thorough investigation of its maturity with respect to RVV. As such, we also investigate the current state and quality of autovectorization support in the Clang compiler for RISC-V “V” extension as compared to x86 and their AVX vector extensions [9].

The remainder of this paper is organized as follows: Section 2 covers existing implementations of RVV as well as the tools available for RISC-V development and evaluation.

---

\*Northwestern University, gadube@sandia.gov

†Sandia National Laboratories, ktpedre@sandia.gov

Section 3 describes our work and an evaluation of the Ocelot core as well as our evaluation of the current compiler and operating system toolchains that support RVV. Finally, we conclude in Section 4.

**2. Background.** RISC-V’s open source instruction set architecture (ISA) is inherently customizable and configurable. The advancement of the ISA is achieved through extensions which enable additional features. One example of these extensions is the RISC-V “V” extension, which enables vectorization and vector operations and is described in more detail in Section 2.3. As such, it is important to have a robust toolchain for development and simulation of RISC-V based cores and accelerators. To aid in development and design of RISC-V cores, the Chipyard framework for designing and evaluating hardware was created [3]. We utilize this infrastructure, as well as the Firesim FPGA-Accelerated simulator [10], in order to evaluate the Ocelot core.

### 2.1. Open Source RISC-V Development Tools.

**2.1.1. Chipyard.** As mentioned previously, Chipyard is a framework for design and evaluation of RISC-V cores and accelerators [3]. Chipyard allows for quick development and testing via use of the Chisel HDL [5] as well as through highly parameterized core and chip designs. It also eases the difficulty of simulating full cores including peripherals such as memory controllers and networks by supplying highly parameterizable implementations and interfaces to use.

The framework includes multiple open source cores, discussed in more depth in Section 2.2, as well as the infrastructure to support a variety of simulators including Verilator [16], Synopsys VCS [17], and FireSim [10].

**2.1.2. Firesim.** FireSim is an FPGA accelerated, open source simulation platform that allows users to simulate designs at up to hundreds of MHz speeds. The FireSim simulation model allows for instantiation of single core simulations as well as large, multi-node simulations which will play a crucial role in the development of RISC-V cores and accelerators for HPC as their popularity continues to grow.

FireSim also supports simulation that is efficient enough to run a full Linux distribution and allow user interaction at the command line when evaluating a core design, which enables incredibly robust, thorough testing. For ease of use and reproducibility, FireSim has the ability to run using cloud instances on AWS EC2 [10]. This further supports the scalability of evaluation which is crucial for realistic evaluation in HPC.

**2.2. Open Source RISC-V Cores.** There are several open source cores that ship with the Chipyard infrastructure. Among the more popular are the Rocket Chip and BOOM, described below.

**2.2.1. Rocket.** Rocket Chip is an open source implementation of a five-stage, in-order scalar RISC-V processor core [4]. It is highly configurable and contains a core, Rocket Core, which is parameterizable and able to be instantiated along with L1 cache into *tiles* that can be replicated to produce larger system on chip designs.

**2.2.2. BOOM.** BOOM, or the Berkeley Out-of-Order Machine, is a ten-stage, out-of-order processor that serves as an alternative to Rocket, but is a similarly parameterizable open source core design [22].

**2.2.3. Ocelot.** While both Rocket and BOOM cores implement the RV64GC ISA, Ocelot is an extension of the BOOM Core (§2.2.2) which implements the RISC-V “V” Extension’s 1.0 specification [1] and the RV64GCV ISA. Ocelot is currently open source and developed by Tenstorrent, Inc. [18]. Ocelot is implemented in System Verilog and attached

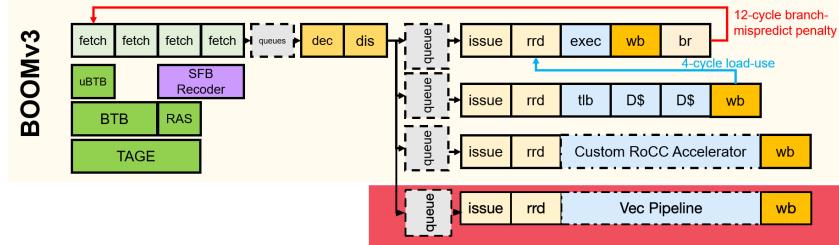


FIG. 2.1. Block Diagram of Ocelot Additions to BOOM (reproduced from Ocelot Documentation [18]), additions to BOOM are shown in red.

to BOOM’s main core via Chisel’s black-box verilog module feature. A block diagram of the extensions provided by Ocelot to the BOOM core is shown in Figures 2.1 and 2.2.

We choose to evaluate the Ocelot core in this work as it serves as a self-contained extension to a well-known, relatively stable BOOM core which can easily be compared with to measure performance of the RVV implementation.

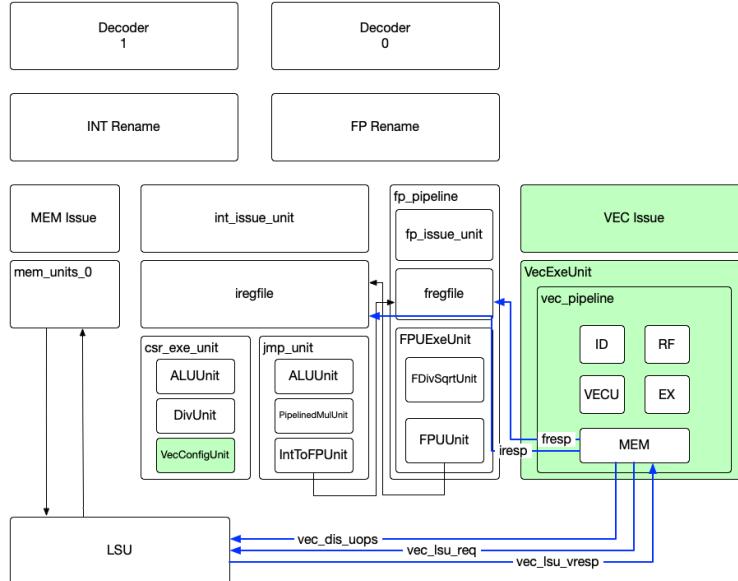


FIG. 2.2. Detailed Block Diagram of Ocelot vector unit and Medium BOOM core (reproduced from Ocelot Documentation [18]), additions to BOOM are shown in green.

**2.3. RISC-V “V” Extension Specification.** The RISC-V “V” extension implementation discussed above is based upon the 1.0 version of the RVV spec. This version of the RVV extension adds 32 vector registers and seven unprivileged CSRs to the base RV64GC ISA [1]. RVV specifies a length-agnostic approach to vectorization, which enables the vector length and element width to be adjusted in order to fit an application’s needs. This includes the ability to use multiple vector registers as operands to a single instruction to simulate the existence of a larger vector containing more elements of a given size.

Prior implementations of earlier versions of the RVV spec [14] as well as non-standard vector accelerators [13] exist, however we narrow our focus on the Ocelot core and their implementation of the 1.0 version of the spec.

**3. Evaluation of RISC-V Vectorization on Ocelot.** Our evaluation of the RVV HW/SW stack consists of three main parts: The Ocelot core, Linux support for RVV, and Clang compiler support for auto-vectorization. We first look at the ability of the Ocelot core to perform vector operations in machine-mode simulation via Verilator, and evaluate the support available for RVV in supporting libraries such as LibGLOSS [20] and Newlib [21]. We then progress to FireSim-based simulation where we boot the Ocelot core under Linux and investigate the performance of Clang’s autovectorizer as well as evaluate the ability to run applications with RVV instructions.

**3.1. Infrastructure for Machine-Mode Simulation.** The Chipyard framework comes with toolchain utilities which support machine mode simulation of RISC-V programs on custom chips designed within the framework such as LibGLOSS and Newlib. These provide wrappers around important system calls which support simulating basic programs and tests to check functionality of a custom design.

In our testing, we simulate Ocelot using Verilator and the provided infrastructure for simulation within Chipyard. To test basic functionality of the vector unit implemented within Ocelot, we developed a simple vector addition kernel which performs the operation shown in Listing 3.1. Vector code is generated from this simple for loop via Clang’s auto-vectorization, which is further discussed in Section 3.3.

```
void vector_addition(double *a,
                     double *b,
                     double *c,
                     int size) {
    for (int i = 0; i < size; i++) {
        c[i] = a[i] + b[i];
    }
}
```

LISTING 3.1  
*Example of a simple vector addition program in C.*

```
...
/* Set mstatus vector bits to active
 * to enable vector unit
 */
li t0, (MSTATUS_FS | MSTATUS_XS | MSTATUS_VS)
csrs mstatus, t0

#ifndef __riscv_vlen
/* Initialize Vector Unit if present */
la t0, 1f
csrw mtvec, t0

/* set vstart and vl registers to known values, this will set 16 element,
 * 8-bit vectors on a machine with vector width of 128bits
 */
li a0, 0x1000
vsetvli t0, a0, e8

/* Initialize all vector registers to zero */
vmv.v.i v0, 0x0
vmv.v.i v1, 0x0
...

```

LISTING 3.2  
*Initialization of the Ocelot vector unit in crt0.S within LibGLOSS*

Our basic experiments found a lack of support in LibGLOSS for RVV. This leads to vector instructions being interpreted as illegal instructions within the simulator, causing

our tests to fail. We add support, remedying the issue by setting the initial state of the vector unit via the `_start` function within `crt0.S`. The vector initialization code, shown in Listing 3.2, first sets the `mstatus` register, a control status register (CSR) that contains part of the hardware execution context current operating state, such that the two bits corresponding with support for RVV are set to their *initial* state. The next step initializes the `vtype` and `vl` registers using a `vsetvl` instruction. This is required by the RVV spec in order to ensure that `vtype`, a register containing the default data type used to interpret vector register contents, and `vl`, a CSR containing the number of elements to be updated from a given vector instruction, can be read and restored. We achieve this in Listing 3.2 by setting these registers to a known initial value. Lastly, we then initialize the 32 vector registers to zero, clearing any unintended results from them. This code mirrors much of the setup for the floating point unit which can also be found in `crt0.S`.

After adding the vector unit initialization code, our example vector code from Listing 3.1 is able to be run within the Ocelot core under Verilator simulation. After successfully running machine-mode simulation, our next steps take us towards full simulation of an example operating system and running user-mode applications on Linux within FireSim’s FPGA-accelerated simulation environment.

**3.2. Operating System Support.** RVV support is in its nascent form within the Linux kernel, with Linux 6.5 being the first version that supports the RISC-V vector extensions by default. Currently Linux 6.5 is not yet released, so we perform our tests on release candidate two of the Linux 6.5 kernel (Linux 6.5-rc2) [19]. This version of Linux supports vector instructions in line with the RVV 1.0 spec that is implemented under Ocelot. We run Fedora with the 6.5-rc2 kernel within FireSim for our experiments.

RVV support in Linux, as of our version, is initialized via an illegal instruction exception which takes place upon first use of a vector instruction. Within the illegal instruction handler, Linux checks whether or not the instruction causing the exception is, in fact, a vector instruction. If a vector instruction caused the exception and the processor supports RVV, the kernel will then allocate the vector context and enable the vector unit for use before continuing on with computation.

This behavior is different from how Linux handles the ARM64 SIMD context, where the vector context is initialized upon the start of a new process. It also differs from x86 where AVX, is initialized along with the floating point unit state, then later cloned into all processes [19].

Although this differs from other popular architectures initialization techniques, it does not seem to pose any directly perceivable challenges to a functional implementation of RVV. Current performance standards seem to be much more limited by the ability of current compiler support for generating RVV as well as the completeness of available implementations of the RVV 1.0 spec, as discussed in Section 3.4. The method used for RVV seems to be a result of the extreme extensibility of RISC-V. Linux leaves the vector unit uninitialized until it is certain it will be used, thus saving resources until they are absolutely necessary.

**3.3. Compiler Autovectorization of “V” Extension.** Section 1 described the importance of autovectorization at the compiler level in supporting the development of RVV within codebases. The Clang compiler, an LLVM-based compiler toolchain for C and C++ code, began supporting auto-vectorization for RISC-V in version 15.0.0 [12]. We use the most recent release of Clang/LLVM, version 16.0.0, which has slightly improved support for RVV for our tests.

We find that the auto-vectorization provided by Clang will successfully vectorize basic loops such as the example code referenced in Listing 3.1 or the WAXPBY kernel within HPCG [7], however it struggles to perform effectively with more complex kernels such as

a sparse matrix-vector multiply (SPMV) kernel. This is a common issue among auto-vectorizers in modern compilation toolchains [6]. Due to the indirect memory accesses and reduction performed in the SPMV kernel, it is difficult for the compiler to effectively determine whether or not it is able to perform auto-vectorization, as shown in Figure 3.1.

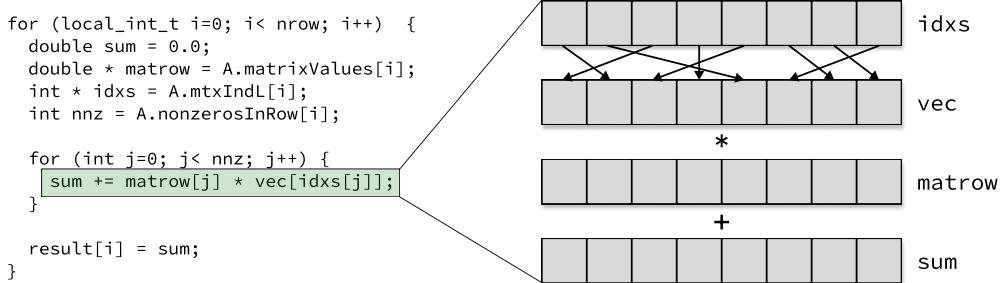


FIG. 3.1. The inner loop of the SPMV kernel, shown in green, is difficult to auto-vectorize for many compilers due to the indirect memory accesses visualized in the left portion of this figure. It is difficult for the compiler to accurately determine whether the indices within the vector array will be unique and therefore difficult to guarantee correctness of a vectorized version of the code.

Interestingly, Clang's x86-64 back-end will generate vector instructions for the difficult-to-vectorize SPMV kernel. The emitted instructions, however, do not efficiently take advantage of the vector unit. Instead, the vector operations performed operate on only the lowest element of the x86 vector registers, the vfmadd132sd instructions in Figure 3.2, along with minimal unrolling of operations. Alternatively, the RISC-V Clang back-end will not emit any vector instructions, opting to perform scalar operations only. These differences are shown in Listing 3.2. Due to limitations and lack of hardware resources to test with, it is unclear whether this difference in emitted instructions results in similar performance.

Additionally, the auto-vectorized code that is generated by Clang's RISC-V back-end mirrors the code generated by the x86 back-end. This seems to indicate that auto-vectorization will perform similarly regardless of whether it takes place on an architecture supporting length agnostic vectors or fixed-length vectors, though further investigation would be required to confirm this [2].

**3.4. Challenges of the Current HW/SW Stack.** Though the OS and compilation toolchains yielded successful results in generating auto-vectorized code and displaying ability to run it successfully, the Ocelot core lacks several features which pose challenges for running larger applications using RVV. One of the major limitations of the current release of Ocelot is its lack of support for exceptions during RVV instructions as described in its documentation [18]. This limits its ability to run larger scale applications or even basic benchmarks such as STREAM and HPCG. The core is able to execute small problem sizes for our basic vector addition example from Listing 3.1, however as problem size increases above 1024 fp64 elements, the processor becomes increasingly unreliable. We have found this to be the result of exceptions that occur non-deterministically during computation of a vector instruction, which causes the Ocelot processor to hang indefinitely, stalling our FireSim simulation.

This presents a serious challenge to simulating larger problems and more complex applications within Ocelot, however these do not seem to be fundamental problems with the core, rather an engineering effort that must be applied in order to introduce the capability to handle these exceptions correctly in the hardware. As such, future work includes an evaluation of alternative implementations of the RVV 1.0 spec in open-source core designs, such

```
.LBB0_14:
    movsx r12, dword ptr [rbx+4*r15]
    vmovsd xmm1, qword ptr [r11+8*r15]
    vmovsd xmm2, qword ptr [r11+8*r15+8]
    vfmaadd132sd xmm1, xmm0, qword ptr [rcx+8*r12]
    movsx r12, dword ptr [rbx+4*r15+4]
    vfmaadd231sd xmm1, xmm2, qword ptr [rcx+8*r12]
    vmovsd xmm0, qword ptr [r11+8*r15+16]
    ...
    .LBB0_7:
        lw      a5, 0(a0)
        fld    ft2, 0(a2)
        slli   a5, a5, 3
        add    a5, a5, a1
        fld    ft3, 0(a5)
        fmadd.d ft1, ft2, ft3, ft1
```

FIG. 3.2. Snippets of generated assembly code for the SPMV kernel on x86 (left) and RISC-V (right). This output is a result of Clang/LLVM.

```
.LBB0_25:
    vmulsd xmm2, xmm1, qword ptr [rdx+8*rdi]
    vfmaadd231sd xmm2, xmm0, qword ptr [rax+8*rdi]
    vmovsd qword ptr [rcx+8*rdi], xmm2
    .LBB0_35:
        vl1re64.v      v9, (a4)
        vl1re64.v      v10, (a5)
        vfmul.vf       v9, v9, fa1
        vfmacc.vv      v9, v8, v10
        vs1r.v v9, (a3)
```

FIG. 3.3. Snippets of generated autovectorized assembly code for the WAXPBY kernel on x86 (left) and RISC-V (right). This output is a result of Clang/LLVM.

as the New Ara core [14], as well as a modified version of Ocelot which includes the ability to handle such exceptions. This work will allow us to perform a more complete evaluation of the RVV 1.0 open-source implementations.

**4. Conclusion.** In this work, we look at the current state of the RVV 1.0 spec as it relates to the open-source hardware and software stack. We use simple, vectorizable test code to demonstrate the ability of the compiler and operating system toolchains to run vector code on an open source RISC-V processor, Ocelot, and uncover limitations which must be improved upon if this and similar cores are to be adopted by the larger HPC community.

## REFERENCES

- [1] *Risc-v “v” vector extension 1.0*. <https://github.com/riscv/riscv-v-spec/releases/tag/v1.0>, 2021.
- [2] N. ADIT AND A. SAMPSON, *Performance left on the table: An evaluation of compiler autovectorization for risc-v*, IEEE Micro, 42 (2022), pp. 41–48.
- [3] 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. ASANOVIC, AND B. NIKOLIC, *Chipyard: Integrated design, simulation, and implementation framework for custom socs*, IEEE Micro, 40 (2020), pp. 10–21.
- [4] K. ASANOVIC, R. AVIZENIS, J. BACHRACH, S. BEAMER, D. BIANCOLIN, C. CELIO, H. COOK, D. DABBELT, J. HAUSER, A. IZRAELEVITZ, S. KARANDIKAR, B. KELLER, D. KIM, J. KOENIG, Y. LEE, E. LOVE, M. MAAS, A. MAGYAR, H. MAO, M. MORETO, A. OU, D. A. PATTERSON, B. RICHARDS, C. SCHMIDT, S. TWIGG, H. VO, AND A. WATERMAN, *The rocket chip generator*, Tech. Rep. UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.
- [5] J. BACHRACH, H. VO, B. RICHARDS, Y. LEE, A. WATERMAN, R. AVIŽENIS, J. WAWRZYNEK, AND K. ASANOVIC, *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.
- [6] Y. CHEN, C. MENDIS, M. CARBIN, AND S. AMARASINGHE, *Vegen: a vectorizer generator for simd and beyond*, in Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2021, pp. 902–914.

- [7] J. DONGARRA, P. LUSZCZEK, AND M. HEROUX, *Hpcg technical specification*, Sandia National Laboratories, Sandia Report SAND2013-8752, (2013).
- [8] M. FLYNN, *Very high-speed computing systems*, Proceedings of the IEEE, 54 (1966), pp. 1901–1909.
- [9] INTEL, *Intel intrinsics guide*. <https://software.intel.com/sites/landingpage/\IntrinsicsGuide>. Online.
- [10] 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. ASANOVIC, *Firesim: Fpga-accelerated cycle-exact scale-out system simulation in the public cloud*, in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), June 2018, pp. 29–42.
- [11] M. KLEMM, A. DURAN, X. TIAN, H. SAITO, D. CABALLERO, AND X. MARTORELL, *Extending openmp\** with vector constructs for modern multicore simd architectures, in OpenMP in a Heterogeneous World, B. M. Chapman, F. Massaioli, M. S. Müller, and M. Rorro, eds., Berlin, Heidelberg, 2012, Springer Berlin Heidelberg, pp. 59–72.
- [12] C. LATTNER AND V. ADVE, *Llvm: A compilation framework for lifelong program analysis & transformation*, in International symposium on code generation and optimization, 2004. CGO 2004., IEEE, 2004, pp. 75–86.
- [13] Y. LEE, C. SCHMIDT, A. OU, A. WATERMAN, AND K. ASANOVIC, *The hwacha vector-fetch architecture manual, version 3.8. 1*, EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-262, (2015).
- [14] M. PEROTTI, M. CAVALCANTE, N. WISTOFF, R. ANDRI, L. CAVIGELLI, AND L. BENINI, *A “new era” for vector computing: An open source highly efficient risc-v v 1.0 vector processor design*, in 2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2022, pp. 43–51.
- [15] R. M. RUSSELL, *The cray-1 computer system*, Commun. ACM, 21 (1978), p. 63–72.
- [16] W. SNYDER, D. GALBI, AND P. WASSON, *Introduction to verilator*, veripool. org, (2009).
- [17] SYNOPSYS, *Vcs functional verification solution*. <https://www.synopsys.com/verification/simulation/vcs.html>. Online.
- [18] TENSTORRENT, *Risc-v ocelot*. <https://github.com/tenstorrent/riscv-ocelot>, 2023.
- [19] L. TORVALDS, *Linux 6.5 release candidate 2*. <https://github.com/torvalds/linux/tree/v6.5-rc2>, 2023.
- [20] UCB BAR, *Libgloss htif*. <https://github.com/ucb-bar/libgloss-htif>, 2022.
- [21] J. C. VINSCHEN, *Newlib*. <http://sourceware.org/newlib>, 2001.
- [22] J. ZHAO, B. KORPAN, A. GONZALEZ, AND K. ASANOVIC, *Sonicboom: The 3rd generation berkeley out-of-order machine*, in Fourth Workshop on Computer Architecture Research with RISC-V, May 2020.

## EVALUATING SYSTEM-LEVEL PROVENANCE TOOLS FOR PRACTICAL USE

SAMUEL GRAYSON\* AND REED MILEWICZ†

**Abstract.** Tracking provenance has many applications in computational science, especially for improving reproducibility, but it is not yet widely used in practice. In this report, we execute a literature rapid review to find system-level provenance tools and methods for use in practice based on the method of collection, source availability, and platform compatibility.

**1. Introduction.** The Oxford English Dictionary defines **provenance** as “a record of the ultimate origin and passage of an item through its previous owners.” In a scientific context, the origin of an artifact is some experimental procedure, so provenance is a description of that; each input used in the procedure has its own provenance, which might be included in the final product, depending on the depth requested. **Computational provenance** refers to the input files (usually software programs, configuration files, and data) used to generate a specific artifact. Provenance can be collected at the application-level, workflow-level, or system-level [38, 18].

- To collect **application-level provenance**, one would modify each application to emit provenance data. Application-level provenance is the most semantically rich but least general, as it only enables collection by that particular modified application [38].
- To collect **workflow-level provenance**, one would modify the workflow engine, and all workflows written for that engine would emit provenance data. Workflow engines are only aware of the dataflow, not higher-level semantics, so workflow-level provenance is not as semantically rich as application-level provenance. However, workflow-level is more general than application-level provenance, as it enables collection in any workflow written for that modified engine [18].
- To collect **system-level provenance**, one uses operating system facilities to report the inputs and outputs that a process makes. System-level provenance is the least semantically aware because it does not even know dataflow, just a history of inputs and outputs, but it is the most general, because it supports any process (including any application or workflow engine) that uses watchable I/O operations [18].

The workflow-level provenance graph (Figure 1.1b) knows what data goes where, but not what the data represents, so it uses arbitrary labels A, B, C, D for data and X, Y for programs.

If X and Y are called as functions from within one process, system-level provenance would see Figure 1.1c. D does not really depend on A, but the system-level graph (Figure 1.1d) would have no way of knowing that from just the input/output log. The system-level graph also does not know that E only depends on the information in A and C which is also present in B and D. Most applications (e.g., reusing cached results) would prefer false positives than false negatives to the question, “does this depend on that?”, so we conservatively assume an input affects any future output. The system-level graph also does not know the any of the transformations (e.g., from A to B). However, if X and Y are called as subprocesses, the system-level graph may be closer to the workflow-level provenance graph.

Computational provenance is useful for a number of applications:

---

\*UIUC and Sandia, grayson5@illinois.edu

†Sandia, rmilewi@sandia.gov

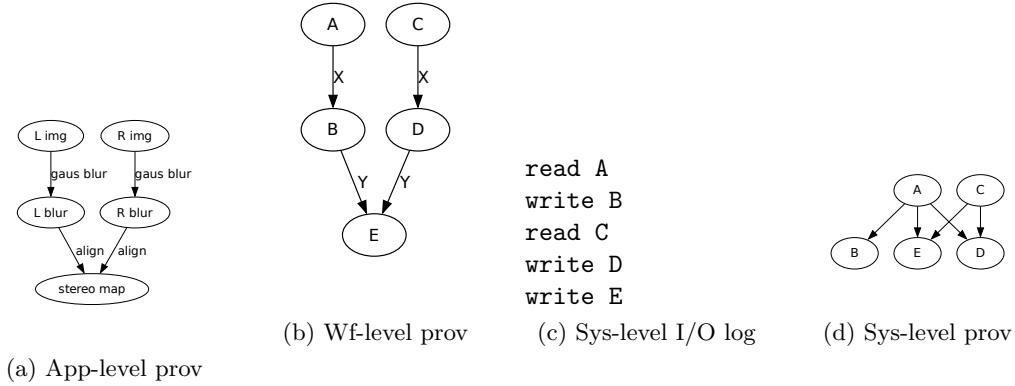


Fig. 1.1: Several provenance graphs collected at different levels.

- While provenance cannot guarantee reproducibility, it can serve as a starting point for one seeking to reproduce a computational artifact.
- With enough details, it can help scientists know which outputs are out-of-date (incremental computing/reusing cached results).
- One could automatically generate the artifact descriptions required by conferences from provenance data.
- It helps scientists debug differences in outputs by viewing differences in their methods.
- Public provenance data can be used to help distribute credit to tool- and data-authors.

There are many possible applications for provenance. Moreover, users collect system-level retrospective provenance (henceforth, **SLRP**) quite easily, reaping these benefits without having to modify the code. They only need to run their experiment in a system that records the inputs and outputs of its child processes. However, there is no SLRP tool used in common practice.

This study aims **to evaluate SLRP tools proposed in prior literature and see if they are appropriate for use in a government labs.**

**1.1. Prior work.** There are many provenance tools and primary studies that we will find in Section 3.1. However, these studies do not use the same benchmarks. We also want to provide additional evaluation of “ease-of-use” and appropriateness for our specific use-case, not a generic benchmark.

Like this study, ProvMark [14] seeks to evaluate SLRP tools. However, ProvMark evaluates the *completeness* of each tool not performance. As performance is not the authors’ goal, ProvMark does not use a realistic application as the benchmark.

Interested readers should consult Freire’s survey [18] to learn the conceptual design space for computational provenance, including retrospective/prospective, capture mechanisms, provenance models, storage formats, and query languages.

**2. Methodology.** We began a rapid review to identify the research state-of-the-art tools for automatic SLRP.

Rapid Reviews are a lighter-weight alternative to systematic literature reviews with a focus on timely feedback for decision-making. Schünemann and Moja show that Rapid

Table 3.1: Primary studies in our search results

Primary study	Tools introduced
Tariq et al. [49]	SPADE on LLVM pass
Sultana et al. [48]	FiPS
Muniswamy-Reddy et al. 2009 [38]	PASSv2
Muniswamy-Reddy et al. 2006 [39]	PASS
Gehani et al. [19]	SPADE
Pohly et al. [41]	Hi-Fi
Pasquier et al. [40]	CamFlow
Rupprecht et al. [42]	Ursprung
Fadolakarim et al. [17]	PANDE
Wang et al. [52]	LPROV

Reviews can yield substantially similar results to a systematic literature review, albeit less detailed [44]. Cartaxo et al. show that Rapid Reviews, although developed in the field of medicine, are useful to inform software engineering design decisions [13, 12].

We conducted a rapid review with the following parameters:

- **Objective:** identify system-level retrospective provenance collection tools.
- **Search terms:** “system-level” AND “provenance”. Note that “retrospective” is less used in prior literature; system-level provenance is usually retrospective.
- **Search engine:** Google Scholar
- **Number of results:** 50 (in order to complete the review in a timely manner). This threshold is the point of diminishing returns as no new tools came up in the results numbered 40 – 50.

### 3. Results.

**3.1. Candidates from Rapid Review.** Several other tools came up, especially workflow-level provenance implementations, but we restrict this report to SLP tools. Of the search results, we list the primary and secondary studies that came up in Tables 3.1 and 3.2. The union of these leads us to large set of SLP tools Table 3.3, for which we found the original publication and characterized the collection method.

**3.2. Suitability review.** Our resulting candidates use a variety of different methods of collecting provenance events. We describe the collection methods below and assess their suitability in Sandia’s context:

- **Tracing:** The Linux kernel also exposes ways for the user to trace a running program (often to implement debuggers) called ‘ptrace(2)’ [5]. Unprivileged users can use the `ptrace(2)` syscall [5] which can modify and trap one of their own processes. This method does not require super-user and is scoped to a subset of the processes and filesystem. Tracing services are **suitable**, provided their performance overhead is small, which the performance study aims to evaluate.
- **Audit service:** The Linux kernel exposes multiple ways to log input and output operations for security auditing purposes (intrusion detection, digital forensics, etc.). Two of which are: Linux Auditing Framework (also called ‘auditd’) and enhanced Berkeley Packet Filter (eBPF) [2]. These methods generally require super-user access, and while that is a security risk, the attack surface can be minimized by encapsulating the audit service in a privileged daemon exposed to unprivileged users (same way Docker works). These methods are **suitable**.

Table 3.2: Secondary studies in our search results

Secondary study	SLRP Tools mentioned	Summary
Li et al. [32]	SPADE, CamFlow, BEEP, Ma et al. [34], MPI, Pro-Tracer, UIScope, Winnower, LDX, MCI, RAIN, RTAG, libdft, OmegaLog	To survey SLP tools for threat-detection
Berrada et al. [9]	SPADE, OPUS	To establish baselines results for anomoly detection algorithms
Braun et al. [10]	PASS, GenePattern, TREC, Lineage FS	To summarize issues in automatic provenance collection
Chen et al. [15]	SPADE+auditd, OPUS, PASS, Hi-Fi, LPM	To compare the expressiveness of OPUS and SPADE
Carat et al. [11]	PASS, PASSv2, SPADE	To survey provenance collection and usage
Zipperle et al. [56]	ETW, Lineage FS, DTrace, PASS, PASSv2, Panorama, SPADE, Hi-Fi, BEEP, LogGC, Sysmon, LPM, Provmon, DataTracker, PROV-Tracer, INSPECTOR, ProTracer, RecProv, MPI, RAIN, CamFlow, LPROV	To survey provenance-based intrusion-detection systems
Lee et al. [30]	PASS, SPADE	To survey secure provenance collection in the cloud

- **Filesystem instrumentation:** Many kernels support software file systems, which pass through I/O calls to an underlying file system after modifying or logging the request. For example Linux offers a Filesystem in User SpacE (FUSE) interface [4] and an older kernel-space Virtual File System (VFS) interface [20]. Filesystem-level provenance collection is **suitable, if they do not involve kernel modification.**
- **Library instrumentation** (also called library interposition and the LD\_PRELOAD trick): Dynamically-linked programs ask the system to find an implementation of a library/application binary interface (ABI) at runtime. Library instrumentation supplies a wrapper library that implements another library’s ABI by logging and passing certain function calls to an underlying implementation. For example, one could write a wrapper around the libc’s `open` function, so when programs open a file, that request gets logged. Some applications (e.g., Go programs) do not use libc at all and some may link against libc statically (especially MUSL libc), both of which would bypass library instrumentation methods. Despite this, the authors of Darshan DXT, an I/O performance tool which uses library interposition, shows that library interposition has a low overhead and most programs in the wild do use libc [53]. Library instrumentation method is **suitable**, at least as a candidate to

the performance test.

- **Static or dynamic binary instrumentation:** Binary instrumentation rewrites a binary executable before it runs (static) or while it is running (dynamic). This method approaches the power of VM extensions without the overhead and the power of kernel-level syscall trapping but at the user-level. However, dynamic instrumentation often requires an Intel CPU tool called Pin [33], which would go against the grain of reproducibility, which wants as many users as possible (especially cloud users) to be able to use the provenance system. Static instrumentation is simpler than dynamic binary instrumentation, as it would only require a specific instruction-set (e.g., x86\\_64). Binary instrumentation is **suitable, if it is not hardware dependent**.
- **Compile-time instrumentation:** A compiler pass can analyze and emit provenance data, especially intra-program control flow, yielding a finer granularity with more precise dataflow. LLVM is the natural choice for instrumentation tooling, because its modular makes writing a new pass simpler. However, compile-time instrumentation would require HPC system administrators to maintain independent builds of the entire software stack. Compile-time instrumentation is **too expensive** to be suitable.
- **Kernel instrumentation:** All input, output, and process launching has to go through the kernel, so the kernel can be modified to emit provenance data. Kernel instrumentation involves modifying the kernel to wrap or hook into syscall handlers. However, modifying the kernel directly or through a kernel module increases the attack surface to an extent that it may be difficult to get approval on classified networks. Kernel-level provenance collection poses **too much security risk**.
- **VM instrumentation:** Virtual machines, such as QEMU, can run programs with dynamic taint tracking often implemented as shadow memory e.g., Panorama[55]. The performance overhead of QEMU with shadow memory is quite great ( 20x for Panorama). VM extensions for provenance collection are **too slow** to be suitable.
- **ISA extensions:** Intel released an extension called Processor Trace [27] which can trace the intra-process control-flow at a fine granularity, but it is **too hardware-specific** to be suitable (see the discussion for dynamic binary analysis).

One should also note whether the target applications use containers or not. Containers in Linux are essentially processes, where the kernel creates a separate namespace for specific subsystems (files, PIDs, users, and network routing), isolated from the rest of the system, as well as performance isolation in the form of cgroups. Kernel-level and file-system level provenance collection methods would still work in that case, but library instrumentation would require modification of the container, which defeats the purpose. However, these approaches trade security risk for performance: kernel-level auditing has to be carefully controlled, so process cannot spy on processes they should not ordinarily be able to see; File-system instrumentation has some unavoidable overhead due to using FUSE, even in its most optimized configuration [51].

Within these tools, we filter based on the following criteria:

1. We rule out tools based on whether their collection method is suitable for use in Sandia's context.
2. We look for collection methods with caveats (i.e., hardware specific, platform specific, and methods involving kernel modification)
3. We check for source code in the original paper, Google search (the first 20 results), GitHub (all results), and BitBucket (all results).

Filtering leaves us with just three candidates: **SPADE+auditd**, **SPADE+FUSE**, and **OPUS**.

Unfortunately, OPUS uses now unsupported Python 2.7 (end of life in 2020), JDK 1.6 (end of active support in 2015), and Neo4J 1.9 (end of life in 2014). Nonetheless, using a Nix environment [16], we got OPUS to build and run<sup>2</sup>. However, it appears OPUS is fatally broken in a way that prevents it from recalling provenance data it stored, and debugging that issue is out-of-scope for this project. Since the ProvMark artifact is available, we study how they use OPUS, but the ProvMark artifact does not actually contain the code which sets up the OPUS tool.

**3.3. Threats to validity. Construct validity:** We ruled out several categories of collection methods, but we believe that our ruling out is justified by suitability needs (minimal Type II errors).

**Ecological validity:** It is possible that even after filtering down the list, these candidates are not actually applicable to work at national labs (acceptable Type I errors).

One concern with any rapid review is completeness; did we actually get enough papers to summarize the state-of-the-art? Our search had diminishing returns and we came across several secondary studies which themselves try to summarize state-of-the-art.

**4. Conclusion.** We identified several candidates for further analysis and possible usage at Sandia. Of these, tracing are the most promising methods that do not require super-user access.

**5. Future work.** We did not attempt a performance and completeness evaluation of the above tools. The obvious next step is to complete that evaluation. Besides benchmarks in prior literature, a future performance evaluation might use extreme-scale parallel applications. xSDK in particular has a set of example applications [7].

Future work might involve user-facing studies, including usability testing, efficacy comparisons, and focus groups. We might give the user a tutorial on the provenance tracing method, have them complete timed challenges, and then interview them on their experience.

Provenance research has more to offer than just tools for collection; there are tools for offline/online filtering of provenance, querying provenance, displaying provenance, etc. In particular, filtering would be important for a government labs use-case. In some common programming patterns, one process ends up doing the bulk of the work; system-level provenance at a coarse granularity can only tell that any of the inputs might affect any of the outputs. This problem is called **dependence explosion** [31]. To mitigate dependence explosion prior work suggests detecting intraprogram dataflow (using compile-time instrumentation or binary instrumentation) or even more complex methods like taint tracking.

## REFERENCES

- [1] *About DTrace*.
- [2] *BPF documentation*.
- [3] *Event tracing - win32 apps*.
- [4] *FUSE*.
- [5] *ptrace*.
- [6] N. BALAKRISHNAN, T. BYTHEWAY, R. SOHAN, AND A. HOPPER, {OPUS}: A lightweight system for observational provenance in user space.

---

<sup>2</sup>See our source code <https://github.com/charmoniumQ/opus#making-opus-work-in-2023>

<sup>2</sup>The ProvMark artifact [14] does not have the infrastructure to set up OPUS; where ProvMark would have that infrastructure, the code only says here:

As OPUS is not published anywhere in the internet, it is not able to generate a vagrant file that completely install opus within... You will need to obtain your own copy of OPUS and extracted within the vagrant VM in order to use OPUS with Prov Mark system.  
OPUS is available on GitHub at the time of writing.

- [7] R. BARTLETT, I. DEMESHKO, T. GAMBLIN, G. HAMMOND, M. A. HEROUX, J. JOHNSON, A. KLINVEX, X. LI, L. C. MCINNES, J. D. MOULTON, D. OSEI-KUFFUOR, J. SARICH, B. SMITH, J. WILLENBRING, AND U. M. YANG, *xSDK foundations: Toward an extreme-scale scientific software development kit*, 4, pp. 69–82. Number: 1.
- [8] A. BATES, D. J. TIAN, K. R. B. BUTLER, AND T. MOYER, *Trustworthy {Whole-System} provenance for the linux kernel*, pp. 319–334.
- [9] G. BERRADA, J. CHENEY, S. BENABDERRAHMANE, W. MAXWELL, H. MOOKHERJEE, A. THERIAULT, AND R. WRIGHT, *A baseline for unsupervised advanced persistent threat detection in system-level provenance*, 108, pp. 401–413.
- [10] U. BRAUN, S. GARFINKEL, D. A. HOLLAND, K.-K. MUNISWAMY-REDDY, AND M. I. SELTZER, *Issues in automatic provenance collection*, in Provenance and Annotation of Data, L. Moreau and I. Foster, eds., vol. 4145, Springer Berlin Heidelberg, pp. 171–183. Series Title: Lecture Notes in Computer Science.
- [11] L. CARATA, S. AKOUSH, N. BALAKRISHNAN, T. BYTHEWAY, R. SOHAN, M. SELTZER, AND A. HOPPER, *A primer on provenance*, 57, pp. 52–60.
- [12] B. CARTAXO, G. PINTO, AND S. SOARES, *Rapid reviews in software engineering*, in Contemporary Empirical Methods in Software Engineering, M. Felderer and G. H. Travassos, eds., Springer International Publishing, pp. 357–384.
- [13] ———, *The role of rapid reviews in supporting decision-making in software engineering practice*, in Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, EASE '18, Association for Computing Machinery, pp. 24–34.
- [14] S. C. CHAN, J. CHENEY, P. BHATOTIA, T. PASQUIER, A. GEHANI, H. IRSYAD, L. CARATA, AND M. SELTZER, *ProvMark: A provenance expressiveness benchmarking system*, in Proceedings of the 20th International Middleware Conference, Middleware '19, Association for Computing Machinery, pp. 268–279.
- [15] S. C. CHAN, A. GEHANI, J. CHENEY, R. SOHAN, AND H. IRSYAD, *Expressiveness benchmarking for {System-Level} provenance*.
- [16] E. DOLSTRA, *The purely functional software deployment model*.
- [17] D. FADOLALKARIM, A. SALLAM, AND E. BERTINO, *PANDDE: Provenance-based ANomaly detection of data exfiltration*, in Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY '16, Association for Computing Machinery, pp. 267–276.
- [18] J. FREIRE, D. KOOP, E. SANTOS, AND C. SILVA, *Provenance for computational tasks: A survey*, 10, pp. 11–21. interest: 97.
- [19] A. GEHANI AND D. TARIQ, *SPADE: Support for provenance auditing in distributed environments*, in Middleware 2012, P. Narasimhan and P. Triantafillou, eds., Lecture Notes in Computer Science, Springer, pp. 101–120.
- [20] GOOCH, *Overview of the linux virtual file system*.
- [21] W. U. HASSAN, M. LEMAY, N. AGUSE, A. BATES, AND T. MOYER, *Towards scalable cluster auditing through grammatical inference over provenance graphs*, in Proceedings 2018 Network and Distributed System Security Symposium, Internet Society.
- [22] W. U. HASSAN, M. A. NOUREDDINE, P. DATTA, AND A. BATES, *OmegaLog: High-fidelity attack investigation via transparent multi-layer log analysis*.
- [23] Y. JI, S. LEE, E. DOWNING, W. WANG, M. FAZZINI, T. KIM, A. ORSO, AND W. LEE, *RAIN: Refinable attack investigation with on-demand inter-process information flow tracking*, in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, pp. 377–390.
- [24] Y. JI, S. LEE, M. FAZZINI, J. ALLEN, E. DOWNING, T. KIM, A. ORSO, AND W. LEE, *Enabling refinable {Cross-Host} attack investigation with efficient data flow tagging and tracking*, pp. 1705–1722.
- [25] Y. JI, S. LEE, AND W. LEE, *RecProv: Towards provenance-aware user space record and replay*, in Provenance and Annotation of Data and Processes, M. Mattoso and B. Glavic, eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 3–15.
- [26] V. P. KEMERLIS, G. PORTOKALIDIS, K. JEE, AND A. D. KEROMYTIS, *libdft: practical dynamic data flow tracking for commodity systems*, in Proceedings of the 8th ACM SIGPLAN/SIGOPS conference on Virtual Execution Environments, VEE '12, Association for Computing Machinery, pp. 121–132.
- [27] A. KLEEN AND B. STRONG, *Intel® processor trace on linux*. Tracing Summit 2015.
- [28] Y. KWON, D. KIM, W. N. SUMNER, K. KIM, B. SALTAFORMAGGIO, X. ZHANG, AND D. XU, *LDX: Causality inference by lightweight dual execution*, in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '16, Association for Computing Machinery, pp. 503–515.
- [29] Y. KWON, F. WANG, W. WANG, K. H. LEE, W.-C. LEE, S. MA, X. ZHANG, D. XU, S. JHA, G. CIOCARLIE, A. GEHANI, AND V. YEGNESWARAN, *MCI : Modeling-based causality inference in audit logging for attack investigation*, in Proceedings 2018 Network and Distributed System Security Symposium, Internet Society.

- [30] B. LEE, A. AWAD, AND M. AWAD, *Towards secure provenance in the cloud: A survey*, in 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 577–582.
- [31] K. H. LEE, X. ZHANG, AND D. XU, *High accuracy attack provenance via binary-based execution partition*, in Proceedings of the 2017 Network and Distributed System Security (NDSS) Symposium.
- [32] Z. LI, Q. A. CHEN, R. YANG, Y. CHEN, AND W. RUAN, *Threat detection and investigation with system-level provenance graphs: A survey*, 106, p. 102282.
- [33] C.-K. LUK, R. COHN, R. MUTH, H. PATIL, A. KLAUSER, G. LOWNEY, S. WALLACE, V. J. REDDI, AND K. HAZELWOOD, *Pin: building customized program analysis tools with dynamic instrumentation*, 40, pp. 190–200.
- [34] S. MA, K. H. LEE, C. H. KIM, J. RHEE, X. ZHANG, AND D. XU, *Accurate, low cost and instrumentation-free security audit logging for windows*, in Proceedings of the 31st Annual Computer Security Applications Conference, ACSAC ’15, Association for Computing Machinery, pp. 401–410.
- [35] S. MA, J. ZHAI, F. WANG, K. H. LEE, X. ZHANG, AND D. XU, *{MPI}: Multiple perspective attack investigation with semantic aware execution partitioning*, pp. 1111–1128.
- [36] S. MA, X. ZHANG, AND D. XU, *ProTracer: Towards practical provenance tracing by alternating between logging and tainting*, in Proceedings 2016 Network and Distributed System Security Symposium, Internet Society.
- [37] MARKRUSS, *Sysmon - sysinternals*.
- [38] K.-K. MUNISWAMY-REDDY, U. BRAUN, D. A. HOLLAND, P. MACKO, D. MACLEAN, D. MARGO, M. SELTZER, AND R. SMOGOR, *Layering in provenance systems*, in Proceedings of the 2009 conference on USENIX Annual technical conference, USENIX’09, USENIX Association, p. 10.
- [39] K.-K. MUNISWAMY-REDDY, D. A. HOLLAND, U. BRAUN, AND M. SELTZER, *Provenance-aware storage systems*, in 2006 USENIX Annual Technical Conference. Accepted: 2015-12-07T19:07:43Z.
- [40] T. PASQUIER, X. HAN, M. GOLDSTEIN, T. MOYER, D. EYERS, M. SELTZER, AND J. BACON, *Practical whole-system provenance capture*, in Proceedings of the 2017 Symposium on Cloud Computing, SoCC ’17, Association for Computing Machinery, pp. 405–418.
- [41] D. J. POHLY, S. McLAUGHLIN, P. McDANIEL, AND K. BUTLER, *Hi-fi: collecting high-fidelity whole-system provenance*, in Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC ’12, Association for Computing Machinery, pp. 259–268.
- [42] L. RUPPRECHT, J. C. DAVIS, C. ARNOLD, Y. GUR, AND D. BHAGWAT, *Improving reproducibility of data science pipelines through transparent provenance capture*, 13, pp. 3354–3368.
- [43] C. SAR AND P. CAO, *Lineage file system*.
- [44] H. J. SCHÜNEMANN AND L. MOJA, *Reviews: Rapid! rapid! rapid! ... and systematic*, 4, p. 4.
- [45] M. STAMATOGLIANNAKIS, P. GROTH, AND H. BOS, *Decoupling provenance capture and analysis from execution*, in Proceedings of the 7th USENIX Conference on Theory and Practice of Provenance, TaPP’15, USENIX Association, p. 3.
- [46] ———, *Looking inside the black-box: Capturing data provenance using dynamic instrumentation*, in Provenance and Annotation of Data and Processes, B. Ludäscher and B. Plale, eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 155–167.
- [47] C. H. SUEN, R. K. KO, Y. S. TAN, P. JAGAPRAMANA, AND B. S. LEE, *S2logger: End-to-end data tracking mechanism for cloud data provenance*, in 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 594–602. ISSN: 2324-9013.
- [48] S. SULTANA AND E. BERTINO, *A file provenance system*, in Proceedings of the third ACM conference on Data and application security and privacy, CODASPY ’13, Association for Computing Machinery, pp. 153–156.
- [49] D. TARIQ, A. MASAIM, AND A. GEHANI, *Towards automated collection of {Application-Level} data provenance*.
- [50] A. VAHDAT AND T. ANDERSON, *Transparent result caching*, in Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC ’98, USENIX Association, p. 3.
- [51] B. K. R. VANGOOR, V. TARASOV, AND E. ZADOK, *To {FUSE} or not to {FUSE}: Performance of {User-Space} file systems*, pp. 59–72.
- [52] F. WANG, Y. KWON, S. MA, X. ZHANG, AND D. XU, *Lprov: Practical library-aware provenance tracing*, in Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC ’18, Association for Computing Machinery, pp. 605–617.
- [53] C. XU, S. SNYDER, V. VENKATESAN, P. CARNS, O. KULKARNI, S. BYNA, R. SISNEROS, AND K. CHADALAVADA, *DXT: Darshan eXtended tracing*.
- [54] R. YANG, S. MA, H. XU, X. ZHANG, AND Y. CHEN, *UISCOPE: Accurate, instrumentation-free, and visible attack investigation for GUI applications*, in Proceedings 2020 Network and Distributed System Security Symposium, Internet Society.
- [55] H. YIN, D. SONG, M. EGELE, C. KRUEGEL, AND E. KIRDA, *Panorama: capturing system-wide information flow for malware detection and analysis*, in Proceedings of the 14th ACM conference on Computer and communications security, CCS ’07, Association for Computing Machinery, pp. 116–

127.

- [56] M. ZIPPERLE, F. GOTTWALT, E. CHANG, AND T. DILLON, *Provenance-based intrusion detection systems: A survey*, 55, pp. 135:1–135:36.

Table 3.3: SLP Tools mentioned in primary and secondary studies in our search results. Rows are sorted from most suitable to least suitable. Instrumentation → ins., method → meth., Filesystem → FS., dynamic → dyn.

Tool	Suitable? 3.2	Sec.	Collection method	Collection tool	Notes
SPADE [19]	Yes		Audit, FS, or compile-time	Multiple <sup>1</sup>	Can use multiple low-level sources
OPUS [6] OmegaLog [22]	Yes No source code		Library instrumentation Static binary ins., audit	libc instrumentaiton Angr, auditd	Finds relationships in logs in mulitple layers
FiPS [48] LogGC	No source code No source code		FS. ins. Audit	VFS Auditd	The contribution is deleting unnecessary parts of the log
RecProv [25] PANDDE [17] Winnower [21]	No source code Kernel modif. Wrong platform		Tracing FS. ins. Audit	rr, ptrace Custom VFS auditd, SELinux, SPADE, Docker Swarm	Specific to Docker Swarm
URSpring	Wrong platform		Audit, file-system ins.	auditd, IBM Spectrum Scale	Specific to IBM Spectrum Scale FS
Event Tracer for Windows [3]	Wrong platform		Audit	NT Kernel	Implemented for Windows
UIScope [54]	Wrong platform		Audit	ETW, accessibility service	Tracks grpahical user interaction
TREC [50]	Wrong platform		Audit	Proc filesystem	Implemented for Solaris
DTrace [1]	Wrong platform		Audit	Respective kernels	Can do event processing in kernel-space
Sysmon [37]	Wrong platform		Audit	NT Kernel	Implemented for Windows
BEEP [31] libdft [26] RAIN [23]	HW-specific HW-specific HW-specific		Dyn., static binary ins. Dyn. binary ins. Library, kernel ins., dyn. binary ins.	Intel Pin, PEBIL Intel Pin libc ins., custom kernel module, Intel Pin	Records syscalls during runtime, replays offline under DIFT
DataTracker [46] INSPECTOR	HW-specific HW-specific		Dynamic binary ins. Library ins., ISA extensions	libthreads ins., Intel PT ISA extensions	
MPI[35]	Unsuitable meth.		Compile-time ins.	LLVM pass	Requires manual input
LDX [28] MCI [29] Ma et al. [34] S2Logger [47]	Unsuitable meth. Unsuitable meth. Unsuitable meth. Unsuitable meth.		Compile-time ins. Compile-time ins. Kernel ins. Kernel ins.	LLVM pass LDX (LLVM pass) ETW Kernel module or Linux Security Module	
ProTracer [36]	Unsuitable meth.		Kernel ins.	Linux tracepoints, custom kernel module	
Hi-Fi [41] Lineage FS [43] PASS/Pasta [39] PASSv2/Lasagna [38]	Unsuitable meth. Unsuitable meth. Unsuitable meth. Unsuitable meth.		Kernel ins. Kernel ins. Kernel ins., filesystem ins. Kernel ins., filesystem, library ins.	Linux Security Module Modified kernel Modified kernel, VFS Modified kernel, instrumented libc, VFS	
RTAG [24]	Unsuitable meth.		Kernel ins.	Modified kernel	Record/reply like RAIN
LPM/ProvMon [8]	Unsuitable meth.		Kernel ins.	Modified kernel, kernel module, NetFilter	
CamFlow [40]	Unsuitable meth.		Kernel ins.	Linux Security Module, Net-Filter	
LPROV [52]	Unsuitable meth.		Kernel, dyn., static binary ins.	Custom kernel module, custom loader, BEEP	
Panorama [55] PROV-Tracer [45]	Unsuitable meth. Unsuitable meth.		VM ins. VM ins.	QEMU QEMU, PANDA	

## THE SPACE-CONSTRAINED QUANTUM QUERY COMPLEXITY OF BOOLEAN FUNCTIONS

BLAKE HOLMAN\*, JOHN KALLAUGHER†, AND OJAS PAREKH ‡

**Abstract.** The quantum query complexity of a function  $f$  is the number of queries to any input  $x$  required to compute  $f(x)$ . In this work we examine  $S$ -constrained space complexity, the number of queries required for any quantum algorithm with at most  $S$  qubits to compute  $f$ . We introduce a (non-convex) semidefinite program whose optimum is the  $S$ -constrained space complexity of the function  $f$ , and examine the dual of a convex relaxation of the program.

**1. Introduction.** Time-space trade-offs bound the time required for a problem as a function of the maximum space usage  $S$ . In the context of quantum computing, these trade-offs can bound the width-depth product of quantum circuits, or bound the amount of time needed to solve a problem with a fixed number of qubits. We begin to address a more restrictive question: For a fixed  $S$ , how many quantum queries are required of any algorithm with at most  $S$  space computing some function  $f$ ? One of the motivations of quantum query complexity is to describe what advantages quantum algorithms have over classical query algorithms. Our question asks the extent to which these advantages depend on the space complexity of a quantum algorithm.

Time-space trade-offs are well-studied in the context of classical computing, very little has been done in the quantum setting. In particular, all current time-space trade-offs have been in the context of problems that have multiple outputs [3, 4, 6] such as finding  $K$  collisions in a random function, and none of these trade-offs say anything nontrivial about the single output case. Of course, many problems of interest only have a single output (decision problems, for example). Aaronson [1] brought attention to this fact, yet no lower bounds have been discovered thus far. We study  $S$ -space-constrained query complexity, the minimum cost query complexity of any algorithm with at most  $S$  qubits computing some Boolean function  $f$ .

**1.1. Techniques.** One of the most powerful tools in quantum query complexity is the quantum adversary method [9]. Given a Boolean function  $f : \{0, 1\}^n \rightarrow n$ , the adversary method defines a semidefinite program (SDP) whose maximum is asymptotically equal to the quantum query complexity of  $f$ . This SDP satisfies strong duality, meaning there's a corresponding dual SDP whose *minimum* attains the same value as the maximization SDP. Interestingly, this dual SDP corresponds exactly to span programs, a linear-algebraic model of computation [9]. Every feasible solution of the dual SDP corresponds to a span program for computing  $f$ . The objective value of a span program corresponds to a cost metric called *witness complexity*. Interestingly, any span program can be compiled into a quantum algorithm for  $f$  with query complexity asymptotically equal to its witness complexity. More surprisingly, the transformation between span programs and algorithm preserves query and space complexity simultaneously [5].

The problem is that, while we have a correspondence between the space complexity of span programs and their feasible solutions in the primal SDP, constraining the space complexity corresponds to constraining the rank of the feasible solutions. This is not a convex constraint. This means that even if we dualized the program, we would likely not achieve strong duality. The program *will* achieve *weak duality*, meaning the feasible solutions

---

\*Purdue University, holman14@purdue.edu

†Sandia National Labs, jmkall@sandia.gov

‡Sandia National Labs, odperek@sandia.gov

lower bound the optimum of the primal program. However, this lower bound may be trivial.

Instead of bounding the rank of our feasible solutions, we bound the trace, as trace most common heuristic for rank in semidefinite programming. If  $\lambda_1, \dots, \lambda_r$  are the eigenvalues of some matrix  $X$ , then  $\text{Trace}(X) = \sum_i \lambda_i$ , while  $\text{rank}(X) = \sum_i \mathbb{I}\{\lambda_i \neq 0\}$ , where  $\mathbb{I}$  is the indicator function. So, if our feasible solutions tend to have eigenvalues close to 1, then trace will be a good approximation. We find the dual of this new trace-constrained SDP, and test it on a family of functions  $f = \text{THRESHOLD}(n, k)$ , for which  $f(x) = 1$  if  $|x| = k$ , and  $f(x) = 0$  if  $|x| = k - 1$ . We chose this family of functions because it is very simple, and the only algorithms which achieve the optimal query complexity use nontrivial space. In particular, the query complexity of  $f$  is  $\Theta(\sqrt{nk})$ , while the space usage of the known algorithms achieving this query complexity is  $\Omega(k \log n)$ . Overall, we find that our program fails to capture the space constraint in the sense that it is not a lower bound. We find feasible solutions on our SDP with respect to  $f$  with an objective value greater than we can prove (by providing an algorithm) for certain assignments of  $S$ .

**2. Background.** In this section we'll overview the necessary results on span programs and convex optimization.

**2.1. Span Programs and Quantum Query Algorithms.** Span programs are a linear-algebraic model of computation for computing Boolean functions  $f : D \rightarrow \{0, 1\}$  for some  $D \subseteq \{0, 1\}^n$ . We'll see that span programs have a strong relationship with the space and query complexity of quantum algorithms computing  $f$ .

**DEFINITION 2.1** (Span Program [2]). *A **span program**  $P(I, V)$  consists of sets of indices*

$$I = \bigsqcup_{j \in [n], b \in \{0, 1\}} I_{j,b}$$

where the entries of  $I_{j,b}$  are

$$I_{j,b} = \left\{ (j, b, i) : i \in \left[ \left[ I_{j,b} \right] \right] \right\}.$$

The (multi)set  $V \subset \mathbb{R}^{f^{-1}(0)}$  consists of input vectors  $|v_{j,b,i}\rangle$  for each  $(j, b, i) \in I$ . The set of available input vectors for  $x \in \{0, 1\}^n$  correspond to indices  $I(x) = \bigsqcup_{i \in [n]} I_{i,x_i}$ . We also say that these vectors are **activated by**  $x$ . We call  $|t\rangle := \sum_{y \in f^{-1}(0)} |y\rangle \in \mathbb{R}^{f^{-1}(0)}$  the **target vector** of  $P$ . The span program  $P$  **computes** the function  $f : D \rightarrow \{0, 1\}$  if for all  $x \in D$ ,

$$f(x) = \begin{cases} 1 & \text{if } |t\rangle \in \text{span } \{|v_i\rangle : i \in I(x)\} \\ 0 & \text{otherwise.} \end{cases}$$

Let  $\{|i\rangle : i \in I\}$  be the computational basis for  $\mathbb{R}^I$ . Let  $A : \mathbb{R}^I \rightarrow \mathbb{R}^{f^{-1}(0)}$  such that

$$A = \sum_{(j,b,i) \in I} |v_{j,b,i}\rangle \langle j, b, i|$$

be the matrix which maps indices to their respective vectors. As a matrix, the columns of  $A$  are the input vectors of  $P$ . The matrix  $\Pi_x : \mathbb{R}^I \rightarrow \mathbb{R}^I$  such that

$$\Pi_x = \sum_{i \in I(x)} |i\rangle \langle i|$$

is the projection onto the basis vectors activated by  $x$ . If  $|t\rangle \in \text{span } I(x)$ , then there's a witness  $|w\rangle$  such that  $A\Pi_x|w\rangle = |t\rangle$ . We measure the cost of such a witness as  $\| |w\rangle \| ^2$ . This quantifies how much of each index is needed to construct the witness  $|w\rangle$ . Similarly, if  $f(y) = 0$ , then  $A\Pi_x|w\rangle \neq |t\rangle$  for all  $|w\rangle$ . So, there's a vector  $|\omega\rangle \in \text{span } \{v_j : j \in I(x)\}^\perp$  in which  $\langle \omega | t \rangle = 1$ . Such a vector is a witness for  $y$ . The cost associated with the witness  $|\omega\rangle$  is again a quantification of the length of the linear combination of indices needed to produce  $\omega$ :  $\| \langle \omega | A \| ^2$ . So, we'd like our span programs to have the property that only linear combinations that are small in magnitude are needed to produce witnesses for inputs to  $f$ . This is formalized in Definition 2.2.

**DEFINITION 2.2** (Witness Complexity, [2]). *Let  $P = (I, V)$  be a span program that computes  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . A vector  $|w\rangle \in \mathbb{R}^I$  is a witness for  $x \in f^{-1}(1)$  if  $A\Pi_x|w\rangle = |t\rangle$ . The witness cost of  $|w\rangle$  is  $c_P(x, |w\rangle) = \| |w\rangle \| ^2$  and the **witness complexity** of  $x$  is*

$$c_P(x) = \min_{|w\rangle: A|w\rangle=|t\rangle} c_P(x, |w\rangle).$$

The **1-witness complexity** of  $P$  under  $f$  is

$$C_1(P, f) = \max_{x \in f^{-1}(1)} c_P(x).$$

Negative witnesses are fixed. The negative witness for  $y \in f^{-1}(0)$  is simply  $|y\rangle$ . The witness cost of  $|y\rangle$  is  $c_P(y) = \| \langle y | A \| ^2$ . The **0-witness complexity** of  $P$  under  $f$  is

$$C_0(P, f) = \max_{y \in f^{-1}(0)} c_P(y),$$

and the **witness complexity** of  $P$  under  $f$  is

$$C(P, f) = \max \{C_0(P, f), C_1(P, f)\}.$$

When  $P$  and  $f$  are clear from context, we'll let  $C_b := C_b(P, f)$ ,  $C := C(P, f)$ , and  $c := c_P$ .

Jeffery [5] showed a strong connection between span program size and the space complexity of unitary quantum algorithms.

**THEOREM 2.3** (From Span Programs to Quantum Algorithms, [5]). *Let  $f : D \rightarrow \{0, 1\}$ , and let  $P$  be a span program that computes  $f$  with witness complexity  $C(f, P) = K$  and  $\text{size}(P) = Z$ , then there exists a bounded error quantum algorithm which computes  $f$  with query complexity  $O(K)$  and space  $O(\log K + \log Z)$ .*

We'll be making more use of the other direction. Given a one-sided error quantum algorithm  $\mathcal{A}$  computing  $f : D \rightarrow \{0, 1\}$  with at most  $2Q+1$  queries and an  $S$ -qubit workspace and  $n$ -dimensional query register, [9] designs a span program  $P_{\mathcal{A}} = (I, V)$  which computes  $f$  with  $C(P, f) = O(Q)$  and size  $2^{S+o(1)}$ .

**THEOREM 2.4** (From Quantum Algorithms to Span Programs [9, 5]). *Let  $\mathcal{A}$  be a one-sided, unitary quantum algorithm computing  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  with query complexity  $Q(n)$  and space complexity  $S(n)$ . Then there exists a span program  $P_{\mathcal{A}}(I, V, |t\rangle)$  computing  $f$  with  $C(P, f) = O(Q(n))$  and  $\text{size}(P) = 2^{S(n)+o(1)}$ . Moreover,  $|I_{j,b}| = |I_{i,a}|$  for all  $i, j \in [n]$  and  $a, b \in \{0, 1\}$ .*

So, if we can say that all span programs computing  $f$  with size at most  $2^S$  has witness complexity at most  $Q$ , then we've shown that any one-sided unitary quantum algorithm computing  $f$  requires  $\Omega(Q)$  queries.

**2.2. Semidefinite Programs and Lagrange Duality.** We'll briefly review the tools used in the next section. Consider the following optimization problem:

$$\begin{array}{ll} \min_{X \in \mathbb{R}^{n \times n}} & f_0(X) \\ \text{Subject to} & X \succeq 0 \\ & f_i(X) \leq 0. \end{array}$$

for  $i \in [m]$ . The lagrangian dual function assigns variables to each constraint. For each constraint  $f_i(x) \leq 0$ , we'll add the variable  $\lambda_i$ , and for the matrix inequality  $X \succeq 0$ , we add a matrix variable  $W$ . The Lagrangian of the above optimization problem is

$$L(X; \lambda, W) = f_0(X) - \langle X, W \rangle + \sum_i \lambda_i f_i(X).$$

We see that if  $W \succeq 0$ ,  $\lambda \geq 0$ , and  $X$  is feasible ( $f_i(X) \leq 0$  for all  $i$ ), then  $L(X; \lambda, W)$  is at least the objective value of  $X$ . So, we should pick  $W \succeq 0$  and  $\lambda \geq 0$  to achieve the best lower bound. The Lagrangian dual function is then

$$g(W, \lambda) = \min_X L(X; W, \lambda).$$

By our above argument, we know that

$$\max_{W \succeq 0, \lambda \geq 0} g(W, \lambda)$$

lower bounds the optimum of the original SDP. This maximization problem is the Lagrangian dual program, and we refer to the original program as the primal. We've just seen that this program satisfies *weak duality*, meaning feasible solutions of the dual lower bounds the optimum of the primal program. Under certain conditions, they satisfy *strong duality*, which means the optimum of the primal and dual are equal. If each  $f_i$  is convex, and there exists a strictly feasible solution ( $f_i(X) < 0$  for all  $i$ ), then strong duality holds. This is useful because it applies to the adversary method SDP.

**3. Semidefinite Programs for Space-Constrained Query Complexity.** The quantum adversary method is a powerful tool for producing query lower bounds for Boolean functions. An adversary matrix  $\Gamma$  for a Boolean function  $f : D \rightarrow \{0, 1\}$  is a symmetric matrix in  $\mathbb{R}^{D \times D}$  with the restriction that  $\Gamma[x, y] = 0$  if  $f(x) = f(y)$ .

**THEOREM 3.1** (Adversary Bound). *Let*

$$Adv(f) := \max_{\substack{\text{Adversary Matrix } \Gamma \\ \|\Delta_i \circ \Gamma\| = 1}} \|\Gamma\|,$$

where  $\Delta_j = \sum_{x, y \in D} |x\rangle \langle y|$  and  $\circ$  denotes element-wise multiplication. The query complexity of  $f$  is  $\Theta(Adv(f))$ .

Interestingly the dual of this program minimizes the witness complexity of span programs computing  $f$ .

**THEOREM 3.2** (Span Program SDP [2]). *For any Boolean function  $f$ ,  $Adv(f)$  is equal*

to the optimum of the following semidefinite program:

$$\begin{aligned} \min & \max_{z \in D} \sum_{i \in [n]} X_i[z, z] \\ \text{subject to} & X_i \succeq 0 \quad \text{for all } i \in [n] \\ & \sum_{i: x_i \neq y_i} X_i[x, y] = 1 \quad \text{for all } x \in f^{-1}(0) \text{ and } y \in f^{-1}(1). \end{aligned}$$

It turns out that the feasible solutions of this SDP correspond exactly with span programs as described in Definition 2.1. Prior results [9, 2, 5] gives a space-query-preserving transformation from quantum algorithm for computing  $f$  to span programs computing  $f$ . We show that the transformation given by [2] between feasible solutions of the SDP from Theorem 3.2 preserves size, implying that to characterize  $S$ -constrained query complexity, it suffices to only consider these feasible solutions.

**THEOREM 3.3** (Quantum algorithms to Feasible Solutions). *For  $S = \omega(\log n)$ , if there exists a one-sided error quantum algorithm  $\mathcal{A}$  which computes  $f : D \rightarrow \{0, 1\}$  with query complexity  $Q(n)$  and space complexity  $S(n)$  then there exists a feasible solution  $X$  of the SDP in Theorem 3.2 with an objective value of  $O(Q(n))$  and  $\text{rank}(X) = 2^{S(n)+o(1)}$ .*

To prove Theorem 3.3, we just need to give a transformation from span programs to feasible solutions which preserves witness complexity and size. It turns out that the known transformation already preserves these metrics, so we'll review the proof that any span program can be converted into a feasible solution with some slight changes for the preservation of span program size [2].

**LEMMA 3.4** (Span Programs to Feasible solutions). *Let  $P = (I, V)$  be a span program for  $f : D \subseteq \{0, 1\}^n \rightarrow \{0, 1\}$  and  $|w_z\rangle$  be witnesses for each  $z \in D$ . Then there exists a feasible solution  $X$  for the SDP in Theorem 3.2 achieving an objective value of  $C(P, f)$  with  $\text{rank}(X) \leq \sum_j \max_z |I_{j,z}|$ .*

*Proof.* Let  $P = (I, V)$  be a span program for  $f$  and  $|w_z\rangle$  be witnesses for each  $z \in D$ . Additionally let  $d_j = \max_{z \in D} |I_{j,z}|$ . Without loss of generality we can assume the witness of  $x \in f^{-1}(1)$  is supported by  $\Pi_x$ , so  $|w_x\rangle = \bigoplus_{j,b} \delta_{x_j b} |\psi_{j,x}\rangle$  for some  $|\psi_{j,x}\rangle \in \mathbb{R}^{d_j}$ . Thus, we have

$$\begin{aligned} c(x, |w_x\rangle) &= \| |w_x\rangle \|^2 \\ &= \sum_j \langle \psi_{j,x} | \psi_{j,x} \rangle \end{aligned}$$

Likewise, for  $y \in f^{-1}(0)$ , the  $y$ th row of  $A_P$ ,  $A_P^\dagger |y\rangle$ , must be zero in columns corresponding to  $v_{j,y_j,i}$  (since  $\langle v_{j,y_j,i} | y \rangle = 0$  by canonicity). Thus,  $A_P^\dagger |y\rangle = \bigoplus_{j,b} \delta_{j,\bar{y}_j} |\psi_{j,y}\rangle$  for some  $|\psi_{j,y}\rangle \in \mathbb{R}^{d_j}$ . Since  $|w_y\rangle = |y\rangle$ , we have

$$\begin{aligned} c(y, |w_y\rangle) &= \sum_{\substack{j \in [n] \\ i \in [d_{i,\bar{y}_j}]}} \langle y | v_{j,\bar{y}_j,i} \rangle \\ &= \sum_{\substack{j \in [n] \\ i \in [d_j]}} \langle y | A | j, \bar{y}_j, i \rangle \\ &= \sum_{j \in [n]} \langle \psi_{j,y} | \psi_{j,y} \rangle. \end{aligned}$$

Now we can construct our feasible solution  $X$ . Let  $X_j = \sum_{z,w \in D} \langle \psi_{j,z} | \psi_{j,w} \rangle |z\rangle \langle w|$  be the Gram matrix of  $\{|\psi_{j,z}\rangle \mid z \in D\}$ . We have for some  $x \in f^{-1}(1)$  and  $y \in f^{-1}(0)$

$$\begin{aligned} \sum_{j: x_j \neq y_j} X_j[x, y] &= \sum_{j: x_j \neq y_j} \langle \psi_{jx} | \psi_{jy} \rangle \\ &= \langle y | A_p | w_x \rangle \\ &= 1 && |w_x\rangle \text{ is a witness for } x; \text{ canonicity.} \end{aligned}$$

Each  $X_j \succeq 0$  since they are Gram matrices for a set of vectors. Finally

$$\begin{aligned} \max_z \sum_j X_j[z, z] &= \max_z \langle \psi_{j,z} | \psi_{j,z} \rangle \\ &= \max_z c(z) \\ &= C(P, f). \end{aligned}$$

We also see that  $\text{rank}(X) \leq \sum_{j \in [n]} d_j$ .  $\square$

The preservation of size when mapping span programs to feasible solutions follows similarly [2].

**LEMMA 3.5** (Feasible Solutions to Span Programs). *Fix a function  $f : D \subseteq \{0, 1\}^n \rightarrow \{0, 1\}$  and let  $X$  be a feasible solution to the SDP from Theorem 3.2. There exists a span program  $P = (I, f)$  computing  $f$  with  $C(P, f) = \max_{z \in D} \sum_{j \in [n]} X_j[z, z]$  and  $|I| \leq 2 \cdot \text{rank}(X)$*

Putting it all together, we see that the optimum of the SDP from Theorem 3.2 when constrained to feasible solutions of rank at most  $2^S$  lower bounds the query complexity of any algorithm using  $S$  space computing  $f$ , proving Theorem 3.3. Let  $P_{\mathcal{A}} = (I, V)$  be the span program from Theorem 2.4 corresponding to  $\mathcal{A}$ . We know that  $C(P, f) = O(Q(n))$  and  $\text{size}(P) = 2^{S(n)+o(1)}$ . By Theorem 3.4 there's a feasible solution  $X$  with objective value at most  $C(P, V)$  and  $\text{rank}(X) \leq \sum_j \max_b |I_{j,b}| \leq \text{size}(P) \leq 2^{S(n)+o(1)}$

Using these arguments, we can say something similar in the other direction. However, there's some inherent slack when going from feasible solutions to quantum algorithms, as it's only known how to transform span programs into **bounded error** quantum algorithms and transform **one-sided error** quantum algorithms into span programs.<sup>1</sup>

**THEOREM 3.6** (Feasible Solutions to Quantum Algorithms [5]). *For  $S = \omega(\log n)$ , if there exists a feasible solution  $X$  of the SDP in Theorem 3.2 with objective value  $v$  and  $\text{rank}(X) = R$ , then there exists a bounded error quantum algorithm which computes  $f$  with query complexity  $O(v)$  and space complexity  $O(\log v + \log R)$ .*

**3.1. Dual SDPs for Space-Constrained Query Lower Bounds.** We'll follow the proofs of [2] and [7] for showing that the SDP in Theorem 3.1 is the dual of the SDP in Theorem 3.2, while adding the relevant space constraints. Ideally, we'd like to constrain the rank of  $X$ , but this isn't convex. Instead, we relax this constraint and bound the trace of

---

<sup>1</sup>The notion of approximate span programs do correspond to bounded error quantum algorithms [5], but for now we only deal with exact span programs.

$X$ , a common heuristic for rank. We can rewrite the span program SDP as

$$\min \quad t \quad (3.1)$$

$$X \succeq 0 \quad (3.2)$$

$$\sum_i \langle z, i | X | z, i \rangle \leq t \quad \text{for all } z \in D \quad (3.3)$$

$$\sum_{i: x_i \neq y_i} \langle x, i | X | y, i \rangle = 1 \quad \text{for all } x \in f^{-1}(1) \text{ and } y \in f^{-1}(0). \quad (3.4)$$

By Theorem 3.3, we want to constrain the rank of  $X$  with respect to our space parameter  $S$ , resulting in the following SDP

$$\min \quad t \quad (3.5)$$

$$\text{subject to} \quad X \succeq 0 \quad (3.6)$$

$$\sum_i \langle z, i | X | z, i \rangle \leq t \quad \text{for all } z \in D \quad (3.7)$$

$$\sum_{i: x_i \neq y_i} \langle x, i | X | y, i \rangle = 1 \quad \text{for all } x \in f^{-1}(1) \text{ and } y \in f^{-1}(0). \quad (3.8)$$

$$\text{rank}(X) \leq 2^S \quad (3.9)$$

Of course,  $\text{rank}(X) \leq 2^S$  is not a convex constraint, so we want to replace it with a convex constraint that will hopefully approximate rank. The most common heuristic for rank constraints is to instead constrain  $\text{Trace}(X)$ . The idea is that if the eigenvalues of  $X$  are close to 1 then the trace will be close to its rank. So, we'll be focusing on the following SDP.

$$\min \quad t \quad (3.10)$$

$$\text{subject to} \quad X \succeq 0 \quad (3.11)$$

$$\sum_i \langle z, i | X | z, i \rangle \leq t \quad \text{for all } z \in D \quad (3.12)$$

$$\sum_{i: x_i \neq y_i} \langle x, i | X | y, i \rangle = 1 \quad \text{for all } x \in f^{-1}(1) \text{ and } y \in f^{-1}(0). \quad (3.13)$$

$$\text{Trace}(X) \leq 2^S \quad (3.14)$$

Let  $F = f^{-1}(1) \times f^{-1}(0) \cup f^{-1}(0) \times f^{-1}(1)$ . The Lagrangian of this SDP is

$$\begin{aligned} L(X, t, \alpha, \beta, Y, \lambda) &= t - \langle X | Y \rangle + \sum_{(x,y) \in F} \alpha_{x,y} \left( 1 - \sum_{i \in [n]} \langle x, i | X | y, i \rangle \right) + \\ &\quad \sum_{z \in D} \beta_z \left( \sum_{i \in [n]} \langle z, i | X | z, i \rangle - t \right) + \lambda \text{Trace}(X) - \lambda 2^S \\ &= t \left( 1 - \sum_z \beta_z \right) + \langle X | B(\beta) + \lambda I - A(\alpha) - Y \rangle + \sum_{x,y} \alpha_{x,y} - \lambda 2^S, \end{aligned}$$

where

$$A(\alpha) = \sum_{x,y} \alpha_{x,y} \sum_{i \in [n]} |x, i\rangle \langle y, i|$$

and

$$B(\beta) = \sum_{z \in D} \beta_z \sum_{i \in [n]} |z, i\rangle \langle z, i|.$$

Let  $g(\alpha, \beta, Y, \lambda) = \min_{X, t} L(X, t, \alpha, \beta, Y, \lambda)$ . We have for  $Y \succeq 0$  and  $\lambda, \beta_z \geq 0$  for all  $z \in D$ ,

$$g(\alpha, \beta, Y, \lambda) = \begin{cases} \sum_{(x,y) \in F} \alpha_{x,y} - \lambda R & \text{if } B(\beta) + \lambda I = A(\alpha) + Y \text{ and } \sum_z \beta_z = 1 \\ -\infty & \text{otherwise,} \end{cases}$$

as if  $B(\beta) \neq A(\alpha) - Y$  then we can choose  $X$  as to let the minimum go to  $-\infty$ . Since  $B(\beta) + \lambda I - A(\alpha) = Y$  and we require  $Y \succeq 0$ , we can simply require  $B(\beta) + \lambda I \succeq A(\alpha)$ . Now let  $g(\alpha, \beta, \lambda) := g(\alpha, \beta, 0, \lambda)$ .

We have

$$\max_{\substack{\alpha, \beta, \lambda \\ B(\beta) + \lambda I \succeq A(\alpha) \\ \lambda \geq 0 \\ \sum_{z \in D} \beta_z = 1 \\ \beta_z \geq 0}} g(\alpha, \beta, \lambda)$$

is equivalent to

$$\max \sum_{(x,y) \in F} \alpha_{x,y} - \lambda R \quad (3.15)$$

$$\text{subject to } B(\beta) + \lambda I \succeq A(\alpha) \quad (3.16)$$

$$\sum_{z \in D} \beta_z = 1 \quad (3.17)$$

$$\lambda, \beta_z \geq 0 \quad \text{for all } z \in D. \quad (3.18)$$

Eq. (3.15) is the SDP we'll be finding feasible solutions for. To recover the original adversary bound, we can simply set  $\lambda = 0$ .

**THEOREM 3.7** (Space Adversary Bound). *For any Boolean function  $f$  and space parameter  $S$  for which the SDP in Eq. (3.10) is strictly feasible, the minimum objective value of Eq. (3.10) is equal to the maximum objective value of Eq. (3.15).*

Now we will examine this SDP in the context of threshold functions.

**4. Threshold Functions.** The problem  $f := \text{THRESHOLD}(n, k)$  is the problem of determining whether a string  $x \in \{0, 1\}^n$  has hamming weight  $k$  or  $k - 1$ , so

$$f^{-1}(1) = \{x \in \{0, 1\}^n \mid |x| = k\}$$

and

$$f^{-1}(0) = \{x \in \{0, 1\}^n \mid |x| = k - 1\}.$$

We'll see that the optimal query complexity for  $\text{THRESHOLD}(n, k)$  is  $\Theta(\sqrt{nk})$ . A matching algorithm is simply to use Grover search to find an index  $i_1$  such that  $x_{i_1} = 1$ , searching for an index  $i_2 \neq i_1$  such that  $x_{i_2} = 1$ , and so on until all positive indices are counted. The query complexity of this is then  $O(k \cdot \sqrt{n/k}) = O(\sqrt{nk})$ , and the space complexity is  $O(k \log n)$ . We conjecture that  $k \log n$  space is required to obtain query complexity  $O(\sqrt{nk})$ . We will first give a standard adversary bound for  $\text{THRESHOLD}(n, k)$  (fixing  $\lambda = 0$  in our new SDP), then we'll give the more general bound by allowing any  $\lambda \geq 0$ .

**4.1. Adversary Bound.** We define  $A(\alpha) = a\Gamma$ , where

$$\Gamma = \sum_{\substack{x \in f^{-1}(1) \\ y \in f^{-1}(0) \\ |x-y|=1}} |x\rangle\langle y|,$$

for some  $a \in \mathbb{R}$  to be chosen later. Likewise, our choice of  $|\beta\rangle$  is  $|\beta\rangle = \sum_{x \in f^{-1}(1)} \frac{1}{2^{\binom{n}{k}}} |x\rangle + \sum_{y \in f^{-1}(0)} \frac{1}{2^{\binom{n}{k-1}}} |y\rangle$ . We want to pick  $a$  as large as possible such that

$$\Delta_j \circ \begin{bmatrix} \frac{1}{2^{\binom{n}{k-1}}} I & -a\Gamma^\dagger \\ -a\Gamma & \frac{1}{2^{\binom{n}{k}}} I \end{bmatrix} \succeq 0.$$

We can simplify this bound via Schur's complement, which implies that the symmetric matrix  $M = \begin{bmatrix} A & B^\dagger \\ B & C \end{bmatrix}$  is positive semidefinite if  $A$  is invertable and  $C - BAB^\dagger$  is positive semidefinite. Using this fact, we just need to find  $a$  such that

$$\frac{1}{2^{\binom{n}{k}}} I \succeq 2a^2 \binom{n}{k-1} \Gamma \Gamma^\dagger.$$

We have

$$\Gamma \Gamma^\dagger = I_{f^{-1}(1)}.$$

Since largest eigenvalue of  $\Gamma \Gamma^\dagger$  is 1, we can set  $a$  such that  $2a^2 \binom{n}{k-1} = \frac{1}{2^{\binom{n}{k}}}$ , so  $a = \sqrt{\frac{1}{4^{\binom{n}{k}} \binom{n}{k-1}}}$ . The objective value is

$$\sqrt{\frac{1}{4^{\binom{n}{k}} \binom{n}{k-1}}} \cdot k \cdot \binom{n}{k} = k \sqrt{\frac{n-k+1}{4k}} = \Omega(\sqrt{nk}).$$

This shows that any quantum algorithm with unbounded space requires  $\Omega(\sqrt{nk})$  queries. Now we'll look at our space-bounded SDP.

**4.2. Space Adversary Bound.** Now, we are able fix some space parameter  $S$  and choose the value  $\lambda \geq 0$ . Besides  $\lambda$ , the only difference from the above construction is that we now set  $|\beta\rangle = \frac{1}{\binom{n}{k-1}} \sum_{y \in f^{-1}(0)} |y\rangle$ . Now we want to assign  $a$  and  $\lambda$  as to maximize the objective value as long as

$$\begin{bmatrix} \left( \frac{1}{\binom{n}{k-1}} + \lambda \right) I & -a\Gamma^\dagger \\ -a\Gamma & \lambda I \end{bmatrix} \succeq 0$$

or equivalently,

$$\lambda I \succeq a^2 \left( \frac{1}{\binom{n}{k-1}} + \lambda \right)^{-1} I.$$

Setting  $a = \sqrt{\frac{\lambda}{\binom{n}{k}} + \lambda^2}$ , we have that the objective value is

$$\Theta\left(\max\left\{k\sqrt{\lambda\binom{n}{k}}, \lambda k\binom{n}{k}\right\} - \lambda 2^S\right),$$

which is unbounded upon maximizing over  $\lambda \geq 0$  unless  $S = \Omega(k \log n)$ .

This result shows that our trace heuristic is a poor fit for this problem. This is because it is computable in logspace (by simply summing over all indices), whereas if this heuristic was asymptotically tight, it would imply that  $\text{THRESHOLD}(n, k)$  is uncomputable in  $o(k \log n)$  space. In the next section, we'll briefly go over some other directions that are more likely to give a provable lower bound on  $S$ -constrained space complexity.

**5. Future Work.** We introduced a (non-convex) minimization problem, parameterized by a Boolean function  $f$ , whose optimum is asymptotically equal to the minimum quantum query complexity of any algorithm computing  $f$  with at most  $S$  qubits. In an attempt to prove *lower bounds* on the space-constrained query complexity, we provide a heuristic for the non-convex constraint as to find feasible solutions of the dual optimization problem. We find the nontrivial asymptotic solutions to this dual program for the Boolean function  $\text{THRESHOLD}(n, k)$ . Overall, it seems that using trace as a heuristic for rank may be a poor fit for this problem. Nevertheless, if we can show that any low-rank feasible solution of our primal program has low trace, then we can use our results to prove nontrivial lower bounds on space-constrained query complexity.

The problem with the trace constraint is that it is simply a heuristic, meaning lower bounds on the optimum of the SDP aren't necessarily lower bounds on the non-relaxed SDP (we saw this to be the case in the previous section). One possible solution is to replace the rank constraints with constraints that still include all rank  $2^S$  solutions. Moreover, if it's possible to dualize the SDP with these constraints, then they need not be convex. Similar ideas were recently successful in the context of tensor quadratic programs [8], which can model our SDP.

For example, we could instead have a new variable  $Y \in \mathbb{R}^{2^S \times n|D|}$  (representing the Gram vectors  $|\psi_{z,j}\rangle \in \mathbb{R}^{2^S}$  of  $X$ ). Our rank constraint is equivalent to requiring  $X = YY^\dagger$ . The idea is that if we can add constraints with respect to  $X$  and  $Y$  while maintaining that there's an assignment in which  $X = YY^\dagger$  is an optimal solution, then the optimum of this new SDP will lower bound the optimum of the rank-constrained SDP.

## REFERENCES

- [1] S. AARONSON, *Open problems related to quantum query complexity*, ACM Transactions on Quantum Computing, 2 (2021), pp. 1–9.
- [2] A. BELOVS, *Applications of the adversary method in quantum query algorithms*, arXiv preprint arXiv:1402.3858, (2014).
- [3] X. BONNETAIN, A. CHAILLOUX, A. SCHROTENLOHER, AND Y. SHEN, *Finding many collisions via reusable quantum walks*, arXiv preprint arXiv:2205.14023, (2022).
- [4] Y. HAMOUDI AND F. MAGNIEZ, *Quantum time-space tradeoff for finding multiple collision pairs*, arXiv preprint arXiv:2002.08944, (2020).
- [5] S. JEFFERY, *Span programs and quantum space complexity*, Theory OF Computing, 18 (2022), pp. 1–49.
- [6] H. KLAUCK, R. SPALEK, AND R. DE WOLF, *Quantum and classical strong direct product theorems and optimal time-space tradeoffs*, SIAM Journal on Computing, 36 (2007), pp. 1472–1493.
- [7] L. LI AND M. SHIRLEY, *The general adversary bound: A survey*, arXiv preprint arXiv:2104.06380, (2021).
- [8] R. MADANI, M. ASHRAFIJUO, M. KHEIRANDISHFARD, AND A. ATAMTURK, *Parabolic relaxation for quadratically-constrained quadratic programming-part i: Definitions & basic properties*, arXiv preprint arXiv:2208.03622, (2022).
- [9] B. REICHARDT, *Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function*, in 2009 50th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2009, pp. 544–551.

## PERFORMANCE PORTABLE RIGID BODY DYNAMICS

ANDERS JOHANSSON\* AND STAN MOORE†

**Abstract.** Rigid body dynamics are an important extension of traditional molecular dynamics simulations. LAMMPS, one of the leading molecular dynamics codes, has previously only supported CPU execution of rigid body dynamics — a major limitation in the modern GPU-dominated high performance computing landscape. This project implements rigid body dynamics in LAMMPS using the Kokkos performance portability library, to ensure efficient execution on both CPUs and GPUs from different vendors, as well as future hardware architectures. We benchmark and demonstrate the correctness, scalability and performance of the implementation in its current state and outline future plans for feature completeness and performance optimizations.

**1. Introduction.** LAMMPS has been one of the leading molecular dynamics codes for nearly three decades [5]. Throughout this time, its capabilities have rapidly expanded to go far beyond the simple atomistic Newtonian dynamics of traditional molecular dynamics simulations. One such extension is rigid body dynamics, which is conceptually simple but poses challenges both mathematical and computational.

There are two main motivations for rigid body dynamics. First, this constraint on particle motion can remove high-frequency degrees of freedom, thus allowing time integration with a larger timestep. A typical example is molecules containing lighter atoms such as hydrogen, whose motion can often be ignored while still describing the larger scale molecular motion with sufficient accuracy. Second, these constraints can be motivated by simulating physical processes involving rigid bodies, often applied to non-equilibrium molecular dynamics simulations. For example, nanoindentation simulations often employ a rigid indenter, commonly combined with rigid slabs to steer the indentation process.

From a computational perspective, rigid bodies can be handled in different ways, with the optimal choice of approach depending heavily on the problem at hand. In its initial implementation of rigid bodies, LAMMPS maintains copies of all rigid bodies on all processors, with the total force and torque on each rigid body requiring global communication. This method is ideal for simulations containing a few large rigid bodies that actually span the majority of processors, while it imposes a communication overhead leading to poor scalability in the case of small rigid bodies that only span a small subset of the processors. For the second case, LAMMPS gained a separate implementation in 2019 [4] that only requires local communication.

Concurrently with the addition of new simulation capabilities, LAMMPS is also responding to the evolving hardware in the world of high performance computing. Over the past decade, GPUs have gone from niche hardware to omnipresent, with 7 out of the top 10 supercomputers in the world having the majority of their computing power in GPUs. Among scientific codes, LAMMPS has been at the forefront of maintaining performance portability between CPUs and GPUs, and between GPUs from different vendors, through the use of the Kokkos performance portability library [6]. The process of porting LAMMPS functionality to Kokkos is ongoing, and rigid body dynamics has been among the missing features. Meanwhile, HOOMD-blue [1] supports rigid body dynamics on GPUs [3], albeit only on NVIDIA GPUs with experimental AMD GPU support.

This project implements performance portable rigid body dynamics in LAMMPS through Kokkos. The second algorithm mentioned above, specialized for small rigid bodies, was chosen due to its improved scalability [4] and wider applicability.

---

\*Harvard University, andersjohansson@g.harvard.edu

†Sandia National Laboratories, stamoor@sandia.gov

## 2. Methods.

**2.1. Molecular dynamics.** Molecular dynamics is the study of molecules, materials, etc. under the assumption of Newtonian atomic motion. Each atom  $i$  is treated as a point particle following Newton's second law as its equation of motion,

$$m_i \ddot{\vec{r}}_i = m_i \vec{a}_i = \vec{F}(\{\vec{r}_j - \vec{r}_i\}), \quad (2.1)$$

where the forces are computed by some given function of the relative atomic positions. The equations of motion are then integrated numerically, with the Velocity-Verlet algorithm being a common choice,

$$\vec{r}_i(t + \Delta t) = \vec{r}_i(t) + \vec{v}_i(t)\Delta t + \frac{1}{2}\vec{a}_i(t)(\Delta t)^2, \quad (2.2)$$

$$\vec{v}_i(t + \Delta t) = \vec{v}_i(t) + \frac{1}{2}(\vec{a}_i(t) + \vec{a}_i(t + \Delta t))\Delta t. \quad (2.3)$$

LAMMPS supports various extensions to the equations of motion to simulate different thermodynamic ensembles through thermostats and barostats. Finally, molecular dynamics simulations of bulk systems employ periodic boundary conditions to mimic infinite-size systems with finite numbers of particles.

**2.2. Rigid body dynamics.** When simulating rigid bodies, the motion of each particle according to its individual forces (eq. (2.1)) is replaced with the collective Newtonian motion of a body of particles according to the *total* force and torque on the body. Thus the center of mass  $\vec{R}_b$  of rigid body  $b$  with total mass  $M_b$  evolves according to

$$M_b \ddot{\vec{R}}_b = \sum_{i \in b} \vec{F}_i. \quad (2.4)$$

In a molecular dynamics simulation containing rigid bodies, the particles belonging to rigid bodies are excluded from the Verlet integration in eqs. (2.2) and (2.3). Instead, the center of mass, angular velocity and quaternion (i.e., orientation) of the rigid body evolve according to the total force and torque on the rigid body, and the positions and velocities of the constituent particles are updated accordingly, with the mathematical details given in [3].

**2.3. Spatial decomposition.** In order to use multiple processors or GPUs, LAMMPS needs a way to distribute the work of computing forces, time integration, etc. With the common assumption that atoms only interact with their neighbors that are within some finite cut-off distance, the natural choice of work distribution is a spatial distribution, where each processor is responsible for the set of atoms living inside its spatial simulation domain. For example, in a cubic simulation box with a cubic number of processors, each processor will be responsible for one cubic subset of the domain.

With this spatial decomposition, there are two kinds of communication needed between processors. First, atoms need to be migrated between processors when they move from one processor domain to another. This also requires communicating all the attributes belonging to those atoms. By maintaining a larger-than-necessary list of atom-neighbor pairs, the frequency with which atoms need migration is reduced, and the communication overhead is mitigated. The second kind of communication pertains to the atoms located near the boundary between processor domains, in what is often referred to as the halo region. These atoms interact with the atoms across the processor boundary, thus their positions need to be communicated to the neighboring processor. In the mode where LAMMPS uses Newton's third law to halve the number of pairwise interactions, the forces on an atom also need to be communicated back to its "home" domain from the neighboring domains. The copies of an atom maintained by the neighboring processors are referred to as "ghost" atoms.

For rigid body dynamics, communication between processors has proven to be the least straight-forward part of the implementation, compounded by the choice of the small-body algorithm with local communication over the large-body algorithm with global data structures and collective communication. In the small-body algorithm, each rigid body is “owned” by the atom closest to its geometric center. During the first “exchange” kind of communication, all the properties of the rigid body owned by an atom need to be migrated along with the atom. Then, at every timestep, the positions, velocities, angular momenta, etc. for bodies owned by ghost atoms need to be communicated to the corresponding processors. Finally, the forces on atoms belonging to rigid bodies owned by ghost atoms need to be reverse communicated to the “home” processors of those ghost atoms. For all of these communication steps, data only needs to be transferred for ghost atoms that own a rigid body, leading to irregular communication patterns.

**2.4. Implementation.** The implementation is essentially a port of the non-Kokkos implementation by Nguyen and Plimpton [4], described in detail in section 4 of that paper. We here describe design choices and challenges particular to the Kokkos implementation.

**2.4.1. Additions to the rest of LAMMPS.** In order to fully avoid data transfer between the CPU and the GPU, the Kokkos-enabled inter-process communication routines need to be used. Prior to this project, LAMMPS only supported “forward” and “exchange” communication for dynamics modifiers (“fixes”) such as rigid body dynamics. We have added Kokkos-enabled “reverse” communication, which is required for contributing the forces on an atom belonging to a rigid body that is owned by a ghost atom. Furthermore, we have made slight modifications to allow changing between non-Kokkos and Kokkos communication. This allows the Kokkos version of rigid body dynamics to call the non-Kokkos routines for creating the initial rigid body data structures, which use non-Kokkos communication. After the data has been created and copied to the GPU, a flag is changed to signal that Kokkos-enabled communication should be used. Previously, this flag was only ever checked once, which made it impossible to combine different modes of communication.

Various helper routines have also been ported to Kokkos. The mathematical functionality for dealing with quaternions, moments of inertia, etc. had already been ported on a separate, inactive pull request, and these were copied into this project. Additionally, new helper functions were made for dealing with periodic boundaries and periodic image remapping that can be called within Kokkos kernels on arbitrary data. Prior functionality only supported loops over the entire, LAMMPS-owned list of atomic coordinates, while rigid body dynamics requires remapping of center-of-mass coordinates and relative atomic distances therefrom.

**2.4.2. Data movement.** The typical approach in Kokkos-enabled LAMMPS features is to use so-called dual views of the data, which maintain one copy on the CPU (“host”) and one copy on the GPU (“device”), with the host copy pointing to the same data as the non-Kokkos version. Then, whenever data is needed somewhere, Kokkos is told to synchronize the data to that memory space, which requires a data copy if the data has been modified somewhere else. After modifying the data, the dual view is marked as modified on the corresponding memory space. This ensures maximal compatibility with the non-Kokkos version and enables gradual porting of functionality, at the cost of extensive bookkeeping.

For the rigid body Kokkos implementation, we instead chose the approach where the data simply never leaves the GPU after its initial creation. Before the simulation, the legacy non-Kokkos functions are called for setting up the rigid body data structures on the CPU, but then these are copied to the GPU and never again accessed on the CPU. This avoids the bookkeeping of modifications and synchronizations, at the cost of having to write the entire implementation before testing it. A limitation is that this requires the Kokkos version

of sorting and atom migration to be used even when running the Kokkos code on the CPU, e.g., with OpenMP, while this would normally resort to the non-Kokkos routines when not specifying certain command-line flags.

**2.4.3. Communication.** Compared to most other non-core LAMMPS functionality, rigid body dynamics requires more involved inter-process communication. In typical use cases, the majority of atoms neither own nor belong to rigid bodies, thus the data communicated is highly irregular compared to other LAMMPS extensions with a fixed number of per-atom properties. The severe degree of irregularity also implies that the irregularity should be exploited for performance, rather than, for example, simply packing “dummy” data for atoms not owning rigid bodies.

In the serial non-Kokkos code, the irregularity poses less of a problem because the data buffers are packed with a serial loop over the atoms, with the varying amount of data being sent per atom tallied up along the way. The cumulative sum of data then determines where the buffer for the next atom begins. In parallel code, such as OpenMP and CUDA through Kokkos, cumulative sums are problematic because they require synchronizations of the partial sums.

*Atom migration.* For the atom migration that only occurs infrequently, whenever neighbor lists need to be rebuilt, our initial implementation simply sends the same amount of data for all atoms, with most of the data being disregarded for the atoms not belonging to or owning a rigid body. While this is a potential point of future optimization, simple benchmarking shows that this extra communication has only a negligible performance impact. Instead, more effort has been put into the repacking of the non-migrated atoms and their rigid bodies, as well as the packing of the incoming atoms. In particular, when atoms are migrated away from a processor, the departing atoms leave “holes” in the data structures. For the per-atom data, LAMMPS provides a “copy list” of which atoms should be copied into those holes from the end of the atom list. For the rigid bodies themselves, however, there is no one-to-one correspondence between atoms and rigid bodies, thus a separate copy list needs to be constructed for the rigid bodies. This is illustrated in fig. 2.1. In the parallel implementation, the procedure for this specific case is as follows:

1. Count how many bodies have migrated away (3), mark their spots as “open”. There are  $9 - 3 = 6$  bodies that need to be densely packed. This step is performed in the same kernel as the one copying per-atom data (such as body ownership).
2. Count how many of the final 3 bodies need to be repacked (2), and do a cumulative count to remember their locations (6 and 8).
3. Loop through the first 6 slots, do a cumulative count to compute the number of each open spot. Slot 0 is open slot 0, slot 4 is open slot 1. Then fill in the corresponding bodies that need moving (from slots 6 and 8).

The initial counting of open spots is a parallel reduction over the number of open spots, while the remaining steps require cumulative sums for which Kokkos provides the `parallel_scan` routine. This performs cumulative sums efficiently across different hardware architectures using a multi-pass algorithm.

On the receiving side, the unpacking of the received data buffer is far simpler. An initial loop over the received atoms unpacks the per-atom data (such as body ownership) and simultaneously counts the number of received rigid bodies with a reduction. Then, a `parallel_scan` over the atoms does a cumulative sum to compute the relative indices of the rigid bodies and contiguously pack them into the local data structure. If, for example, the processor in fig. 2.1 receives two rigid bodies, they will be packed into slots 6 and 7. If it receives more rigid bodies than the currently allocated array can hold, it will first reallocate the data and then fill in the new rigid bodies from slots 6 and onwards.

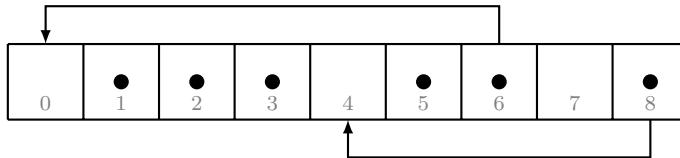


FIG. 2.1. *The problem of packing rigid bodies (black dots) densely after some rigid bodies have migrated away from the processor. In this case, three rigid bodies have left the processor, leaving open spots. As indicated by the arrows, the last two rigid bodies are moved to the first two open spots for a dense packing.*

*Forward and reverse communication.* At every timestep, the rigid body dynamics requires several passes of forward communication and one of reverse communication. These steps communicate data pertaining to atoms close to processor boundaries that own rigid bodies, as copies of these atoms are created as “ghost” atoms on the neighboring processors. The forward communication of updated center-of-mass coordinates, quaternions etc. to the neighboring processors is required for correct calculation of torques and other quantities for rigid bodies owned by those ghost atoms. The reverse communication step sends the forces and torques contributed by non-ghost atoms on rigid bodies owned by ghost atoms back to the “home” processor of the ghost atoms for use in time integration.

In the non-Kokkos communication routines, this packing of data is once again handled very simply in a serial loop over the atoms, with data packed only for those atoms that own a rigid body. The initial Kokkos implementation handled this irregular data in the same way as for the atom migration, with disregarded dummy data for atoms not owning rigid bodies. Due to these communication steps occurring at every single timestep, this extraneous data caused a noticeable communication overhead, indicating that a more optimized communication implementation was required.

The optimized version avoids looping over the atoms and instead maintains a list of rigid bodies that need to be communicated. This simultaneously both reduces the number of iterations in the loop and eliminates any need for cumulative counts of rigid bodies, with the latter otherwise being required for a dense packing of the minimal amount of data. The list of rigid bodies to send is created once after the atom migration, and serves the same purpose as the list of atoms to send created by LAMMPS. On the receiving side, the received bodies are packed contiguously for each neighboring processor. The sending buffers and the number of rigid bodies sent and received are stored at the migration step and reused for every subsequent timestep.

Fig. 2.2 shows the forward communication schematically. The sending processor has a list of four atoms that need to be sent to the receiver, two of which own rigid bodies. The initial implementation, as well as the non-Kokkos implementation, performed a loop over the atoms to see which ones own rigid bodies. Since this information is static between each atom migration, the optimized implementation stores this list of bodies that need to be communicated. Subsequent steps only perform a loop over those rigid bodies and pack the data indicated by the dashed arrows.

### 3. Results.

**3.1. Benchmark system: A cube of cubes.** For testing the correctness and performance of our implementation, we constructed a test system that would invoke all functionality such as atom migration, forward and reverse communication, and sorting of atoms. The unit cell of the constructed system contains 1000 cubes packed inside a cubic cell. Each cube has lattice parameter 1 and one atom in each corner. The lattice parameter of the whole cell is 30, and the cubes were packed together using PACKMOL [2]. The particles interact using a

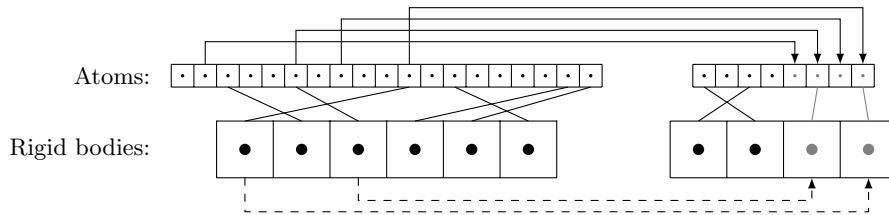


FIG. 2.2. *Forward communication of atoms and their rigid bodies. Gray circles and lines on the receiving processor represent ghost atoms and their rigid bodies. The per-atom data is communicated with the solid arrows after the atom migration and do not need updating until the atoms again migrate. At every timestep, the rigid body information requires updating, indicated by the dashed arrows.*

pairwise Lennard-Jones potential with a cut-off distance of 2.5,  $\varepsilon = 1$  and  $\sigma = 1$ . This setup maximizes the performance impact of the rigid body implementation, thus stress testing the implementation, by choosing one of the fastest pair potentials, letting all atoms be part of a rigid body, and having only 8 atoms for every rigid body.

**3.2. Correctness.** Molecular dynamics is inherently chaotic in nature, thus trajectories will eventually diverge due to minuscule differences such as different orderings of floating point additions. Still, the different trajectories should remain statistically equivalent with thermodynamic variables oscillating around the same mean value with fluctuations of the same magnitude. One such thermodynamic variable is the potential energy of a system, shown in fig. 3.1 for the non-Kokkos implementation run in serial on CPUs and in parallel on NVIDIA GPUs, as compared to the previous non-Kokkos implementation. For the first few thousands of steps, including the approach to equilibrium, the potential energies, and thus the trajectories, are virtually identical, before the numerical differences become large enough to make the trajectories diverge.

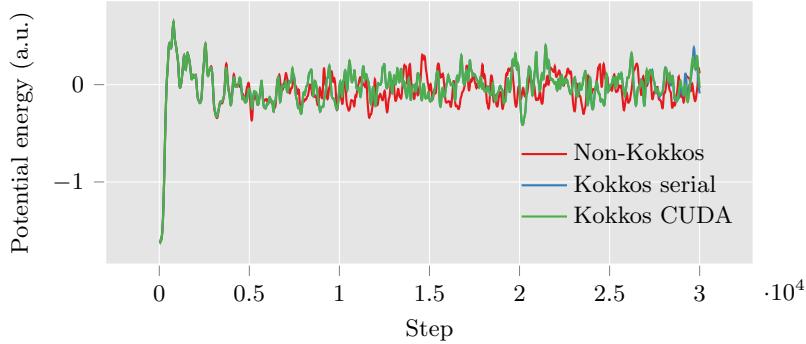


FIG. 3.1. *Correctness check for the new Kokkos implementation run in serial and on GPUs by tracking the potential energy as a function of the timestep for 1000 cubes. The potential energy is practically indistinguishable for several thousands of steps, thus including several rounds of communication and atom sorting, before numerical differences eventually cause a divergence due to the inherent instability of molecular dynamics. Even after the divergence, the results remain statistically equivalent.*

**3.3. Performance portability.** While the main goal of a Kokkos implementation is the use of hardware accelerators such as GPUs, its promise of performance portability indicates that a good implementation should run efficiently also on traditional CPUs. Fig. 3.2 shows a performance comparison between the non-Kokkos version and the Kokkos version running on both CPUs and GPUs. The CPU implementations were run on an AMD node

with 128 MPI tasks, while the GPU benchmarks were run on 4 A100 GPUs. While Kokkos run in serial on CPUs can sometimes be slightly slower than the non-Kokkos version due to algorithmic choices favoring the GPU, the optimized communication implementation allows the Kokkos-enabled rigid body dynamics to be ever so slightly faster on CPU than the non-Kokkos implementation.

When there are few atoms and rigid bodies, the simulation does not efficiently take advantage of the massive parallelism offered by GPUs with their many cores, leading to lower performance than the CPU implementations. This is compounded by the launch latency of GPU kernels. When the number of atoms increases to a few hundreds of thousands, however, we see that the GPU starts outperforming the CPU implementations, with the gap in performance increasing with the number of atoms. For even larger system sizes, the GPUs would be properly saturated, and the time per timestep would become linear in the number of atoms.

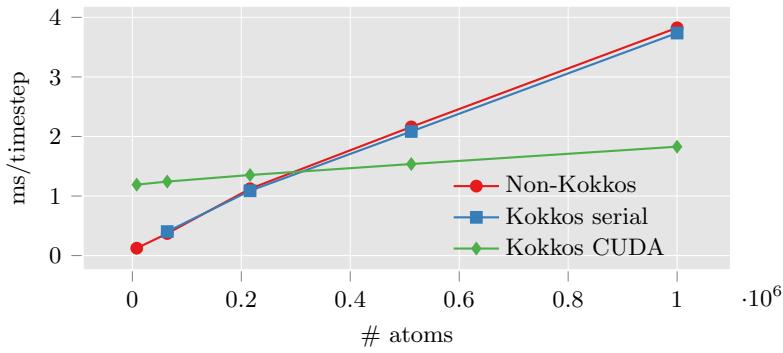


FIG. 3.2. *Performance comparison of the non-Kokkos implementation with the new Kokkos implementation run in serial and on GPUs for varying numbers of atoms. The CUDA numbers used 4 A100s, while the CPU numbers used 128 MPI tasks on one AMD EPYC node. With few atoms, there is not enough work to saturate the GPUs, leading to low performance, while the GPU outperforms the CPU implementations for larger numbers of atoms. The optimized communication implementation allows the Kokkos code to slightly outperform the non-Kokkos implementation even on CPUs.*

#### 4. Remaining work.

**4.1. Feature completeness.** LAMMPS is an exceptionally flexible molecular dynamics code, and the previous non-Kokkos implementation of rigid body dynamics is no exception. While this is great for users, it vastly increases the development effort of covering every corner case and combination of functionalities. As of writing, the Kokkos implementation appears correct and reasonably performance portable for the base case of running a single simulation of rigid body dynamics without any modifications to the dynamics, such as thermostats and barostats, or varying the number of rigid bodies, as done by for example the granular package in LAMMPS. Furthermore, additional functionality such as writing and restarting from restart files, multiple subsequent simulations, etc. remain untested.

The necessity of adding these features to the Kokkos implementation will be assessed based on demand and frequency of use. Restart files and subsequent implementations will certainly be tested and corrected as necessary. Thermostats and probably barostats will also be supported, which involves making subclasses that overload certain integration methods. For the cases that are deemed to be beyond the point of diminishing return for the development effort, we will make sure that these gracefully fall back to the non-Kokkos implementation.

#### 4.2. Optimizations.

*Setup.* As shown in [4], the creation of rigid bodies can take significant time in the extreme case of millions of rigid bodies on tens of thousands of processors, thus the non-Kokkos implementation goes to great lengths to optimize this operation via a rendezvous scheme. This parallelizes the creation of rigid body data structures by assigning each rigid body and its constituent atoms to a random processor that computes moments of inertia, quaternions, etc., before communicating the results back to the processor owning the rigid body via all-to-all operations. The current Kokkos implementation simply uses the same routines and only copies the data to the Kokkos data structures at the end of the setup step. Future work may include porting also this part of the simulation to Kokkos for an optimal implementation.

*Dynamics.* In terms of the number of floating point operations and memory read/write operations, the Kokkos implementation should perform slightly fewer than the non-Kokkos version (due to the optimized communication buffer packing), but on GPUs, the number of operations is not the only thing that matters. First of all, GPUs lean heavily towards arithmetic intensity, essentially how much you do with each number that you read from memory. Whereas traditional CPUs have been balanced to do roughly one operation with each number, GPUs are optimized for operations such as matrix-matrix multiplication, which do  $N^3$  operations on  $N^2$  data. Rigid body dynamics are suboptimal in this regard, as all steps of the simulation do a small handful of operations for each rigid body.

Another important aspect of optimization for GPUs is the memory layout. On CPUs, threads are independent workers that should be assigned contiguous chunks of memory for their work. GPUs, on the other hand, require a team of threads to work on one contiguous chunk of data together, before they all move on to the next contiguous chunk (called memory *coalescence*). The current Kokkos implementation uses the same data structures as the non-Kokkos implementation, with each rigid body stored contiguously in memory. This is optimal for the CPU, but not for the GPU, which would prefer each property of all the rigid bodies (for example, the  $x$ -component of all the centers of masses) to be contiguous by itself. For tackling this problem of disagreeing memory layouts for different hardware architectures, Kokkos has a memory layout abstraction for multidimensional arrays that automatically switches between row-major and column-major layouts depending on the targeted architecture. The loops over rigid bodies could thus be significantly accelerated by abandoning the one-dimensional array of rigid body objects in favor of multidimensional arrays of rigid body properties, at the cost of some readability and convenience.

**5. Conclusion.** An initial performance portable version of rigid bodies has been implemented in LAMMPS using the Kokkos library, specifically the algorithm targeting small rigid bodies that involves only local communication. Additional features have been added to the remainder of the LAMMPS source code as necessary, such as Kokkos-enabled reverse communication for “fixes”. We have put significant effort into optimizing the rigid body communication routines, in terms of both avoiding non-Kokkos work and minimizing the amount of data sent and the size of the loops required for packing the data. The resulting implementation is slightly faster when running Kokkos in serial than the previous non-Kokkos implementation on CPUs. On GPUs, significant acceleration is achieved when the system size approaches one million atoms per node, but work remains on optimizing the memory layouts for memory coalescence. The Kokkos implementation has also been tested for correctness on a fairly general benchmark system containing one thousand small rigid cubes. As of writing, it only supports the base case of classical rigid body dynamics, with significant development work still required for supporting the full suite functionality available in the non-Kokkos implementation, such as thermostats, barostats and granular simulations.

## REFERENCES

- [1] J. A. ANDERSON, J. GLASER, AND S. C. GLOTZER, *HOOMD-blue: A Python package for high-performance molecular dynamics and hard particle Monte Carlo simulations*, Computational Materials Science, 173 (2020), p. 109363.
- [2] L. MARTÍNEZ, R. ANDRADE, E. G. BIRGIN, AND J. M. MARTÍNEZ, *PACKMOL: A package for building initial configurations for molecular dynamics simulations*, Journal of computational chemistry, 30 (2009), pp. 2157–2164.
- [3] T. D. NGUYEN, C. L. PHILLIPS, J. A. ANDERSON, AND S. C. GLOTZER, *Rigid body constraints realized in massively-parallel molecular dynamics on graphics processing units*, Computer Physics Communications, 182 (2011), pp. 2307–2313.
- [4] T. D. NGUYEN AND S. J. PLIMPTON, *Aspherical particle models for molecular dynamics simulation*, Computer Physics Communications, 243 (2019), pp. 12–24.
- [5] A. P. THOMPSON, H. M. AKTULGA, R. BERGER, D. S. BOLINTINEANU, W. M. BROWN, P. S. CROZIER, P. J. IN'T VELD, A. KOHLMAYER, S. G. MOORE, T. D. NGUYEN, ET AL., *LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales*, Computer Physics Communications, 271 (2022), p. 108171.
- [6] C. R. Trott, D. LeBrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibáñez, ET AL., *Kokkos 3: Programming model extensions for the exascale era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2021), pp. 805–817.

## GENERATING KOKKOS CODE FROM SPARSE TENSOR ALGEBRA IN PYTHON USING MLIR

ALI KHAN\*, BRIAN KELLEY†, KIM LIEGEOIS ‡, AND SIVA RAJAMANICKAM §

**Abstract.** Sparse tensors are ubiquitous in scientific computing and deep learning. Unfortunately, operations on these sparse tensors are still lacking in large frameworks and libraries, much less for the parallel use case. Moreover, modern deep learning and scientific high performance computation (HPC) often rely on different software and hardware ecosystems. Researchers are also relying more on Python instead of the C++ code that traditional HPC software is written in. We demonstrate a method to take sparse tensor algebra written in Python and use Multi-Level Intermediate Language (MLIR) to compile and generate parallel C++ using the Kokkos library, an ecosystem for performance portability.

**1. Introduction.** Machine learning and scientific computing software are essential tools for the modern researcher’s toolbox. Deep learning has shown promise in wide array of fields and high performance computing is an effective tool for simulating complex problems. Matrices and linear algebra are ubiquitous in modern machine learning and scientific computing. Historically linear algebra on matrices have proven critical for fast and efficient computation and are a key component to modern deep learning and scientific computing. Tensor algebra, a generalization of vectors and matrices, is a powerful tool for handling high dimensional data [4]. While large dense tensors would consume a large amount of memory, most real-world tensors are sparsely populated allowing for a much smaller memory footprint and faster computation. Support for sparse tensor algebra in software frameworks has increased in recent years, but additional work is still required in order to integrate it into optimized numerical linear algebra libraries such as Eigen[5] and Tpetra[14, 15].

Taking advantage of the sparsity of tensors often requires development of hand-optimized algorithms that are often error-prone and difficult to maintain. Recent work have tried to address the gap in support for sparse tensors by moving the tedious implementation details of sparse tensor support to the compiler instead [8]. Other related projects such as Torch-MLIR [10] and MLIR-HLO [13] utilize the Multilevel Intermediate Representation (MLIR) compiler ecosystem in the LLVM project [9] to address the issue of portability but are fragmented and targeted for particular use cases. The authors of [8] have extended their work to take advantage of the MLIR ecosystem to allow code written with TACO and produce MLIR code [1].

Previous work by Kelley et. al (2022) [7] have adapted the Torch-MLIR frontend to generate C++ code that uses the Kokkos [16] library to address the issue of portability. The Kokkos ecosystem [3] is a shared-memory parallel programming model for performance portability across multiple GPU and CPU architectures. The machine learning Python code written using the machine learning framework PyTorch [11] and is automatically compiled using MLIR to generate Kokkos-based C++ source code. This code can be executed on any Kokkos backend (CPU or GPU) allowing use in scientific applications across a variety of hardware. The proposed pipeline, however, is limited to dense matrices used in PyTorch ecosystem and lacks support for general sparse tensors.

Our contribution is as follows: we propose a pipeline that takes a sparse tensor algebra expression in Python, lowers the python code through MLIR, and emits C++ source code that utilizes the Kokkos ecosystem. In particular, we build on top of the work already

---

\*Sandia National Laboratories, [aykhan@sandia.gov](mailto:aykhan@sandia.gov),

†Sandia National Laboratories, [bmkelle@sandia.gov](mailto:bmkelle@sandia.gov)

‡Sandia National Laboratories, [knliege@sandia.gov](mailto:knliege@sandia.gov)

§Sandia National Laboratories, [srajama@sandia.gov](mailto:srajama@sandia.gov)

performed by Kelley et. al (2022) [7] and use Bik et. al (2022) [1] to support general parallel sparse tensor algebra. We further demonstrate our pipeline and MLIR-Kokkos backend in two applications: (1) Sparse matrix-vector multiplication (SpMV), and (2) Matricized tensor times Khatri-Rao product (MTTKRP).

## 2. Background.

**2.1. TACO.** TACO (Tensor Algebra COmpiler) is a C++ library that showcases a new system for sparse tensor algebra by moving the tedious implementation details for sparse support into the compiler and letting the user treat sparsity as a property instead. They also offer a PyTACO – a Python package with a domain specific language (DSL) for sparse tensor computation. Through PyTACO, researchers can easily create and operate on tensors in a mix of dense and sparse storage formats using index-notation. PyTACO uses a novel storage scheme that describes any tensor format by specifying a data type, storage order and whether each dimension is dense or sparse.

Tensors in PyTACO can be declared in a single line similar to Numpy [6] matrices but with an additional parameter for the tensor’s data storage format and storage order. A dense vector would be represented as `[dense]` while a compressed sparse row matrix would be represented as `[dense, compressed]`. Table 2.1 shows some of the common data formats and their PyTACO representations.

<code>[dense, dense]</code>	(Row-major) dense matrix format
<code>[dense, compressed]</code>	Compressed sparse row matrix format (csr)
<code>[compressed, dense]</code>	Compressed sparse column matrix format (csc)
<code>[compressed, compressed]</code>	Doubly compressed sparse column matrix format (dcsc)
<code>[compressed, compressed, compressed]</code>	Compressed sparse fiber tensor format (csf)

TABLE 2.1  
*Storage formats and their PyTACO equivalent*

Computing on tensors, both sparse and dense, are also simple by using tensor index notation. For example, the MTTKRP operation for a three-dimensional tensor matricized along the first dimension can be expressed in index notation as

$$A_{ij} = B_{ikl} \cdot D_{lj} \cdot C_{kj}$$

where  $A, C, D$  are two-dimensional tensors and  $B$  is a three-dimensional tensor.  $i, j, k$ , and  $l$  are the indices corresponding to the dimension of the tensors. Using the PyTACO API, the computation can similarly be expressed as

```
i, j, k, l = get_index_vars(4)
A[i, j] = B[i, k, l] * D[l, j] * C[k, j]
```

where the first line initialized the index variables  $i, j, k$ , and  $l$  with the function `get_index_vars()`. The PyTACO API moreover includes support for reductions and broadcasts across dimensions as well as expressing the transpose of tensor using index variables.

**2.2. MLIR.** MLIR, or Multi-Level Intermediate Representation — part of the LLVM open-source project — is a an ecosystem for developing compiler architecture [9]. It is fundamentally an abstract language specification and uses higher-level constructs that can be organized into different dialects. The LLVM IR itself is one low-level dialect. This multi-level approach allows for optimization making use of techniques such as polyhedral compilation

and static-single assignment (SSA) at a higher level while still offering efficient low-level code. A high level function such as `linalg.matmul()` which performs an accumulative matrix multiplication  $C += A \cdot B$ , produces loops and conditionals using the lower-level `scf` dialect. Likewise the `sparse_tensor` and `tensor` dialects offer high-level constructs for sparse tensor storage but produce memory reference allocations such as `memref.alloc()` operators in the `memref` dialect at the low-level. We utilize several different dialects in the lowering pipeline but focus on in particular the `sparse_tensor`, `linalg`, `scf`, and `memref` dialects below.

**2.2.1. Tensor and SparseTensor Dialects.** The MLIR `tensor` dialect along with the `sparse_tensor` dialect is a data allow sparse tensors to be expressed as high-level MLIR. These two dialects are responsible for the creation and manipulation of the abstract tensor type. Large and dynamically sized tensors are concretely represented in memory using the `memref` dialect through a lowering pass called bufferization. With the `sparse_tensor` dialect, tensors can be annotated with mixed sparsity and storage format with the dialect acting as a bridge between high-level tensor types and the actual low-level sparse storage schemes operating on only the non-zero elements.

**2.2.2. Linalg Dialect.** The `Linalg` dialect defines transformations and operations declaratively as linear algebra semantics, leveraging both high-level compiler optimizations as well as efficient library implementations. Some of its capabilities include lowering itself down to serial or parallel loops, operate on tensor or buffers, and parametric tiling.

**2.2.3. SCF Dialect.** The structured control flow `SCF` dialect offers looping and conditional operations that map to standard functions such as `for`, `while`, and `if`. The versatile `scf.parallel` operation describes a parallel loop over a multi-dimensional iteration space. A reduction can optionally be performed over a subset of dimensions. Other higher dialects such as `linalg` can be decomposed into operations of the `scf` dialect.

**2.2.4. Memref Dialect.** The `memref` dialect provides lower-level operations for working with memory buffers. As seen previously, `memref` operations map well to C-style memory management. Most higher-level data structures such as tensors can be lowered down to concrete `memref` operations.

**2.3. Kokkos.** The Kokkos ecosystem [16, 3], is a performance portable C++ ecosystem that allows HPC software to be written in an architecture agnostic way. It provides efficient abstractions for hardware features like SIMD and GPU shared memory. The Kokkos Core library implements the Kokkos programming model for several backends (CUDA, OpenMP, HIP and SYCL are some examples). A Kokkos-based application can therefore be compiled to target any of the common CPU and GPU architectures used in HPC. The Kokkos Kernels library is a suite of linear algebra and graph algorithm kernels implemented using Kokkos, while also leveraging platform-specific vendor libraries when possible [12].

Kokkos provides three main parallel operations: `parallel_for()`, `parallel_reduce()`, and `parallel_scan()`. The iteration space for an operation is described through an Execution Policy.

The fundamental Kokkos constructs of `Kokkos::View` and parallel patterns map well to MLIR. The `memref` dialect offers operations that work on strongly-typed multidimensional buffers which correspond to Kokkos multidimensional views while the `scf.parallel` operation maps to `Kokkos::parallel_for()`.

**3. Methodology.** Our pipeline takes Python mixed dense and sparse tensors that use the PyTACO API and emit parallel C++ Kokkos code. We accomplish this in several steps

as outlined in Figure 3. We take advantage of the lowering pipeline that already exists in the MLIR compiler ecosystem.

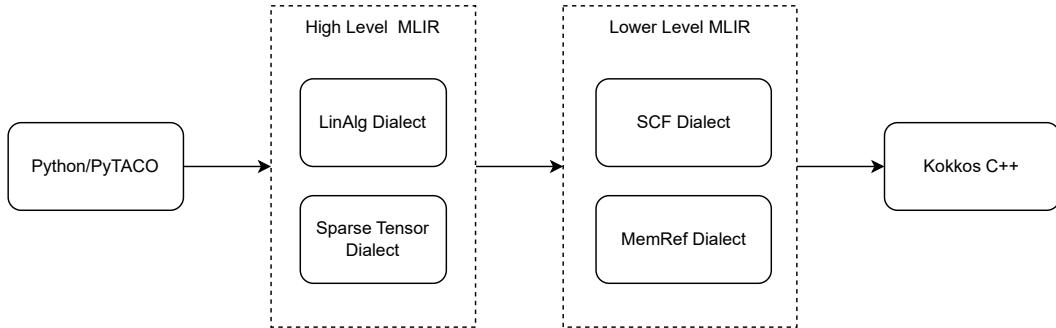


FIG. 3.1. *Outline of the Python to Kokkos pipeline*

Like Kelley and Rajamanickam (2022) [7], we also utilize an external frontend to convert Python code into high-level MLIR and produce Kokkos-based C++. But instead of using the Torch-MLIR frontend[10], we use MLIR-PyTACO API[1]. PyTACO offers several distinct advantages over PyTorch. First, it natively supports sparse tensor algebra. Second, through its support of a variety of tensor storage formats and simplified algebra, it offers a more natural use of tensors without excessive boilerplate. Lastly, MLIR-PyTACO has first class support within the MLIR project and the LLVM community at large, directly mapping to the `sparse_tensor` and `linalg` dialects.

The MLIR-Kokkos C++ emitter was expanded to support the PyTACO tensor format. Several new functions were added to the emitter, and MLIR-PyTACO was modified to support the Kokkos compilation stack instead of the LLVM JIT runtime. Table 3.1 lists the some internal functions used by functionality that was added to the Kokkos emitter. Since MLIR is in SSA syntax –variables are assigned and defined only once before they are used– emitting C++ is simple. The Kokkos emitter would generate a corresponding string for every MLIR operation. The C++ emitter was further modified to support the `sparse_tensor` dialect and the parallelization of the sparse tensors.

TABLE 3.1  
MLIR Sparse tensor-storage methods support added to the Kokkos emitter

Sparse Tensor Function	Description
<code>newSparseTensor</code>	Creates a sparse tensor, including memory allocation
<code>sparseValues</code>	Obtains direct access to the values array
<code>sparseIndices</code>	Obtains direct access to the indices array for the given dimension
<code>sparsePointers</code>	Obtains direct access to the pointers array for the given dimension
<code>lexInsert</code>	Inserts elements in lexicographical index order
<code>expInsert</code>	Inserts using expansion
<code>endInsert</code>	Finalizes lexicographic insertions

Unlike PyTorch which passes dense memrefs as raw data pointers to the Kokkos emitter, PyTACO passes memrefs as `StridedMemRefType<>` which requires machinery to convert them into `Kokkos::Views` and back. The raw data pointer and offset have one-to-one correspondence but the layout of the `StridedMemRefType<>` can affects the strides – the location of the next successive element of the tensor in memory. If the View’s layout is

`LayoutLeft`, then the stride of the memref for the first dimension should be 1 and the stride on subsequent dimensions,  $k$  must be the stride of the previous dimension  $k - 1$  multiplied by the size of dimension  $k - 1$ . If the View's layout is `LayoutRight`, then stride of the memref for the last dimension must be 1, and the stride of dimension  $k$  must be the stride of the successive dimension,  $k + 1$ , multiplied by the size of dimension  $k + 1$ . If the View's layout is instead `LayoutStrided`, then the the memref's strides can be anything.

Support for parallelization can be as simple as emitting `Kokkos::parallel_for()` function for every `scf.parallel` operation but mapping the iteration space requires using Kokkos Execution Policy. Listings 1 and 2 in the Appendix include the parallelization of SpMV. The nested `scf.parallel` operations get mapped to a nested `TeamPolicy`, in this case a `Kokkos::TeamVectorRange` loop with reductions.

Similar to Kelley and Rajamanickam (2022), we follow the above pipeline to generate a C++ source code file, then compile it into a native shared library. An auto-generated Python wrapper module then loads this library and exposes a Python interface to its functions using the CTypes API. The user can evaluate and store the result tensor to a file from Python using `mlir_pytaco_api.write_kokkos()`. Internally, the Python wrapper passes the input tensors to a function in the native library, the result is computed there, and finally is returned back to Python.

**4. Experimental Results.** We tested the pipeline by compiling two common tensor kernels: Sparse Matrix-Vector Multiplication (SpMV) and Matricized tensor times Khatri-Rao product (MTTKRP). SpMV is a fundamental building block for deep learning and scientific computation while MTTKRP is a bottleneck for tensor decompositions such as CP-ALS (canonical polyadic decomposition - alternating least squares). For the SpMV computation, we multiply the *pwtk* matrix – a 217918 by 217918 sparse matrix [2] – by a dense vector of ones. For the MTTKRP computation, we multiply a three dimensional tensor with sizes 2 by 2 by 4 with two subsequent ones-filled matrices of size 4 by 25. The execution time for the computation is recorded below in milliseconds and excludes initialization and the reading and writing of a file.

Table 4.1 shows the serial results for SpMV. *MLIR-LLVM JIT* is the pipeline from Python to MLIR and directly to the LLVM JIT compiler. *MLIR-Kokkos*, our contribution, refers to the Python to MLIR to Kokkos C++ pipeline as described before. The Appendix contains the snippets of the generated MLIR and emitted Kokkos-based C++ code for the SpMV example. *Serial Kokkos* refers to an independent serial installation of Kokkos that utilizes the the benchmark SpMV C++ code from the Kokkos Kernels project. Since there is no equivalent benchmark code for MTTKRP within the Kokkos Kernels project, the results for *Serial Kokkos* is left blank.

Table 4.2 compares the results of parallel execution of SpMV computations across an increasing number of OpenMP threads. For each computation the environmental variables utilized are '*OMP\_PROC\_BIND=spread*' and '*OMP\_PLACES=threads*', while '*OMP\_NUM\_THREADS*' is used to specify the number of threads.

TABLE 4.1  
Execution time (in ms) for generated code for serial SpMV and MTTKRP

	SpMV	MTTKRP
MLIR-LLVM JIT	39.045	52.686
MLIR-Kokkos	23.401	0.212
Serial Kokkos	12.293	-

TABLE 4.2  
*Execution time (in ms) for parallel (OpenMP) SpMV*

Threads	SpMV	
	Vanilla Kokkos	MLIR-Kokkos
1	17.942	20.066
2	12.244	15.983
4	6.432	10.042
8	3.503	5.391

**5. Discussion.** The results demonstrate a significant speedup of *MLIR-Kokkos* over the *MLIR-LLVM JIT* for both SpMV and MTTKRP. Furthermore, it also shows that while traditional Kokkos-based computations can be faster than the MLIR to Kokkos pipeline, the MLIR-Kokkos pipeline can still scale to take advantage of multiple threads.

Current generated Kokkos code is emitted from low-level MLIR, leaving room on the table for performance. By emitting C++ from a higher level, the MLIR-Kokkos emitter can grant use of optimized linear algebra from a Kokkos Kernel [12]. Furthermore, additional work is needed to explore different partitioning schemes of tensors for parallelization.

**6. Conclusion.** Sparse tensor algebra is a key component in deep learning and scientific computing, but support for sparse tensors across different architectures is lagging. Previous research showcased a pipeline for taking machine learning in Python and generating parallel C++ by automatically compiling into high-level MLIR, low-level MLIR, and emitting Kokkos-based C++. We build on top of this work and demonstrate parallel sparse tensor support for Python-based sparse tensor algebra and the performance of this pipeline on two types of linear algebra computations.

## REFERENCES

- [1] A. BIK, P. KOANANTAKOOL, T. SHPEISMAN, N. VASILACHE, B. ZHENG, AND F. KJOLSTAD, *Compiler support for sparse tensor computations in MLIR*, ACM Trans. Archit. Code Optim., 19 (2022).
- [2] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011).
- [3] H. C. EDWARDS, C. R. TROTT, AND D. SUNDERLAND, *Kokkos: Enabling manycore performance portability through polymorphic memory access patterns*, Journal of Parallel and Distributed Computing, 74 (2014), pp. 3202 – 3216. Domain-Specific Languages and High-Level Frameworks for High-Performance Computing.
- [4] T. GALE, M. ZAHARIA, C. YOUNG, AND E. ELSEN, *Sparse GPU kernels for deep learning*, in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2020, pp. 1–14.
- [5] G. GUENNEBAUD, B. JACOB, ET AL., *Eigen v3*. <http://eigen.tuxfamily.org>, 2010.
- [6] C. R. HARRIS, K. J. MILLMAN, S. J. VAN DER WALT, R. GOMMERS, P. VIRTANEN, D. COURNAPEAU, E. WIESER, J. TAYLOR, S. BERG, N. J. SMITH, R. KERN, M. PICUS, S. HOYER, M. H. VAN KERKWIJK, M. BRETT, A. HALDANE, J. F. DEL RÍO, M. WIEBE, P. PETERSON, P. GÉRARD-MARCHANT, K. SHEPPARD, T. REDDY, W. WECKESSER, H. ABBASI, C. GOHLKE, AND T. E. OLIPHANT, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.
- [7] B. KELLEY AND S. RAJAMANICKAM, *Unified language frontend for physic-informed AI/ML*, tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2022.
- [8] F. KJOLSTAD, S. KAMIL, S. CHOU, D. LUGATO, AND S. AMARASINGHE, *The tensor algebra compiler*, Proc. ACM Program. Lang., 1 (2017).
- [9] C. LATTNER, M. AMINI, U. BONDHUGULA, A. COHEN, A. DAVIS, J. PIENAAR, R. RIDDLE, T. SHPEISMAN, N. VASILACHE, AND O. ZINENKO, *MLIR: Scaling compiler infrastructure for domain specific computation*, in 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2021, pp. 2–14.
- [10] LLVM, *The Torch-MLIR Project*. <https://github.com/llvm/torch-mlir>, 2021.

- [11] A. PASZKE, S. GROSS, S. CHINTALA, G. CHANAN, E. YANG, Z. DEVITO, Z. LIN, A. DESMAISON, L. ANTIGA, AND A. LERER, *Automatic differentiation in PyTorch*, (2017).
- [12] S. RAJAMANICKAM, S. ACER, L. BERGER-VERGIAT, V. DANG, N. ELLINGWOOD, E. HARVEY, B. KELLEY, C. R. TROTT, J. WILKE, AND I. YAMAZAKI, *Kokkos kernels: Performance portable sparse/dense linear algebra and graph kernels*, arXiv preprint arXiv:2103.11991, (2021).
- [13] THE TENSORFLOW MLIR TEAM, *MLIR-HLO*. <https://github.com/tensorflow/mlir-hlo>, 2020.
- [14] THE TPETRA PROJECT TEAM, *The Tpetra Project Website*, 2020.
- [15] T. TRILINOS PROJECT TEAM, *The Trilinos Project Website*, 2020.
- [16] 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. TURKSIN, AND J. WILKE, *Kokkos 3: Programming model extensions for the exascale era*, IEEE Transactions on Parallel and Distributed Systems, 33 (2022), pp. 805–817.

## APPENDIX

```

scf.parallel (%arg2) = (%c0) to (%c5) step (%c1) {
    %7 = memref.load %6[%arg2] : memref<?xf64> loc(#loc)
    %8 = memref.load %2[%arg2] : memref<?xindex> loc(#loc)
    %9 = arith.addi %arg2, %c1 : index loc(#loc)
    %10 = memref.load %2[%9] : memref<?xindex> loc(#loc)
    %11 = scf.parallel (%arg3) = (%8) to (%10) step (%c1) init (%7) -> f64 {
        %12 = memref.load %3[%arg3] : memref<?xindex> loc(#loc)
        %13 = memref.load %4[%arg3] : memref<?xf64> loc(#loc)
        %14 = memref.load %5[%12] : memref<?xf64> loc(#loc)
        %15 = arith.mulf %13, %14 : f64 loc(#loc)
        scf.reduce(%15) : f64 {
            ^bb0(%arg4: f64 loc(unknown), %arg5: f64 loc(unknown)):
                %16 = arith.addf %arg4, %arg5 : f64 loc(#loc)
                scf.reduce.return %16 : f64 loc(#loc)
        } loc(#loc)
        scf.yield loc(#loc)
    } {"Emitted from" = "linalg.generic"} loc(#loc)
    memref.store %11, %6[%arg2] : memref<?xf64> loc(#loc)
    scf.yield loc(#loc)
} {"Emitted from" = "linalg.generic"} loc(#loc)

```

Listing 1: Generated low-level MLIR snippet for SpMV

```

// scf.parallel
typedef Kokkos::TeamPolicy<exec_space>::member_type member_type;
int league_size = (5 - 0 + 1 - 1) / 1;
Kokkos::TeamPolicy<exec_space> policy (league_size, Kokkos::AUTO(), Kokkos::AUTO());
Kokkos::parallel_for(policy, KOKKOS_LAMBDA(member_type member)
{
    int64_t unit_v20 = member.league_rank ();
    int64_t v20 = 0 + unit_v20 * 1;
    // memref.load
    double v21 = v19(v20);
    // memref.load
    size_t v22 = v15(v20);
    // arith.addi
    size_t v23 = v20 + 1;
    // memref.load
    size_t v24 = v15(v23);
    // scf.parallel
    double v25;
    Kokkos::parallel_reduce(Kokkos::TeamVectorRange(member, (v24 - v22 + 1 - 1) / 1),
    [&](const int64_t &unit_v26, double& v27)
    {
        int64_t v26 = v22 + unit_v26 * 1;
        // memref.load
        size_t v28 = v16(v26);
        // memref.load
        double v29 = v17(v26);
        // memref.load
        double v30 = v18(v28);
        // arith.mulf
        double v31 = v29 * v30;
        // scf.reduce
        v27 += v31;
        // scf.yield
        ;
    }, v25)
    ;
    // memref.store
    v19(v20) = v25;
    // scf.yield
    ;
})
;
// func.return
return v14;
}

```

Listing 2: Generated Kokkos C++ snippet for SpMV

## IS RMA THE WAY? PERFORMANCE INSIGHTS INTO DEVICE-INITIATED RMA USING KOKKOS REMOTE SPACES

DANIEL MISHLER\*, JAN CIESKO†, STEPHEN OLIVIER‡, AND GEORGE BOSILCA§

### **Abstract.**

Achieving scalable performance on supercomputers requires careful coordination of communication and computation. Applications using host-side communication libraries such as MPI often rely on highly tuned implementations of buffering, sorting, and clustering techniques to achieve performance goals. As interconnects between accelerators become more performant and scalable, programming models such as SHMEM may have the opportunity to enable bandwidth maximization along with ease of programming. In this work, we take a closer look at device-initiated PGAS programming models using NVIDIA Corp's NVSHMEM communication library and our interface through the Kokkos Remote Spaces project. We show that benchmarks can benefit from this programming model in terms of performance and programmability. We anticipate similar results for miniapplications.

**1. Introduction.** The path to Exascale has led us to accelerators, typically programmed using node-level parallel programming models in conjunction with MPI for cross-node communication. With vendors implementing GPU-centric interconnects such as NVLink and xGMI and providing programming libraries for device-initiated communication, can parallel programming models benefit from such APIs and find ways to expose them to the developer? Though it isn't the primary parallel programming paradigm for the high performance computing community, there has already been a substantial body of work on remote memory access (RMA), partitioned global address space (PGAS), and symmetric hierarchical memory (SHMEM)[1][4][3]. That said, the technology lacks traction compared to MPI. PGAS technology will surely suffer vendor fragmentation and even more difficulties should developers try to reinvent the wheel.

For the programming models community, this is an opportunity. Kokkos Remote Spaces, which is currently in its nascent years as a project[2], was developed for the purpose of exposing device-initiated RMA in a cross-vendor software layer built on top of the already successful Kokkos C++ performance portability framework[5]. The objective is to follow up on the Kokkos Paradigm for PGAS: to enable this technology with high gains to programmability at negligible cost to performance. Kokkos provides tools for parallelism such as `parallel_for` while Kokkos Remote Spaces provides tools on top of that for use of RMA.

In this work, we present results using Kokkos Remote Spaces as well as benchmarks that demonstrate the costs and benefits of this additional software layer. We summarize the results of other benchmarks, leaving their detailed analysis to future work. Our evaluation uses NVIDIA Volta100 GPUs leveraging NVIDIA's NVSHMEM communication library (<https://developer.nvidia.com/nvshmem>). The code we present can be accessed at <https://github.com/kokkos/kokkos-remote-spaces>.

**2. Implementation of Communications.** The Kokkos programming model provides a set of `View` classes that store and manage arrays of data. Kokkos offers basic `Kokkos::Views` (referred to as `Normal` views), `Unmanaged` views, and now `Remote` views. `Remote` views are new to Kokkos Remote Spaces, and their implementation leverages the RMA interface of the underlying hardware.

---

\*Sandia National Laboratories and the University of Tennessee, Knoxville, dsmishler@icl.utk.edu

†Sandia National Laboratories, jciesko@sandia.gov

‡Sandia National Laboratories, slolivi@sandia.gov

§University of Tennessee, Knoxville, bosilca@icl.utk.edu

**2.1. MPI and CUDA-aware MPI.** In order to send data from device to device in an MPI installation that is unaware of device pointers, the data must go from sender device to sender host, then to receiver host to receiver device. Consider the following parallel pseudocode, which sends an array  $B$  which belongs to process rank 1 to process rank 0 so that rank 0 can element-wise compute  $A(i) += B(i)$ . This uses a `Normal` view.

```

1 B_host <-- B_device;
2 if (myrank == 1) {
3     Send(B_host, 0);
4 }
5 if (myrank == 0) {
6     Receive(B_host, 1);
7 }
8 B_device <-- B_host;
9 while(i < len(B_device)) {
10     A_device(i) += B_device(i);
11     i++;
12 }
```

CUDA-aware MPI is aware of device pointers. It takes advantage of fast device-to-device interconnects such as NVLink, and shrinks codesize by removing extraneous device/host transfers. While it still requires explicit data transfers, this CUDA-aware MPI is a much more appropriate comparison with Kokkos Remote Spaces.

```

1 if (myrank == 1) {
2     Send(B_device, 0);
3 }
4 if (myrank == 0) {
5     Receive(B_device, 1);
6 }
7 while(i < len(B_device)) {
8     A_device(i) += B_device(i);
9     i++;
10 }
```

**2.2. Device Initiated RMA.** Kokkos Remote Spaces implements a device-initiated call to `RMA_put` or `RMA_get` on any non-local data. This enables data in the PGAS to be accessed through a `Remote` view. See the below parallel pseudocode, which accomplishes the same stream task as above in less than half the lines of code.

```

1 while(i < len(B_device)) {
2     A_device(i) += B_device(i);
3     i++;
4 }
```

**2.3. Remote View Allocation and Implementation.** When a `Remote` view is allocated in Kokkos Remote Spaces, we pass a name for the view and a length to allocate. For a one-dimensional view, the parallel pseudocode might look like this:

```
1 A_device = RemoteView_t("Device::A", 10000);
```

Due to this call, Kokkos will allocate space equally across all devices at runtime. If, in the above example, there are four such processing elements, then each processing element receives one quarter of the data.

```

1 A_device(0); // an access which is local to processing element 0
2 A_device(2499); // local to processing element 0
3 A_device(2500); // local to processing element 1
4 A_device(9999); // local to processing element 3
```

Kokkos Remote Spaces calls the underlying API (in this case, NVSHMEM) to allocate this space at the time of allocation. In multidimensional arrays, the array is always split by its leading dimension. Of course, we remind the reader that Kokkos Remote Spaces is like Kokkos, and code written with the Kokkos Remote Spaces model that happens to use one underlying API can be recompiled to another API without changes to the source code.

Remote views are the only view type in Kokkos Remote Spaces that can offer the potential to access a view remotely - and it is done with syntax very similar accessing a `normal` view. A `normal` Kokkos view is unique to a single processing element, and an `unmanaged` Kokkos view (even if it contains the data from a `remote` view) does not check for the need to call RMA routines when accessed.

**3. Results.** We have performed a number of benchmarks, including OSU message rate, 1-process stream, 2-process stream, and (following the pattern of a recent study by NVIDIA[4]) 1-process stream with varying Kokkos teams size/league size. We selected these benchmarks due to their prevalence in the field of RMA and their ability to showcase the performance of Kokkos Remote Spaces. The specifications of these benchmarks (save for the one seeking to follow NVIDIA) can be found at <http://mvapich.cse.ohio-state.edu/benchmarks/>. In this proceedings article, we will show only a subset of these benchmarks, though we plan to present refined versions of the entire OSU suite for SHMEM in future work. We collected our data on *Weaver*, a machine on the Sandia Open Network, where each node contains four NVIDIA Volta100 GPUs. We used CUDA version 12.0.0 and its corresponding nvcc for our results.

All data was collected using a python script (<https://github.com/DSMishler/python-data-collector>) that ensures all data points have an error at most 1%.

**3.1. Teams.** Fig. 3.1 speaks to the low overhead of Kokkos Remote Spaces as a software layer and to the reproducibility of the data from the NVIDIA Corp developer team. This benchmark computes a stream of a large array on a single processing element while using the Kokkos Remote Spaces model with `remote` views. This benchmark was repeated for `normal` and `unmanaged` views with highly similar performance, such that it is not clear if any overhead between views is detectable. The implementation copies a large stream of data (128MB) within a single device. We see that we approach the theoretical maximum of 256GB/s, and this reproduces the results of NVIDIA's blog. For all other benchmarks listed in this paper, these values are left constant at their default values of 64 for league size and 256 for team size.

**3.2. One Process Stream.** Fig. 3.2 speaks to the low overhead of the `remote` view. Going through the additional check for locality on every access indeed is negligible compared to the speed of memory accesses. In this benchmark, we compare the updates per second for each permutation of views in the “`+=`” version and for each view type in the “`++`” version. The major deterministic factor in performance for this benchmark (though it has a minor effect) is the view type of the array to the left of the “`+=`”, most likely because it is accessed twice. Expectedly, there is a minor performance dip for each “`++`” and “`+=`” when they reach a size such that all the data can no longer fit in cache.

**3.3. Two Process Stream.** Fig. 3.3 speaks to the efficiency of RMA over different interconnects. The purpose of this graph is not to compare interconnects, but to compare NVSHMEM exposed through Kokkos Remote Spaces to CUDA-aware MPI over the same interconnect. A non-CUDA-aware MPI is also shown for completeness. Over NVLink and PCIe, Kokkos Remote Spaces has a clear advantage over MPI because it does not need to wait for the transfer to complete before beginning arithmetic. Over the Infiniband, however, Kokkos Remote Spaces pays can only send data element-wise at the time of writing this

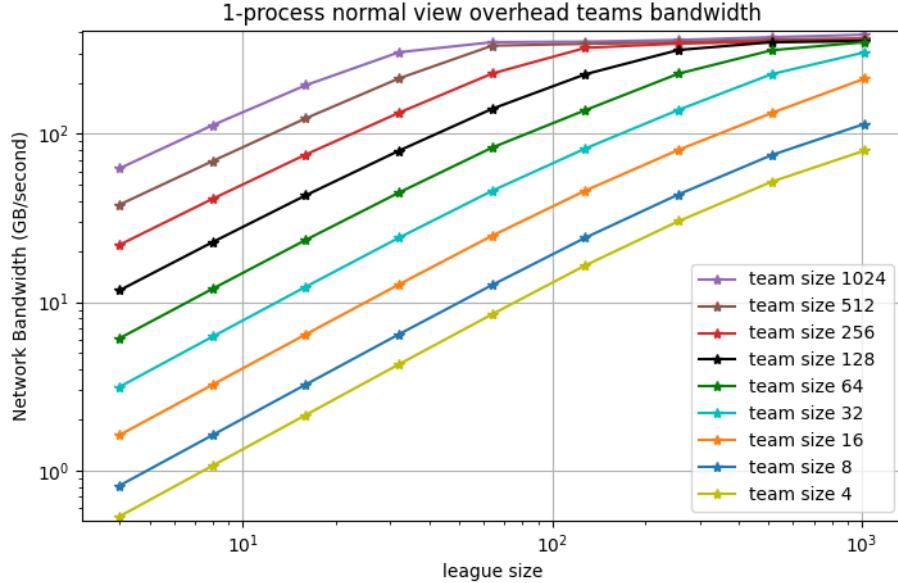


FIG. 3.1. This benchmark copies an array of data within the same process with varying team and league size and averages the result over 10 repetitions. Memory Bandwidth as estimated by a single process as a function of team size and league size. Kokkos Remote Spaces can, indeed, saturate the network bandwidth by increasing team size and league size. This benchmark exists to reproduce the graphs demonstrated by NVIDIA Corp in[4].

proceedings article. Thus, it pays a steep latency cost for each transfer. This can be seen in the alternative representation of the same results in Fig 3.4. Notice that, despite being slower as a whole over infiniband, Kokkos Remote Spaces offers a faster transfer of smaller arrays over infiniband because it does not incur the kernel launch latency.

#### 4. Discussion.

**4.1. Potential.** Fig. 3.3 shows a performance benefit from the use of RMA in Kokkos Remote Spaces due to reduced synchronization costs and natural interleaving of computation and communication thanks to the Kokkos Remote Spaces model.

Our results shown in Figs. 3.1, 3.2, and 3.3, along with the short codesize implementations (2-3x reduction in codesize), show that

1. Kokkos Remote Spaces as a software layer adds an undetectable overhead for the benchmarks shown.
2. Exposing SHMEM technology through Kokkos Remote Spaces has potential to reduce codesize in applications.
3. There exist some regimes and parameters that enable SHMEM to have a speedup over MPI.

**4.2. MPI and MPI One Sided.** After looking at our results, a shrewd reader might protest, citing MPI pipelining and one-sided MPI. This point is a valid one, and we acknowledge the existence of one-sided MPI and understand that well-optimized MPI code could feasibly match PGAS performance for all of our benchmarks. *But how easy was it to saturate network bandwidth using Kokkos Remote Spaces?* Without Kokkos Remote Spaces, Kokkos users must augment it with MPI for communication. Kokkos Remote Spaces can

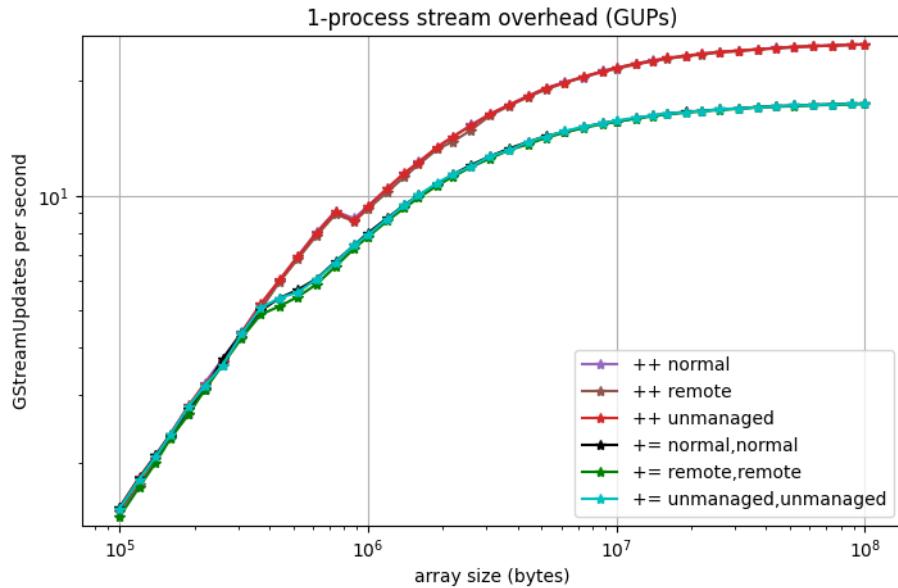


FIG. 3.2. This benchmark copies an array of data within the same process with the intention of measuring the overhead of using a remote view compared to a normal view and averages the result over 500 repetitions. The “++” labeled runs consisted of element-wise incrementing a single Kokkos View, while the “+=” run consisted of two separate views on the same process, requiring an extra load and slightly slowing performance. These “+=” runs were done with all 9 permutations of the three view types, all of which showed nearly indistinguishable performance. GUPs stands for “GigaUpdates per Second”, which was labeled “GStreamUpdates per second” here to ensure that GUPs is not conflated with the popular GUPs benchmark which does random updates. The observable performance dips occur when the amount of data outgrows the L2 cache.

act as a standalone programming model that saturates the network bandwidth with vastly fewer lines of code. We acknowledge that there are more MPI codes that don’t use Kokkos than Kokkos codes that don’t use MPI, but we still can add value to the environment of Kokkos with this API. If future work capitalizes on enabling near-optimal performance with the Kokkos model, then these SHMEM technologies could bring RMA’s benefits to a wide audience of HPC application developers.

**4.3. Traction.** Kokkos Remote Spaces also is behind MPI from the perspective of traction. While the topic has a great deal of interest as a programming model, the HPC community as a whole is relatively skeptical of the performance benefits of RMA. Due to the popularity of MPI and programming models that rely upon it, MPI will likely take up the attention from people interested in codesign and further optimizations of commonly used software. RMA technology has a long road ahead of it without this support - but this should only make any performance victories that come along in the application space all the more promising.

**4.4. Codesize.** In all of the above benchmarks, the codesize of the RMA version provided through Kokkos Remote Spaces was smallest by a large margin. For miniapplications, writing code with Kokkos Remote Spaces has reduced codesize by a factor of 2 or even 3. There is no need for buffers, MPI sends/receives, or memory allocation to specific processes. Serial code can easily be converted to parallel code using Kokkos Remote Spaces, and reduc-

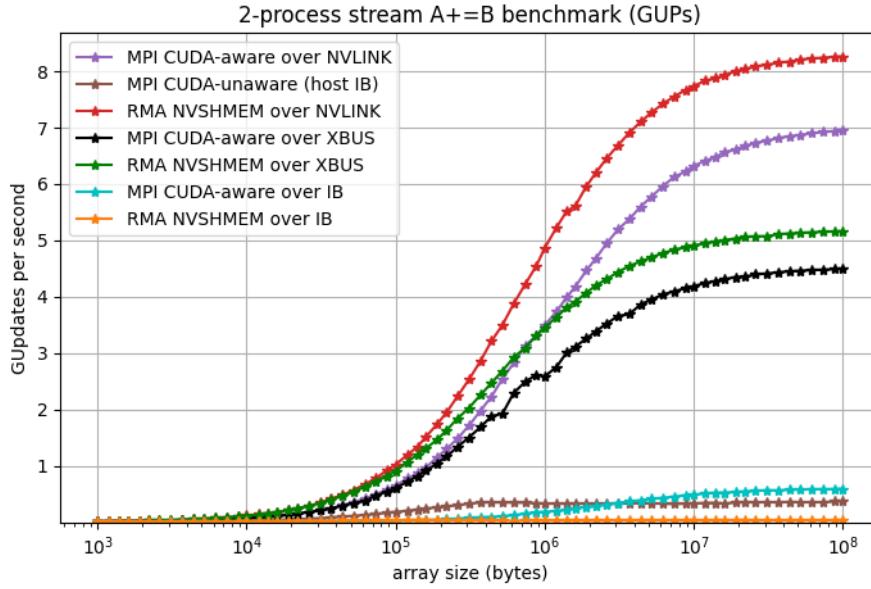


FIG. 3.3. Stream updates over different interconnects using Kokkos Remote Spaces with NVSHMEM RMA compared to MPI plotted as average speed over 500 repetitions. “NVLINK” is device to device communication over NVLink on the same node. “XBUS” is device to device communication over PCIe on the same node. “IB” is infiniband using two separate nodes. Note the linear y-axis. GUPs stands for “GigaUpdates per Second”.

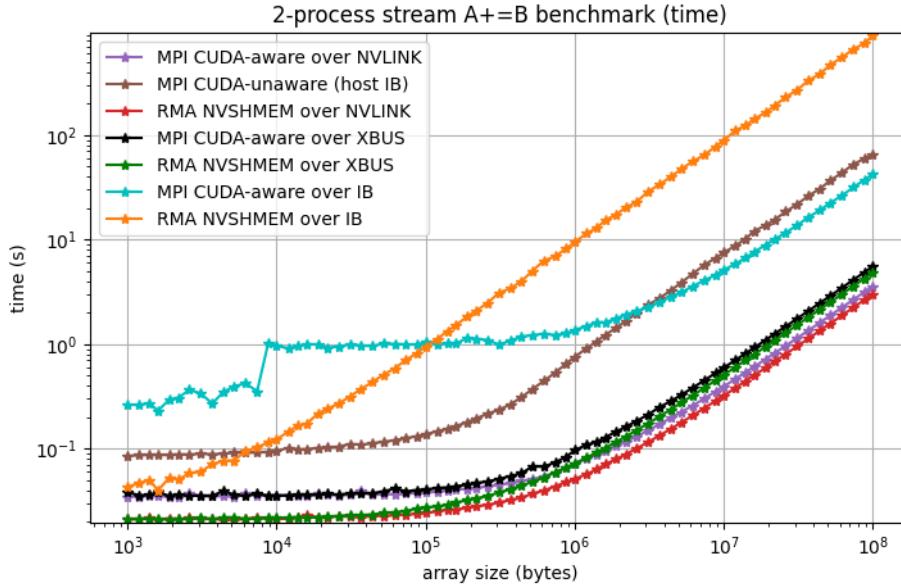


FIG. 3.4. Stream updates over different interconnects using Kokkos Remote Spaces with NVSHMEM RMA compared to MPI plotted as time over 500 repetitions. This is another representation of the data in Fig. 3.3, which more visibly exposes the differences in programming models over infiniband.

ing codesize will continue to be both an objective for the design of Kokkos Remote Spaces and one of its most appealing features.

## 5. Conclusion and Future Work.

**5.1. Future Work.** More work is still needed to evaluate the scalability limits of Kokkos Remote Spaces. With RMA, scalability concerns are not limited to software, but the hardware layers as well. Surely, with  $n$  processing elements, there will not be  $n^2$  fast interconnects. This raises the question about how Kokkos Remote Spaces will scale. Future work will investigate this further. Perhaps Kokkos Remote Spaces will slot into other kernels seeking to utilize cliques of highly local processing elements.

The current state of miniapplications is a little shaky thanks to a number of software blips and bugs - perhaps entirely unrelated to NVSHMEM or Kokkos Remote Spaces. Work is ongoing to investigate these performance results on a number of additional benchmarks and miniapplications. Future work will speak to the usefulness of Kokkos Remote Spaces more holistically, hopefully including specific applications which experience a speedup with this model.

As much as the Kokkos model would love to promise there will be no need for knowledge of the underlying hardware and parallelism concepts, it remains a tool. There are still pitfalls and common traps that Kokkos Remote Spaces, like Kokkos, will likely run into with its developers. Thus, future work will include careful thought about the usability of the API and how memory is managed.

Kokkos Remote Spaces is being tested with other NVIDIA GPUs and with GPUs from AMD, with plans to support any major vendor. Future work will include results from multiple GPUs

Future work will also speak about the unaddressed topic of synchronizations and their implementation and impact in Kokkos Remote Spaces, with MPI as a reference.

**5.2. Closing Remarks.** With the advent of GPU-centric interconnects and the availability of device-initiated RMA, libraries from vendors have motivated the development of the Kokkos Remote Spaces project. The promise of device-initiated communications using SHMEM could enable improvements in productivity and performance in parallel programming. This work is currently being submitted as a poster and extended abstract to IEEE Cluster 2023, with plans to write more papers later. The hope of this work is that it will incite these discussions at Sandia, IEEE Cluster, and beyond.

**Acknowledgment.** We would like to thank the community of 1400 for their support and willingness to help, especially noting helpful guidance from Chris Siefert and Siva Rajamanickam.

## REFERENCES

- [1] B. CHAPMAN, T. CURTIS, S. POPHALE, S. POOLE, J. KUEHN, C. KOELBEL, AND L. SMITH, *Introducing openshmem: Shmem for the pgas community*, Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, (2010).
- [2] J. CIESKO, *Kokkos remote spaces? public preview 3*. <https://www.osti.gov/servlets/purl/1897600>, 2021. SAND2021-14299PE.
- [3] T. HOEFLER, J. DINAN, R. THAKUR, B. BARRETT, P. BALAJI, W. GROPP, AND K. UNDERWOOD, *Remote memory access programming in mpi-3*, ACM Transactions on Parallel Computing, (2015).
- [4] P. MARKTHUB, J. DINAN, S. POTLURI, AND S. HOWELL, *Improving network performance of hpc systems using nvidia magnum io nvshmem and gpudirect async*. <https://developer.nvidia.com/blog/improving-network-performance-of-hpc-systems-using-nvidia-magnum-io-nvshmem-and-gpudirect-async>. Accessed: 2023-08-07.

- [5] C. R. Trott, D. LeBrun-Grandi, 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.

## OFFLOADING JOB-LOCAL FILESYSTEMS IN HIGH-PERFORMANCE COMPUTING ENVIRONMENTS

JOHN SHAWGER\* AND MATTHEW L. CURRY†

**Abstract.** Job-local parallel filesystems are becoming a crucial part of modern high-performance computing systems. Despite their advantages in processing IO compared to traditional parallel filesystems (PFS), the performance impact of such filesystems on compute nodes in an HPC system has not been well studied. Using HPL, a highly coordinated matrix factoring benchmark from the Linpack suite, and IOR, a standard IO benchmark, we characterize the impact of filesystem daemons on compute tasks. Finally, we study the performance of NVMe-over-Fabrics and propose offloading filesystem operations to SmartNICs.

**1. Introduction.** HPC clusters traditionally share a large parallel filesystem cluster amongst all nodes in the cluster. Filesystem activity from one job running on the HPC cluster may negatively affect the IO performance of another job running on the same cluster, significantly increasing performance variability. Job-local storage [30] running on SATA or NVMe SSD devices on each compute node promises to improve IO performance by positioning storage closer to the computation, and improve job isolation by reducing the impact of per-job IO on the broader cluster.

Recent HPC systems have used node- and job-local storage for the benefits listed above. The Summit [11] system, operational in 2018 at Oak Ridge National Laboratory, was designed with a two-tiered storage system [26] – a traditional PFS as well as node-local burst buffers. The burst buffer system was designed for checkpoint-restore, and applications could use several different interfaces to interact with the system. Burst-Buffer API (BB-API) supports N-to-N checkpointing patterns with asynchronous draining of data to the PFS, while Burst Buffer Shared Checkpoint File System (BSCFS) is a log-structured file system built on FUSE which supports N-to-1 checkpointing. Applications could also use Scalable Checkpoint Restart (SCR) [5] to drain data from the SSDs to the PFS. All three of these solutions require modifications to application code, making each relatively difficult to use. Furthermore, BSCFS is not POSIX-compliant, raising other hurdles to application adoption. Data movement to the PFS is done via NVMe-over-Fabrics. While this reduces compute-node CPU burden, node memory and network access can still be affected, and are controlled via QoS and throttling.

Two other systems were developed for Summit which allowed more transparent access of the burst buffer storage layer: Spectral and SymphonyFS [11]. Spectral supports N-N checkpointing to the burst buffer with no application code changes. Applications write directly to XFS on the local SSD. When the application closes the file, Spectral detects the `close()` system call and moves the data to the PFS without application involvement. SymphonyFS is another FUSE-based solution for N-1 checkpointing. It aggregates local SSDs into a single namespace and decomposes metadata and data operations. Metadata operations are passed through to the PFS. Data operations are cached locally on XFS and then drained to the PFS later. While SymphonyFS appears to provide a single namespace and is transparent to application developers, it assumes nodes work on non-overlapping regions of a file, and thus has a much different consistency model than POSIX, which limits its scope to certain checkpointing workloads.

The Fugaku supercomputer in Kobe, Japan at the Riken Center for Computational Science also uses two-tiered storage, and manages the fast NVMe layer with the Lightweight Layered IO Accelerator (FFIO) [1] system. For every group of 16 compute nodes, Fugaku

---

\*University of Wisconsin-Madison, shawgerj@cs.wisc.edu

†Sandia National Laboratories, mcurry@sandia.gov

has a node equipped with NVMe storage which is designated as a storage and compute node. This node handles all higher-tier storage requests for other nodes in its group. FFIO is used for three purposes – a cache for the second-tier PFS, an area for temporary files shared between compute nodes running in a single job, and an area for temporary files exclusive to compute nodes. The temporary areas are not intended for files that will be sent to the second-tier PFS.

Existing research on job-local filesystems does not study the performance impact of running filesystem daemons on compute nodes. Tightly-coupled parallel workloads can suffer from performance degradation and variability when system noise interferes with the parallel workload [7]. A classic case of noise interference is several ranks of an MPI process entering a synchronization barrier. If one rank is delayed due to a context switch, all other ranks must wait to exit the barrier. We show that the effect of a node-local filesystem daemon on parallel application performance is significant, variable, and disproportionate. In other words, application performance is affected by a larger margin than the CPU usage of the filesystem daemon, and the application’s runtime has a greater level of uncertainty.

Each of these solutions trades filesystem consistency or compute for acceptable performance. No system achieves transparent application access to the burst-buffer layer, POSIX-compliance, and low compute overhead. SmartNIC devices are well-positioned to help achieve all three goals. They are becoming increasingly common in HPC systems and data centers, they have their own CPU cores and memory, and they are capable of running general-purpose Linux distributions. We investigate the utility of using SmartNICs to offload node-local filesystems, and how offload could impact compute performance.

## 2. Background.

**2.1. Job-local Filesystems.** Job-local filesystems in HPC systems aggregate storage resources local to compute nodes in the HPC cluster under a single namespace. There are a number of advantages to colocating storage and compute including IO performance and task isolation, or reducing the noisy neighbor problem. Many node-local filesystems are also *ephemeral*. That is, they only exist for the duration of an associated job, workflow, or application run.

Existing job-local filesystems are commonly used as burst buffers. Burst buffers are designed to absorb higher rates of IO than traditional parallel filesystems. After being written to the burst buffer, data may be copied asynchronously to the parallel filesystem.

To improve performance, many recent job-local filesystems relax POSIX semantics, assuming, mostly correctly [28], that HPC applications tend to manage their IO carefully, and the application can be trusted, for example, to not write concurrently to the same file offset. Illustrative examples from the recent literature include UnifyFS [4], which uses commit consistency semantics, where writes are not globally visible until they have been committed, and GekkoFS [27], which provides eventually-consistent metadata handling by avoiding global locks during metadata operations.

Providing weaker consistency guarantees may improve filesystem performance, but could also limit these filesystems for use by existing well-behaved HPC applications. We chose to focus our efforts on BeeOND, an ephemeral job-local filesystem based on BeeGFS [10], which supports full POSIX semantics.

**2.1.1. BeeOND.** BeeOND is an ephemeral filesystem based on BeeGFS, a parallel filesystem designed for use in high performance systems. There are three main components in the BeeGFS filesystem – a *storage service*, a *metadata service*, and a *client service*. BeeGFS decouples data and metadata similarly to other distributed filesystems like Ceph [29] and Lustre [24]. The storage service stores striped BeeGFS data on standard Linux filesystems

such as ext4 or XFS. Typically the storage nodes are in a RAID-6 configuration for fault tolerance. The metadata service controls the striping pattern and data placement. The client service mounts the BeeGFS filesystem on client nodes. Applications may interact with the filesystem as they do with any mounted filesystem. BeeGFS supports full POSIX semantics, so no modification is necessary for applications to use BeeGFS. When accessing data, clients first contact the metadata service, then the storage service directly to access the data. BeeGFS services may be run individually on separate nodes, or services may be colocated on single nodes.

BeeOND is an ephemeral filesystem based on BeeGFS. In our setup, BeeOND startup was integrated with Slurm so that new Slurm allocations had their own private BeeOND instance, which ran for the duration of the allocation. The startup process automatically assigns the first node to host the metadata service, then distributes storage services across all nodes. It is also possible for the metadata service to be distributed over several nodes, in which case each node handles a segment of the filesystem namespace.

**2.2. SmartNICs.** As data center networks continue to grow faster and compute growth slows due to the ending of Moore’s Law, SmartNICs have gained popularity in data centers as a platform for network acceleration and host offload. SmartNICs typically have several hardware accelerators for common network operations – for example, compression, erasure coding, and TCP offload. They also have their own (generally low-power) cores, memory, and persistent flash memory. SmartNICs are PCIe devices connected to a host node. They can operate on the network path, where they manipulate packets coming in or out of the host, or in off-path mode, where they act as discrete hosts [15].

We use Mellanox Bluefield-2 SmartNICs [17] in separated-host mode. The Bluefield-2 has eight ARM cores, 8-16GB memory (16GB in our configuration), and persistent flash memory. It has a ConnectX-6 HCA capable of NVMe-oF offload, and several other offload accelerators which are accessible through the NVIDIA DOCA SDK, or special RDMA verbs supported by the Mellanox drivers. Running in separated-host mode, the Bluefield-2 can run general-purpose Linux distributions, Ubuntu in our case. It can be accessed from the host via `ssh`, and may also communicate with the host via RDMA over PCIe.

**2.3. NVMe-over-Fabrics.** NVMe-over-Fabrics (NVMe-oF) is a protocol for accessing NVMe devices over a network, commonly RDMA via either Infiniband or RoCE. The shift from SATA SSDs to NVMe SSDs has caused an evolution in the Linux kernel IO stack. Prior to Linux 3.13, with the introduction of `blk-mq` [3], a single-threaded IO stack had acceptable performance due to the request latency of spinning disk hard drives, which were predominant. The parallelism of NVMe SSDs requires a different architecture for optimal usage of the disk. Data is written to NVMe devices using paired submission-completion queues. There may be up to 64K queues, and each queue may have up to 64K entries. Queues are typically mapped to cores on the host CPU. Queue entries are scheduled using either round robin or a weighted round robin strategy.

NVMe thus maps naturally to the send, receive, and communication queue pairs in RDMA. When connecting an NVMe-oF initiator to a target, the user may specify the number of queues in the connection. Each queue is assigned its own core by the operating system, on both the initiator and the target. We were able to verify this behavior during our experiments.

An I/O request takes the following path in a filesystem mounted on an NVMe-oF block device. We consider a `write` request on the ext4 filesystem. First, the virtual filesystem (VFS) sends the request to ext4. The ext4 filesystem then creates a block I/O request (`bio`). The `bio` is sent to the `nvme` driver, which is presenting a block device to the initiator system. The block device can either be backed by a physical NVMe device, or an NVMe-oF

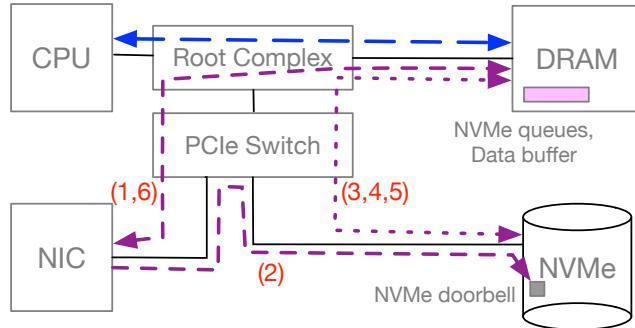


FIG. 2.1. NVMe-over-Fabrics NIC offload with memory accesses

connection to the target. In this case, we assume an NVMe-oF connection via RDMA, so the `bio` is routed to the RDMA subsystem.

Once on the target, the `nvmet` (NVMe-target) driver processes the request by sending the encapsulated `bio` to kernel block layer. Once finished, the target sends an acknowledgment to the initiator, which receives it as a software interrupt.

**2.3.1. NVMe-oF NIC Offload.** Recent Mellanox ConnectX HCAs (5 and higher) support NVMe-oF target offload [21]. We found that the standard Linux `nvmet` driver for RHEL8 did not include offload support, so we used the MLNX\_OFED `nvmet` driver released by Mellanox. When target offload is enabled, the network interface is able to handle the NVMe request received over RDMA directly, without going through the kernel block layer. This reduces CPU usage on the target to nearly zero.

In more detail, these are the steps for an offloaded NVMe `write` operation (Figure 2.1):

1. The offload engine generates a submission queue entry and does a DMA write to the submission queue for NVMe in main memory.
2. The offload engine does a P2P write, ringing the “doorbell” of NVMe device.
3. The NVMe device responds to the doorbell and fetches the submission queue entry from memory via DMA.
4. The NVMe device writes data from the memory buffer and stores it on disk.
5. The NVMe device writes to the completion queue to notify that the operation has completed.
6. The offload engine polls the completion queue to detect when the write operation is done.

Although NVMe-oF offload reduces CPU usage on the target, it still uses memory bandwidth, a source of possible contention [18]. Some NVMe drives have a controller memory buffer (CMB) [19], a region of DRAM accessible via a PCIe base address register, which may be used to store the submission and completion queues. In that case, reading and writing to the queues is done via P2P DMA, and operations can be offloaded without accessing system memory. Our NVMe disks did not have CMBs, so we did not explore this functionality.

**3. Design.** We evaluate the performance impact of BeeOND on compute node performance. Given  $3N$  nodes participating in BeeOND, we run a compute-heavy workload on  $2N$  nodes requiring communication between all ranks, and an IO-heavy workload on  $N$  nodes. Since all nodes are running BeeOND, IO is distributed across all  $3N$  nodes. Figure 3.1 shows our experimental design and the placement of BeeOND services for  $N = 2$  (there are two nodes running the IO workload and four nodes running the compute workload). HPL, IOR, and the BeeOND storage and metadata services run in user space, while the BeeOND

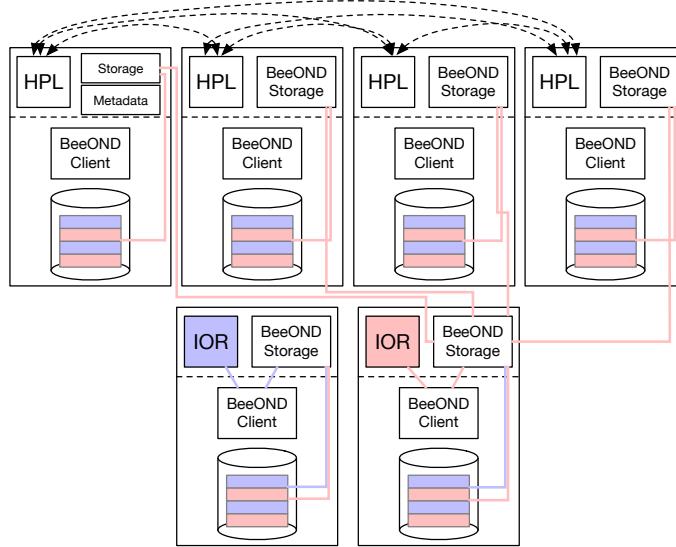


FIG. 3.1. *HPC allocation with BeeOND, HPL, and IOR. Blue storage lines omitted for clarity.*

client is a kernel filesystem module.

**3.1. Compute Workload.** We use the High Performance Linpack (HPL) benchmark [22] to provide a compute workload that is reflective of HPC applications. HPL performs matrix factorization in a tightly coupled MPI job. It is compute intensive. Our hypothesis before running the experiment was that HPL would experience slowdowns due to its tightly coupled nature. When a group of MPI ranks enters a barrier, they may only leave the barrier once the last rank has entered. This makes barrier performance degrade as performance variation increases.

After studying HPL more closely, we found that it uses a more complicated communication scheme than a simple barrier. HPL has six different broadcast algorithms which are configurable at runtime. On every iteration of the LU factorization, ranks must broadcast across their row and column. In the default “increasing-ring” topology, the broadcasting rank shifts to the next rank in the ring on each iteration [16]. Although this is not exactly the communication scheme we expected, it exhibits similar performance degradation to the barrier model due to inter-rank communication dependencies, as we show in the evaluation.

**3.2. IO Workload.** We use IOR to generate IO-intensive workloads for our experiments. IOR is a configurable benchmark suitable for evaluating parallel filesystems. IOR has a POSIX backend and uses MPI for rank synchronization which makes it particularly suitable for HPC systems.

```
srub -N$iornum -n$((cores*iornum)) -w ${ior_nodes} ${IOR_PATH}/ior \
-t 128 -T 20 -D 60 -i $((1024*1024)) -e -C -w -a POSIX -s 1024 -F -Y \
-o=/mnt/beeond/testfile
```

FIG. 3.2. *IOR Runtime*

Our complete IOR runtime is given in Figure 3.2. We discuss the most important flag choices here. We perform 128 byte writes (-t 128) with an `fsync` after each write (-Y). Each process has its own file (-F) to avoid contention on the file level. We repeat the test

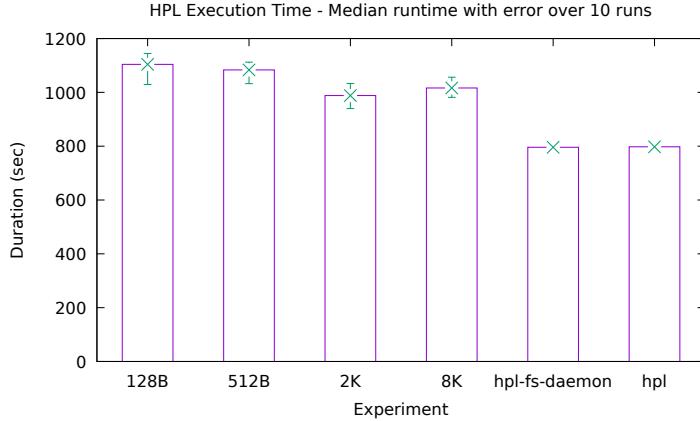


FIG. 4.1. *HPL runs with and without corunning IOR + BeeOND*

(-i) continuous for 20 minutes, or until the completion of HPL. This ensures that we are performing IO for the duration of the experiment.

The small sequential writes and frequent `fsync` calls are designed to generate significant IO activity and exercise BeeOND as much as possible. We experimented with 512B, 2K, and 8K writes and found similar results with all four sizes. Future work could investigate other IO patterns, such as large sequential writes, more fully.

#### 4. Experimental Results.

**4.1. Impact of BeeOND on HPL runtime.** Corunning the HPL compute workload with BeeOND daemons servicing IOR write activity has a substantial impact on compute performance. Using the setup described in Figure 3.1, Figure 4.1 shows the duration of our HPL experiment, averaged over 10 runs with errors bars showing the minimum and maximum, for several different configurations. The column furthest to the left, `hpl`, has no corunning filesystem daemon. The second column from the left, `hpl-fs-daemon`, shows the runtime with BeeOND daemons running, but no IO workload. The other four columns show run times with IOR configured for the sizes of writes labeled on the x-axis.

Notably, HPL runs with very low variance when it is the sole tenant on each node. When BeeOND is servicing an IO workload, HPL runtime increases and becomes more variable. We measured the CPU usage of BeeOND separately, and found that per node, it uses a total of between 3-4 cores of CPU time, out of 56 cores on each node. If the slowdown in HPL runtime was proportional to the CPU usage of BeeOND, we would expect HPL runtime to be about 7% longer. Instead, we find that the median runtime of HPL with BeeOND and IOR is 38% slower, and it is 44% slower on the worst run. Performance improves as write sizes increase because the number of IO operations decreases. However, even with 8K writes, HPL runtime is significantly affected.

**4.2. Impact of BeeOND on HPL communication.** Our hypothesis was that MPI ranks performing panel broadcast in HPL were being delayed by BeeOND, which led to increased execution time. We verified this by instrumenting the HPL code to time when ranks sent broadcast messages. Our results are shown in Figure 4.2. (a) shows rank broadcasts for HPL running alone, and (b) shows rank broadcasts for HPL corunning with BeeOND.

Figure 4.2 plots, for the duration of one HPL run, the difference in time between consecutive messages sent during the broadcast phase for each of the first 16 ranks. Since

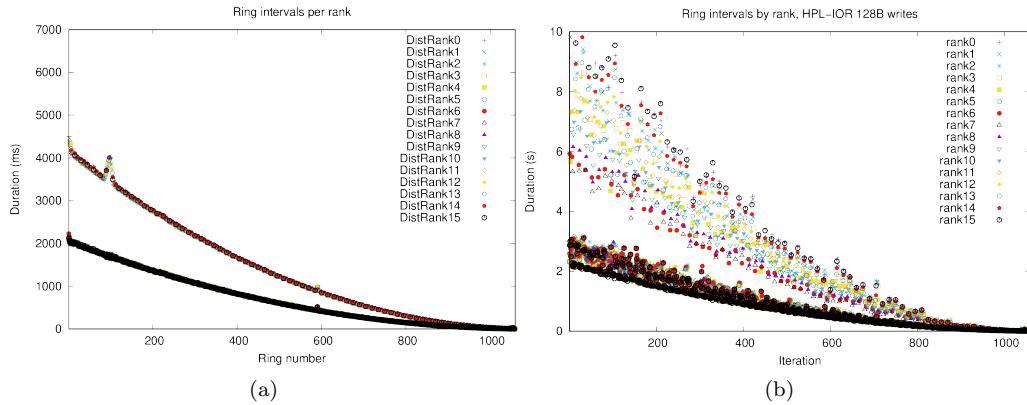


FIG. 4.2. *HPL time between messages sent, first 16 ranks. (a) shows HPL running alone. (b) shows HPL co-running with BeeOND services and IO workload.*

all durations are found by subtracting timestamps from the same rank, we avoid worrying about clock drift between ranks.

The results show that HPL broadcasts have a distinctive communication pattern which falls into two modes – a shorter time between messages sent, and a longer time. When BeeOND is running on the node with an IO-intensive workload, the pattern is still visible, but there is a large increase in communication time variation. Since computation cannot continue until ranks have broadcasted their panels, an increase in variation leads to a decreased runtime.

**4.3. BeeOND SmartNIC Offload.** Thus far, we have shown that co-running BeeOND on nodes in an allocation has a measurable performance impact on compute tasks. We now investigate the feasibility of offloading BeeOND to SmartNICs attached to each node.

The key mechanism we chose to make offload possible is to access host storage using NVMe-over-Fabrics (NVMe-oF). The SmartNIC can transparently access an NVMe device on the host using NVMe-oF offload which is available on modern ConnectX HCAs. Using HCA offload reduces host CPU usage to near zero while the SmartNIC is accessing storage. Alternatively, NVMe-oF can be used with non-NVMe SSDs or RAM disks, but HCA offload is not available. We investigated the following questions using two test setups, described in figures 4.1 and 4.2.

1. Can the SmartNIC access host storage using NVMe-oF?
2. What is the overhead of accessing an NVMe device over fabrics, compared to accessing it locally?
3. Do lower-powered cores on the SmartNIC lead to a performance penalty compared to a full host when accessing storage via NVMe-oF?

We used the Cloudlab setup primarily because it offered more administrative flexibility, although it also let us compare the performance of a SmartNIC relative to another full node. We use terminology similar to iSCSI for NVMe-oF – the “target” hosts the NVMe disk being accessed, while the “initiator” is the system accessing the disk remotely. The relationship is the same whether working with two full nodes, or a SmartNIC and a full node.

The SmartNIC on Singra is configured in separated host mode. Thus, the SmartNIC has its own IP address, and can communicate with the target via RDMA. In this way, the setup for both systems is nearly identical.

Cloudlab sm110p - Two nodes
16-core Intel Xeon Silver 4314
128GB ECC Memory
1TB SSD
4x1TB Samsung 980PRO NVMe SSD
100Gbps network
Dual-port Mellanox ConnectX-6 Dx, Dual-port Mellanox ConnectX-6 Lx

TABLE 4.1  
*Cloudlab test setup*

Singra node with BF-2
32-core AMD EPYC 7513
512GB ECC Memory
2x1TB Samsung 980PRO NVMe SSD
NVIDIA Bluefield-2 SmartNIC via PCIe x8

TABLE 4.2  
*Singra test setup*

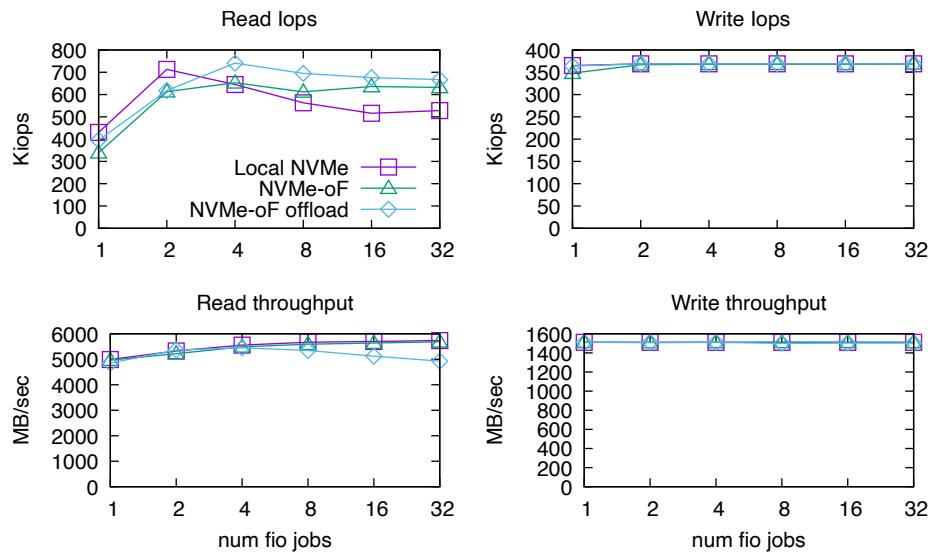


FIG. 4.3. *NVMe-oF FIO Performance Measurements on Cloudlab between two hosts*

**4.3.1. NVMe performance over Fabrics.** First, we determine the performance of NVMe-oF compared to accessing an NVMe device locally. We use FIO for disk microbenchmarks – latency, IOPS, and throughput. We also use LevelDB as an application-level benchmark to measure expected real-world performance of the system.

Figure 4.3 shows the performance of NVMe-oF using the two-node Cloudlab setup (Table 4.1). Results for NVMe-oF with and without target offload are compared to local NVMe performance. NVMe-oF is competitive with accessing an NVMe device locally for throughput and IOPS workloads. In fact, NVMe-oF outperforms local disk access in read IOPS. This was attributed to interrupt coalescing in a previous study of NVMe-oF [9], although the authors did not evaluate the interrupts generated by either the NVMe device or the network interface. We found (Figure 4.4) that fewer interrupts were generated on the NVMe-oF target per operation compared to an equivalent workload with local NVMe access.

NVMe-oF has higher latency than local device access due to network overheads. Interestingly, offloading NVMe-oF reduces latency compared to traditional NVMe-oF. This is because NVMe-oF bypasses the kernel block stack on the target. We give a latency CDF in Figure 4.5. One other benefit we would expect to see in NVMe-oF offload

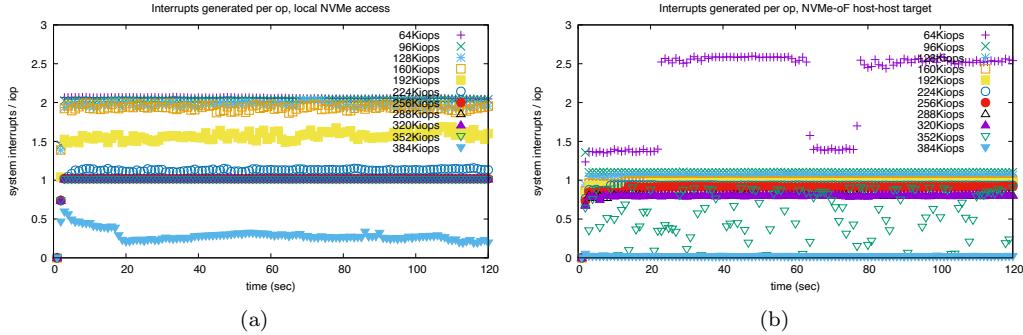


FIG. 4.4. All system interrupts generated per write operation. Measurements collected in two second intervals. (a) Local NVMe access; (b) NVMe-oF target

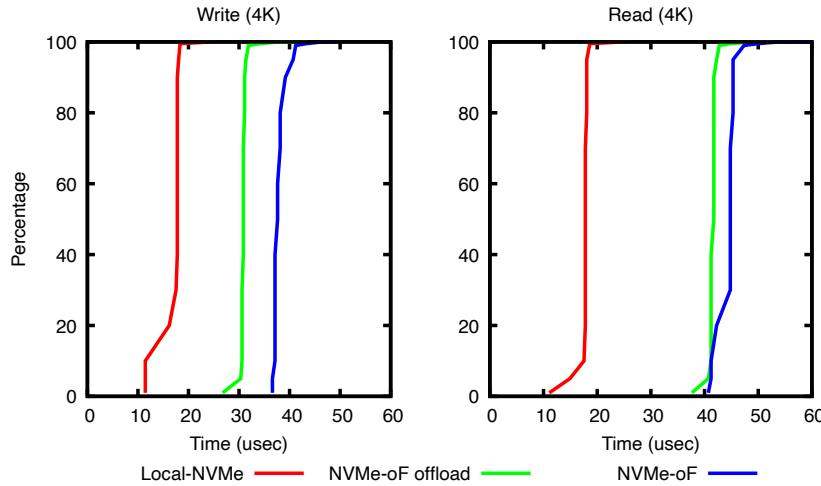


FIG. 4.5. Operation latency CDF – Local NVMe and NVMe-oF

is improved tail latency when the target is under load. We do see improved tail latency for NVMe-oF offload in Figure 4.5, but this measurement was conducted on an unloaded system. NVMe-oF tail latency under load has been studied in a comparison with iSCSI [9], but not with NIC offload.

We also confirm that CPU usage is reduced while using offload. Figure 4.6 shows that system interrupts on the target, as measured by `vmstat`, are at idle levels for all evaluated workloads when offload is enabled.

**4.3.2. NVMe-oF on SmartNIC.** Next, we evaluate NVMe-oF performance accessing host storage from a Bluefield-2 SmartNIC. These experiments used the Singra setup (Table 4.2). The host-host measurements were done on Cloudlab (Table 4.1) and were used to control for the slower cores of the Bluefield-2. The SmartNIC has a harder time generating large amount of IO operations. In the write IOPS benchmark, it needs four FIO jobs to generate the same amount of operations as a single FIO job on a Cloudlab node. It is, however, capable of sustaining the same amount of write IOPS.

We find that the SmartNIC is unable to keep up with a Cloudlab host in read IOPS. Our

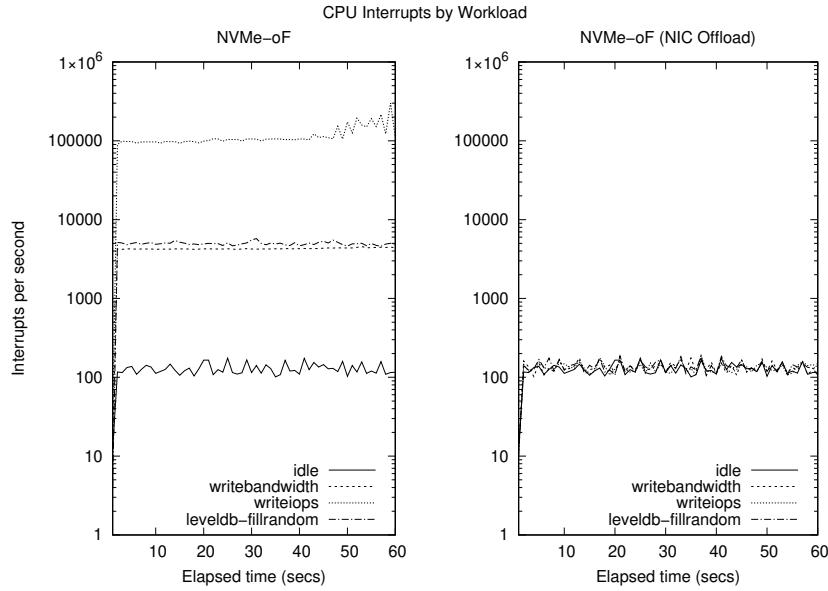


FIG. 4.6. NVMe-oF Offload Target Interrupts

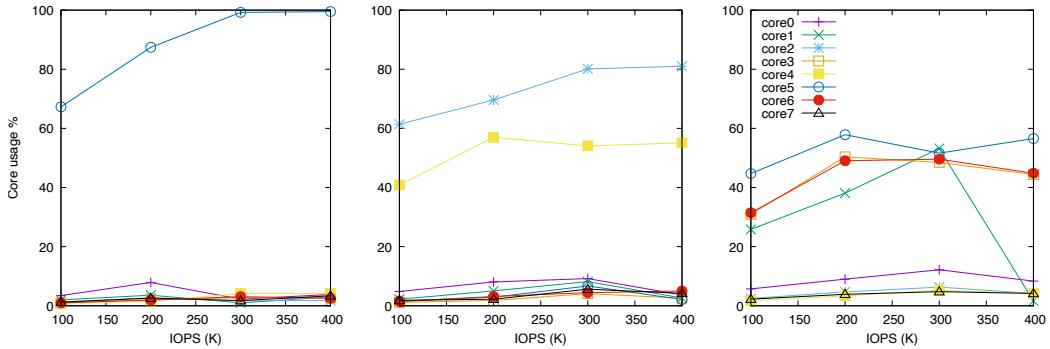


FIG. 4.7. NVMe-oF Bluefield-2 CPU Usage for cores processing software interrupts.

first thought was that using a single NVMe-oF queue saturated a core on the SmartNIC, and limited the number of read IOPS the system was capable of. Experiments with increased numbers of queues showed that this assumption was correct and that core usage went down when the system was configured to use two queues (Figure 4.3.2). This increased the number of read IOPS the system could support, but increasing the number of queues beyond two had limited effect, and the system still had decreased performance compared to a Cloudlab host. Future work may help determine another bottleneck.

Throughput measurements show that the SmartNIC can access the device with at least the same throughput, and better throughput in the case of read throughput, compared to accessing the device with another node. Latency measurements, comparing offloaded requests from a SmartNIC as well as a separate host, show that the SmartNIC has a latency advantage compared to other hosts (Figure 4.8), and operation latency is generally within 10 microseconds of local NVMe performance.

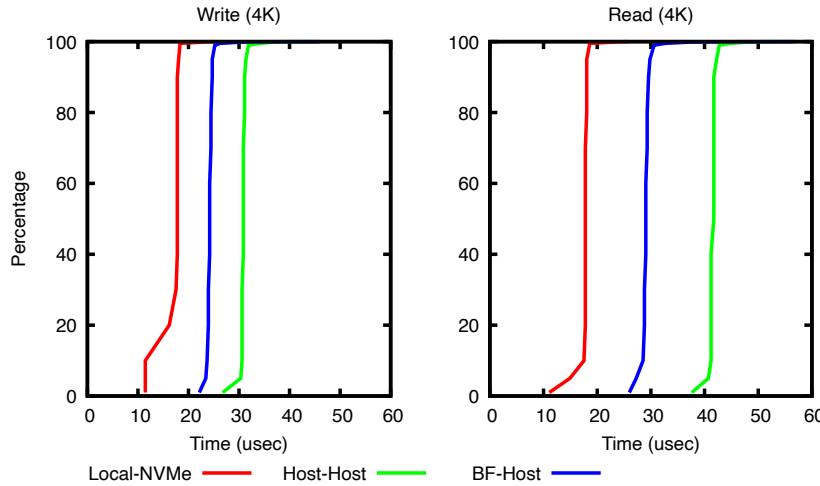


FIG. 4.8. *Operation latency CDF – NVMe-oF offload host-to-host and SmartNIC-to-host*

## 5. Related Work.

**5.1. Scheduling.** One classic approach to reducing performance variability in distributed systems is to co-schedule dependent processes [20, 2, 6]. While more cooperative scheduling may improve compute performance in HPC systems, we find that BeeOND consumes between three and four cores. Therefore, even an optimal scheduling strategy will consume more host resources than desired, and it is beneficial to offload the filesystem.

**5.2. SmartNIC Offload.** SmartNIC systems are a growing research field. SmartNICs are available in a wide variety of configurations, including FPGA devices and systems with general-purpose CPUs, as we have evaluated here. They have been used for diverse purposes, including acceleration of network operations [8, 14], distributed transactions [12, 23], and distributed computation [15].

**5.2.1. LineFS.** LineFS [13] is a SmartNIC-offloaded distributed filesystem which seeks to minimize CPU usage of distributed filesystems in multi-tenant environments. Although LineFS shares our goal of reducing node CPU usage, they have both different goals and different mechanisms. LineFS decomposes distributed filesystems into operations which can be offloaded and completed asynchronously. This reduces CPU usage in multi-tenant systems. HPC systems are not typically multi-tenant, but corunning a filesystem on a node can lead to increased performance variability. LineFS accesses byte-addressable persistent storage on the host via RDMA from the SmartNIC. Our approach uses NVMe-oF, and is not limited to systems with persistent memory.

**6. Conclusion.** We have shown that compute workloads can suffer performance penalties if there is an IO-intensive workload running on a node-local filesystem. These performance penalties are disproportionate to the amount of CPU time used by the filesystem, which makes offload to low-power cores on SmartNICs an attractive solution.

The client-daemon architecture of BeeOND is suitable for offload. BeeOND daemons may run on SmartNICs, which can access host storage transparently using NVMe-oF. BeeOND clients may run without modification on hosts. In our experiments, accessing host storage from the SmartNIC with NVMe-oF proved to have excellent performance,

matching or exceeding local NVMe performance in read/write throughput and write operations per second, and incurring a relatively small latency penalty. More investigation is needed to determine the bottleneck for small random read operations, but this workload is less important for traditional HPC applications. NVMe-oF NIC offload, available in modern ConnectX HCAs, reduces interrupt generation and CPU usage on the host to idle levels. We expect, therefore, that offloading BeeOND filesystem daemons to SmartNICs will substantially lower CPU usage on host nodes, and maintain good filesystem performance. This approach retains POSIX-compliance, making the filesystem usable for workloads other than traditional checkpoint-restore, filling a gap in the current ecosystem of HPC filesystems.

## REFERENCES

- [1] H. AKIMOTO, T. OKAMOTO, T. KAGAMI, K. SEKI, K. SAKAI, H. IMADE, M. SHINOHARA, AND S. SUMIMOTO, *File system and power management enhanced for supercomputer Fugaku*, Fujitsu Technical Review, (2020), pp. 2020–03.
- [2] R. H. ARPACI, A. C. DUSSEAU, A. VAHDAT, L. T. LIU, T. E. ANDERSON, AND D. A. PATTERSON, *The interaction of parallel and sequential workloads on a network of workstations*, in Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, Ottawa, Canada, May 15–19, 1995, B. D. Gaither, ed., ACM, 1995, pp. 267–278.
- [3] M. BJØRLING, J. AXBOE, D. NELLANS, AND P. BONNET, *Linux block io: introducing multi-queue SSD access on multi-core systems*, in Proceedings of the 6th international systems and storage conference, 2013, pp. 1–10.
- [4] M. J. BRIM, A. T. MOODY, S.-H. LIM, R. MILLER, S. BOEHM, C. STANAVIGE, K. M. MOHROR, AND S. ORAL, *UnifyFS: A user-level shared file system for unified access to distributed local storage*, in 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2023, pp. 290–300.
- [5] J. CAO, K. ARYA, R. GARG, S. MATOTT, D. K. PANDA, H. SUBRAMONI, J. VIENNE, AND G. COOPERMAN, *System-level scalable checkpoint-restart for petascale computing*, in 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2016, pp. 932–941.
- [6] A. C. DUSSEAU, R. H. ARPACI, AND D. E. CULLER, *Effective distributed scheduling of parallel workloads*, in Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, Philadelphia, Pennsylvania, USA, May 23–26, 1996, D. A. Reed and B. D. Gaither, eds., ACM, 1996, pp. 25–36.
- [7] K. B. FERREIRA, P. G. BRIDGES, AND R. BRIGHTWELL, *Characterizing application sensitivity to OS interference using kernel-level noise injection*, in Proceedings of the ACM/IEEE Conference on High Performance Computing, SC 2008, November 15–21, 2008, Austin, Texas, USA, IEEE/ACM, 2008, p. 19.
- [8] D. FIRESTONE, A. PUTNAM, S. MUNDKUR, D. CHIOU, A. DABAGH, M. ANDREWARTHA, H. ANGEPAT, V. BHANU, A. M. CAULFIELD, E. S. CHUNG, H. K. CHANDRAPPA, S. CHATURMOHTA, M. HUMPHREY, J. LAVIER, N. LAM, F. LIU, K. OVTCHAROV, J. PADHYE, G. POPURI, S. RAINDEL, T. SAPRE, M. SHAW, G. SILVA, M. SIVAKUMAR, N. SRIVASTAVA, A. VERMA, Q. ZUHAIR, D. BANSAL, D. BURGER, K. VAID, D. A. MALTZ, AND A. G. GREENBERG, *Azure accelerated networking: Smartnics in the public cloud*, in 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9–11, 2018, S. Banerjee and S. Seshan, eds., USENIX Association, 2018, pp. 51–66.
- [9] Z. GUZ, H. LI, A. SHAYESTEH, AND V. BALAKRISHNAN, *NVMe-over-fabrics performance characterization and the path to low-overhead flash disaggregation*, in Proceedings of the 10th ACM International Systems and Storage Conference, 2017, pp. 1–9.
- [10] J. HEICHLER, *An introduction to BeeGFS*. [https://www.beegfs.io/docs/whitepapers/Introduction\\_to\\_BeeGFS\\_by\\_ThinkParQ.pdf](https://www.beegfs.io/docs/whitepapers/Introduction_to_BeeGFS_by_ThinkParQ.pdf), 2014. Accessed: 2023-10-17.
- [11] J. HINES, *Stepping up to Summit*, Computing in science & engineering, 20 (2018), pp. 78–82.
- [12] D. KIM, A. S. MEMARIPOUR, A. BADAM, Y. ZHU, H. H. LIU, J. PADHYE, S. RAINDEL, S. SWANSON, V. SEKAR, AND S. SESAN, *Hyperloop: group-based nic-offloading to accelerate replicated transactions in multi-tenant storage systems*, in Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20–25, 2018, S. Gorinsky and J. Tapolcai, eds., ACM, 2018, pp. 297–312.
- [13] J. KIM, I. JANG, W. REDA, J. IM, M. CANINI, D. KOSTIC, Y. KWON, S. PETER, AND E. WITCHEL, *LineFS: Efficient smartNIC offload of a distributed file system with pipeline parallelism*, in van Renesse and Zeldovich [25], pp. 756–771.

- [14] B. LI, K. TAN, L. L. LUO, Y. PENG, R. LUO, N. XU, Y. XIONG, AND P. CHENG, *Clicknp: Highly flexible and high-performance network processing with reconfigurable hardware*, in Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016, M. P. Barcellos, J. Crowcroft, A. Vahdat, and S. Katti, eds., ACM, 2016, pp. 1–14.
- [15] M. LIU, T. CUI, H. SCHUH, A. KRISHNAMURTHY, S. PETER, AND K. GUPTA, *Offloading distributed applications onto smartnics using ipipe*, in Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19–23, 2019, J. Wu and W. Hall, eds., ACM, 2019, pp. 318–333.
- [16] H. LU, P. SAO, M. MATHESON, R. KANNAN, F. WANG, AND T. POTOK, *Optimizing communication in 2d grid-based mpi applications at exascale*, in Proceedings of the 30th European MPI Users' Group Meeting, 2023, pp. 1–11.
- [17] *Mellanox bluefield data processing unit*. <https://www.nvidia.com/en-us/networking/products/data-processing-unit/>. Accessed: 2023-08-24.
- [18] R. NAKAMURA, Y. KUGA, AND K. AKASHI, *How beneficial is peer-to-peer DMA?*, in Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems, 2020, pp. 25–32.
- [19] *Nvm express revision 1.2.1*. [https://nvmeexpress.org/wp-content/uploads/NVM\\_Express\\_1\\_2\\_1\\_Gold\\_20160603.pdf](https://nvmeexpress.org/wp-content/uploads/NVM_Express_1_2_1_Gold_20160603.pdf). Accessed: 2023-08-22.
- [20] J. K. OUSTERHOUT, *Scheduling techniques for concurrent systems*, in Proceedings of the 3rd International Conference on Distributed Computing Systems, Miami/Ft. Lauderdale, Florida, USA, October 18-22, 1982, IEEE Computer Society, 1982, pp. 22–30.
- [21] T. OVED, *Nvme over fabrics offload*, in OpenFabrics Alliance Workshop, 2019. Accessed: 2023-08-22.
- [22] A. PETITET, R. C. WHALEY, J. DONGARRA, AND A. CLEARY, *Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers*. <https://www.netlib.org/benchmark/hpl/>. Accessed: 2023-08-21.
- [23] H. N. SCHUH, W. LIANG, M. LIU, J. NELSON, AND A. KRISHNAMURTHY, *Xenic: Smartnic-accelerated distributed transactions*, in van Renesse and Zeldovich [25], pp. 740–755.
- [24] P. SCHWAN ET AL., *Lustre: Building a file system for 1000-node clusters*, in Proceedings of the 2003 Linux symposium, vol. 2003, 2003, pp. 380–386.
- [25] R. VAN RENESSE AND N. ZELOVICH, eds., *SOSP '21: ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual Event / Koblenz, Germany, October 26–29, 2021*, ACM, 2021.
- [26] S. S. VAZHUDAI, B. R. DE SUPINSKI, A. S. BLAND, A. GEIST, J. SEXTON, J. KAHLE, C. J. ZIMMER, S. ATCHLEY, S. ORAL, D. E. MAXWELL, ET AL., *The design, deployment, and evaluation of the CORAL pre-exascale systems*, in SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2018, pp. 661–672.
- [27] M.-A. VEF, N. MOTI, T. SÜSS, T. TOCCI, R. NOU, A. MIRANDA, T. CORTES, AND A. BRINKMANN, *Gekkofs-a temporary distributed file system for hpc applications*, in 2018 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, 2018, pp. 319–324.
- [28] C. WANG, K. MOHROR, AND M. SNIR, *File system semantics requirements of HPC applications*, in Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, 2021, pp. 19–30.
- [29] S. A. WEIL, S. A. BRANDT, E. L. MILLER, D. D. LONG, AND C. MALTZAHN, *Ceph: A scalable, high-performance distributed file system*, in Proceedings of the 7th symposium on Operating systems design and implementation, 2006, pp. 307–320.
- [30] H. ZHENG, V. VISHWANATH, Q. KOZIOL, H. TANG, J. RAVI, J. MAINZER, AND S. BYNA, *HDF5 cache VOL: efficient and scalable parallel I/O through caching data on node-local storage*, in 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022, Taormina, Italy, May 16–19, 2022, IEEE, 2022, pp. 61–70.

## A SIMPLE STATISTICAL MODEL FOR RANDOMIZED BENCHMARKING DATA

SAMYAK SURTI\* AND TIMOTHY PROCTOR †

**Abstract.** Randomized Benchmarking (RB) techniques are often used to probe the capabilities of quantum processors. RB consists of running random circuits with varied circuit depths; the mathematical theory underpinning RB shows that the mean success probability of RB circuits decay exponentially in the depth of the circuit. However, RB theory makes no claims about the distribution of success probabilities for RB circuits at a given depth. This makes statistically rigorous analysis of RB data difficult; as a result, RB data is typically analyzed with ad hoc curve-fitting. In our work, we solve this problem with a simple statistical model for RB data. We model RB success probabilities with a Beta distribution with mean that decays exponentially in circuit depth and variance that is modelled by a few-parameter ansatz function. This few-parameter statistical model enables maximum likelihood estimation of RB error rates with rigorous confidence intervals. Our analysis is easily adaptable to a plethora of RB techniques—including standard/Clifford-group RB, direct RB, binary RB, and (a slight adaptation of) mirror RB. Consequently, we demonstrate how it can enable ‘super-efficient’ RB of many-qubit processors.

**1. Introduction.** The advent of quantum information processors (QIPs) in the hundreds-of-qubits regime seemingly grasps at the next foothold on the ascent to universal, fault-tolerant quantum computers. However, boasting high qubit counts is meaningless without methods to understand the noise affecting a QIP. A myriad of quantum characterization, verification, and validation (QCVV) techniques [1] that probe the exact and average noisy behaviors of a quantum device exist, ranging in their granularity, from gate-set tomography (GST) [9]—which determines the per-gate error channel process matrix for a processor’s native gate-set—to randomized benchmarking (RB) [13]—which estimates the average per-layer error rate of a quantum processor. However, unlike other QCVV techniques, RB lacks statistical rigor. The theory underlying RB only defines the *mean* of per-depth RB success probabilities, but provides no information on the *variance*. As a result, ad hoc curve-fitting methods are employed, but there’s room for further progress.

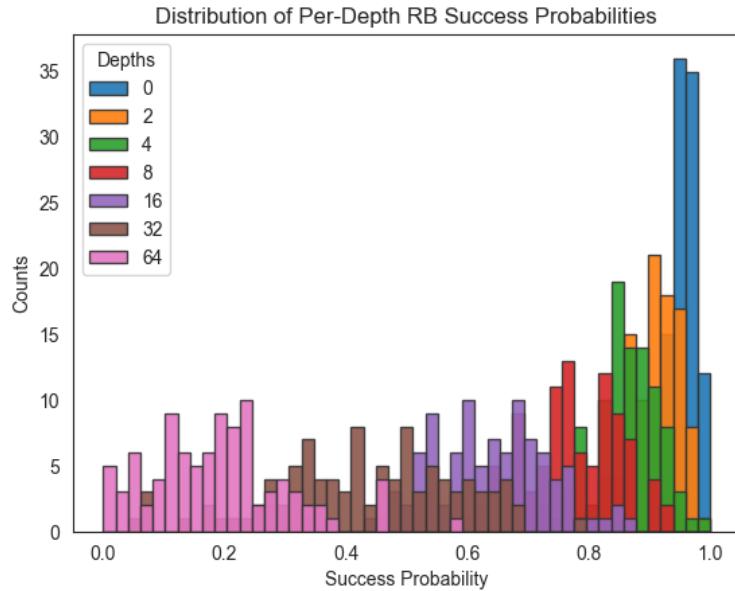
In this work, we address this shortcoming by developing a statistically rigorous model to accurately fit RB success probabilities, motivated by the observed Beta distributed-nature of these success probabilities on a per-depth basis. With this model, we now contextualize RB success probability data as being sampled from a Beta distribution and RB success count data as being sampled from a Beta-Binomial distribution (See Figure 1.1), as opposed to simply being output data from an RB experiment. To fit collected RB data to our model, we use maximum likelihood estimation (MLE). As a proof-of-concept of the validity of this novel statistical fitting method, we utilize Mirror Randomized Benchmarking (MRB) [15]. Additionally, we present an evolution of MRB, refactored in terms of Direct Fidelity Estimation (DFE) [2, 3], (dubbed MRB + DFE) so as to express RB data in terms of success and fail counts rather than Hamming distance-based adjusted success probabilities. We seek to generalize our findings to a broader scope of RB techniques.

The rest of the paper is structured as follows: Section 2 introduces preliminary concepts; we motivate RB as a protocol for understanding average error rates of QIPs, how RB experiments are structured and designed, and a brief discussion of the soundness of RB theory. We also touch on DFE, which leads into an explication of the MRB + DFE protocol. Section 3 provides intuition for the statistically rigorous model for MRB success probabilities and empirical evidence supporting the “goodness” of this model. Finally, Section 4 will discuss the two methods to fit MRB success probabilities with MLE; one method will be

---

\*University of California, Berkeley, spsurti@berkeley.edu

†Sandia National Laboratories, tjproct@sandia.gov



**Fig. 1.1: Roughly Beta-Distributed Nature of RB Success Probabilities** On a per-depth basis, we observe that RB success probabilities are roughly beta-distributed. This data is collected from simulating MRB experiments in pyGSTi on 5 qubits for circuit depths 0 through 64, exponentially-spaced.

based on a statistical model dependent only on circuit depth (number of gate layers in a quantum circuit) while the other method will be based on a statistical model dependent on both circuit width (the number of qubits being benchmarked) and the circuit depth.

## 2. Preliminaries.

**2.1. Randomized Benchmarking. Motivation.** Methods determining the probability of success with which a QIP can run quantum circuits is useful for characterizing the capabilities and limitations of quantum computers. Randomized Benchmarking (RB) [7, 8, 14] is one such protocol, by which the average error per logical gate for a given QIP can be measured by running randomly-sampled circuits with efficiently computable outputs for a range of circuit widths and depths. This average error rate is the single figure of merit for RB. For the purposes of this work, we will focus on RB for non-universal gate sets (namely the Clifford group), however there is active effort towards developing scalable RB methods for universal gate sets as well [4].

*RB Experimental Design.* A standard RB experiment can be distilled into 3 main steps [14]: (1) Based on the QIP's specification,  $K$  random circuits are sampled at each width  $w$  and depth  $d$  on a per-layer basis from a user-defined distribution  $\Omega(L)$ , for  $L$  some gate layer set, for a range of circuit widths and depths. These circuits are constructed so as to ideally implement the identity operation, thus their outputs are efficiently computable. (2) Run each random circuit on the QIP for  $n \geq 1$  shots and record the number of successes. (3) Calculate the observed per-depth success probabilities  $\bar{S}_d$  (estimating the true per-depth success probabilities  $S_d$ ) and fit these probabilities to an exponential decay curve (See Figure 2.1) of the form  $\bar{S}_d = Ap^d + B$  (where the parameter  $B$  varies depending on the RB

technique employed). The parameter  $A$ , a  $d$ -independent quantity, represents the effect of state preparation and measurement (SPAM) errors on  $\bar{S}_d$ ;  $p$  represents an estimate of the true average decay rate of circuit success probabilities;  $B$  represents the asymptote of the success probabilities as  $d \rightarrow \infty$ . To define the RB figure of merit ( $r$ ), we rescale  $p$  in the following way:  $r = (d - 1)(1 - p)/d$  for  $d = 2^n$  or  $d = 4^n$  if we are concerned with entanglement fidelity or average gate fidelity, respectively.

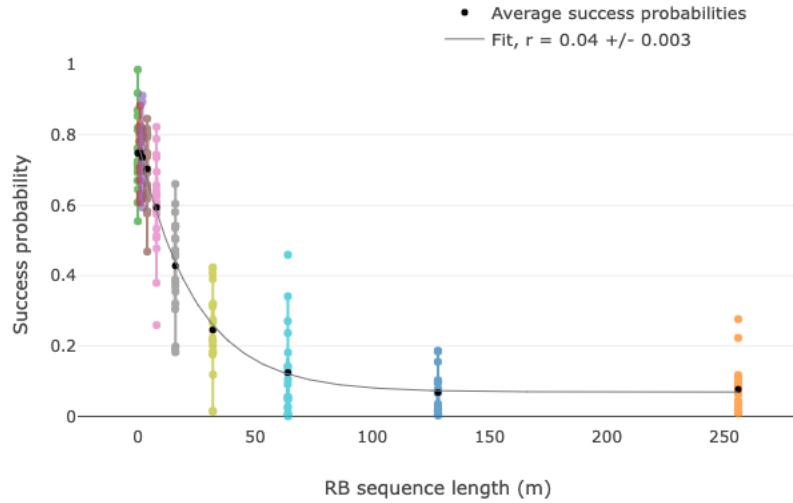


Fig. 2.1: **Exponential Decay of DRB Success Probabilities.** Simulated DRB success probability data collected and plotted using PyGSTi for a processor with 4 qubits connected in a ring. Based on PyGSTi’s ad-hoc fitting to an exponential decay curve of the form  $\bar{S}_d = Ap^d + (1/2)^w$ , we get that the RB number  $r = 0.04 \pm 0.003$ .

**RB Theory.** The primary goal of the theoretical foundations of RB is to prove that (1) per-depth success probabilities do follow an exponential decay curve—that is  $\bar{S}_d \approx Ap^d$ —and (2) the RB error rate  $r$  is a good estimate for the true fidelity decay rate  $\epsilon$ . For an in-depth explanation for why (1) and (2) hold for various RB protocols, the reader can refer to [7, 14, 15].

**2.1.1. Mirror Randomized Benchmarking.** Mirror Randomized Benchmarking (MRB) [15] is a scalable and robust RB technique whose fundamental unit is the randomized mirror circuit (See Figure 2.2). As the name suggests, for some even circuit depth  $d$ , these circuits consist of  $d/2$  gate layers randomly sampled from a user-defined gate layer distribution  $\Omega(L)$ , as before. The inverses of these  $d/2$  gate layers are then applied in reverse order—essentially mirroring the first  $d/2$  gate layers—so as to ideally implement the identity, thus fulfilling the requirement that RB circuits must have an efficiently computable output. The very first layer of these circuits is comprised of randomly-sampled single-qubit Clifford gates and the final gate layer is the inverse of this initial layer.

The above description of randomized mirror circuits is naive as its structure does nothing to prevent errors from being echoed out. Thus, to address this issue, between each of the  $d$  randomly-sampled gate layers, a randomly-sampled Pauli layer is inserted. Another effect of

these Pauli layers is that they twirl error channels on each of the  $d$  gate layers into stochastic Pauli error channels.

Furthermore, when enumerating output data results, effective polarizations ( $S$ ; see Eq 2.1) are utilized, as they more accurately estimate the true fidelity decay of MRB circuits compared to simply counting successes based on matching target and output bit strings. Theoretical details regarding the proof of this fact can be found in [15].

$$P_{\text{adj}} = \sum_{k=0}^n \left( -\frac{1}{2} \right)^k h_k \quad \mid \quad S = \frac{4^n}{4^n - 1} P_{\text{adj}} - \frac{1}{4^n - 1} \quad (2.1)$$

**2.2. Direct Fidelity Estimation.** *Motivation.* An important QCVV task is to assess the quality of preparing some desired quantum state  $\rho$  or the application of some desired unitary  $U$ ; thus, if the actual prepared quantum state is  $\hat{\rho}$  and the actual unitary applied is  $\hat{U}$ , we require some way to measure how close  $\hat{\rho}$  and  $\hat{U}$  are to  $\rho$  and  $U$ , respectively. To characterize the fidelity with which quantum states are prepared and quantum processes are applied, the QCVV techniques of quantum state tomography [6] and quantum process tomography [12] were developed. These techniques are, however, a double-edged sword: although they provide complete information regarding our quantum state or process, they do so in a highly resource intensive manner, preventing scalability of these techniques. For the purposes of this work, we are interested solely in the fidelity with which quantum processes are applied, rather than a complete mathematical description of an applied quantum process. As such, Direct Fidelity Estimation [2, 3] is a technique that efficiently distills an estimate of the fidelity of a quantum process, independent of the system size.

*Brief Description of Process DFE.* At its core, process DFE is a technique to estimate the entanglement fidelity [10] of a quantum channel with a constant number of Pauli expectation measurements:

$$\mathcal{U}[\rho] = U\rho U^\dagger \quad (2.2)$$

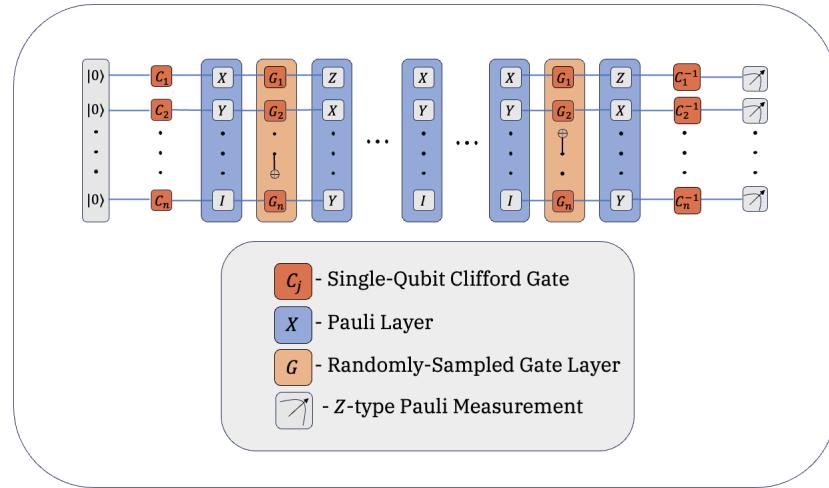
$$\mathcal{E} = \mathcal{U}^{-1} \circ \hat{\mathcal{U}} \quad \mid \quad \overline{F}(\hat{\mathcal{U}}, \mathcal{U}) = \overline{F}(\mathcal{E}, \mathbb{1}) = \frac{1}{4^n} \text{Tr}(\mathcal{E}) = \frac{1}{4^n} \sum_{P \in \mathcal{P}_n} \text{Tr}(P\mathcal{E}[P]) \quad (2.3)$$

*Application of DFE to RB.* Provided that DFE is giving us a simple, efficient way to estimate the fidelity of a quantum process, we can generalize this idea to using DFE to estimate the fidelity of RB circuits. Fundamentally, we can construe the evolution of a quantum state in an RB circuit as the evolution of a Pauli operator, whereby the underlying quantum state is the +1-eigenstate of the evolving Pauli. We then perform measurements with our final evolved Pauli operator (which is necessarily prescribed to be a  $Z$ -type Pauli—a Pauli operator comprised of tensor products of identities and  $Z$ -type Pauli operators) to ensure that we are in a +1-eigenstate of this Pauli; averaging over many RB circuits at a given depth  $d$ , we are able to characterize the average fidelity with which quantum circuits of depth  $d$  are run. Across a range of circuit depths, this technique gives us a new way to estimate the fidelity decay rate as a part of the RB protocol. As a proof-of-concept, in the following sub-section we discuss what an evolution of RB with DFE looks like in the context of MRB.

**2.2.1. Mirror Randomized Benchmarking with Direct Fidelity Estimation.** We describe a method to estimate the average fidelity decay rate by merging DFE with

MRB. As stated above, rather than thinking about which quantum state we're in as gates are applied, we are more interested with which Pauli we're a  $+1$ -eigenstate of as our system evolves.

Recall the structure of randomized mirror circuits (Refer to Figure 2.2). For these circuits, the evolution of the Pauli proceeds as follows: 1) In the all-zeros state, we are in a  $+1$ -eigenstate of some  $Z$ -type Pauli operator  $P_{\text{init}}$ . 2) Upon application of the initial uniformly randomly sampled single-qubit Clifford layer, each qubit is evolved to a  $+1$ -eigenstate of  $\pm X, Y$ , and  $Z$ . However, we'd like our state to be a  $+1$ -eigenstate of some uniformly random Pauli. We know every qubit is necessarily in a  $+1$ -eigenstate of the identity operator. Thus, each qubit is labeled by the identity operator with probability  $1/4$ . This post-processing steps ensures that each qubit is in a uniformly-random  $+1$ -eigenstate of  $I, \pm X, \pm Y$ , and  $\pm Z$ . This is equivalent to evolving to a uniformly randomly sampled Pauli operator  $P = C^\dagger P_{\text{init}} C$  for some choice of  $P_{\text{init}}$ . 3) The "core" circuit  $U$ , which ideally implements the identity operator up to a Pauli, evolves  $P$  to  $P' = U^\dagger P U$ . 4) Finally, we apply the inverse of the initial single-qubit Clifford layer  $C$  to evolve to our final Pauli  $P_{\text{out}} = CP'C^\dagger$ . 5) We ensure that  $P_{\text{out}}$  is a  $Z$ -type Pauli (we can apply  $X$ -type Paulis as necessary) and then perform a computational basis measurement to ensure that we are indeed in a  $+1$ -eigenstate of our evolved Pauli. If we are, we report a success, else we report a fail. Therefore, rather than dealing with adjusted success probabilities as in Eq 2.1, we can work with success counts as in other RB techniques [7, 11]. We compare the accuracies of standard MRB with MRB+DFE in Figure 2.3.



**Fig. 2.2: Randomized Mirror Circuit.** The structure of a randomized mirror circuit for MRB is as follows: 1) Begin with a layer of single-qubit Clifford operators to prepare the initial state. 2) Apply the "core" circuit consisting of  $d/2$  randomly-sampled gate layers, followed by the mirrored inverse of these  $d/2$  gate layers. Interspersed between each gate layer is a randomly-sampled  $n$ -qubit Pauli operator. The "core" circuit ideally implements the identity operator up to some uniformly-random Pauli. 3) Apply the inverse of the initial single-qubit Clifford layer. 4) Measurement layer

**2.3. Simulated Data.** Simulation data was collected by running MRB experiments using pyGSTi for circuit widths (number of qubits benchmarked) 1, 2, 3, 4, 5, and 6 and

circuit depths (number of circuit layers) 0, 2, 4, 8, 16, 32, 64, 128, 256, 512. We assume that the simulated quantum processors have qubits connected in a ring. We randomly sampled 10 Hamiltonian and stochastic Pauli noise models per-width (`pyGSTi` will also add affine and correlated Pauli error generators as necessary to ensure that the generated noise model corresponds to a valid CPTP map) to simulate noisy quantum device behavior.

**2.4. IBM-Q Data.** Experimental data was collected by running MRB experiments on the IBM Quantum Montreal system, a 27-qubit device. These experiments were run by benchmarking 1 qubit, 2 qubits, and so on until all 27 qubits were benchmarked. The order in which qubits were benchmarked was determined by the numbers assigned to each qubit by IBM.

**3. Statistical Model for Fitting Randomized Benchmarking Success Probabilities.** The crux of this work is the development of a statistically rigorous model to describe the behavior of per-depth RB success probabilities. Currently, to fit observed RB success probability data, a standard curve-fitting technique is employed to fit to the exponential decay curve  $Ap^d + B$  (where  $B$  is determined by the RB technique being considered). Although RB theory tells us that fidelity decays exponentially as a function of circuit depth—thus validating the ad-hoc curve fitting approach—our statistically-inspired approach serves to not only provide a structured understanding of the behavior of RB success probabilities, but allow for an accurate characterization of a QIP’s noisy behavior with smaller datasets. Thus, what we’re proposing is a model based on the observation that underlying per-depth RB success probabilities are roughly Beta-distributed, and thus per-depth RB success counts are Beta-Binomially distributed.

**3.1. Modeling RB Circuits Success Counts with the Beta-Binomial Distribution.** *Intuition.* Before discussing the applicability of the Beta-Binomial distribution to RB, it is helpful to understand the relation of the Beta distribution to RB. Fundamentally, the Beta distribution is useful for modeling the distributions of proportions and probabilities [5]. Given that we’re trying to imbue some kind of statistical structure onto RB success probabilities, it is reasonable to consider a Beta distribution for modeling these probabilities. Mathematically, the probability density function of the Beta distribution is defined as follows:

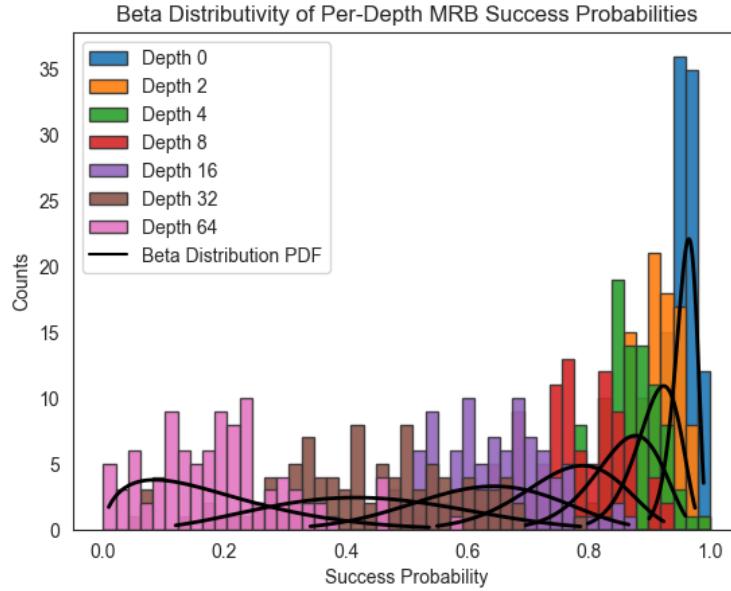
$$f(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \quad \Big| \quad B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \quad (3.1)$$

where  $B(\alpha, \beta)$  is the beta function and  $\Gamma(\alpha)$  is the gamma function. Here, the parameters  $\alpha$  and  $\beta$  can be defined in terms of the mean ( $\mu$ ) and variance ( $\sigma^2$ ) of the distribution, and vice versa. We omit derivations and present the definitions of  $\alpha$  and  $\beta$  in terms of  $\mu$  and  $\sigma^2$ .

$$k = \frac{1-\mu}{\mu} \quad \Big| \quad \alpha = \frac{k - \sigma^2(k+1)^2}{\sigma^2(k+1)^3} \quad \Big| \quad \beta = k\alpha \quad (3.2)$$

To see that the Beta distribution is suitable for modeling RB success probabilities, we refer the reader to Figure 3.1. Based on this plot, it is clear that the modeling Beta distribution has a dependence on benchmarked circuit depth.

*Generalization to Beta-Binomial Distribution.* Having shown that the Beta distribution is reasonably well-suited to model the behavior of RB successes probabilities, we discuss modeling RB success counts.



**Fig. 3.1: Suitability of Modeling MRB Success Probabilities with Beta Distribution.** By collecting the per-depth success probability data, calculating the mean and variance of these observed distributions, and plotting the related Beta distribution based on this mean and variance, we see that the behavior of the observed data can be closely modeled by the Beta distribution.

Consider an arbitrary RB circuit at circuit width  $w$  and circuit depth  $d$ . Assume we have learned the parameters of the Beta distribution  $B$  to appropriately model the success probabilities for this ensemble of circuits. If we take  $N$  shots of this circuit, it is natural to model the number of successful circuit runs with a Binomial distribution. Given that we have learned the underlying success probability distribution to be  $B$ , we let  $B$  define the probability of success for the binomial distribution. This defines a Beta-Binomial distribution modeling the number of successful RB circuit shots. Mathematically, the probability mass function of the Beta-Binomial distribution is defined as follows:

$$f(x | n, \alpha, \beta) = \binom{n}{x} \frac{B(x + \alpha, n - x + \beta)}{B(\alpha, \beta)} \quad (3.3)$$

Notice the dependence on the Beta function, but more precisely, the dependence on the parameters  $\alpha$  and  $\beta$  used to define the Beta distribution. This fact becomes crucial in the following section.

**3.2. Modeling the Mean and Variance of MRB Success Probability Distributions.** In our above toy scenario, we skipped past an integral component of modeling RB success counts: learning the Beta distribution to model the success probabilities of circuits at a given width  $w$  and depth  $d$ . Recall from Eq. 3.1 that a Beta distribution is defined exactly by parameters  $\alpha$  and  $\beta$  and from Eq. 3.2 that  $\alpha$  and  $\beta$  are defined exactly by the mean and variance of this distribution. Thus, in order to learn the form of a Beta distribution for circuits at a particular width  $w$  and depth  $d$ , we require coming up with suitable

few-parameter functional forms for the mean and variance of this distribution

### 3.2.1. Parameterized Function to Model the Mean of RB Success Probability Distributions.

To accurately model the mean of RB success probabilities, we turn to RB theory which tells us that  $\bar{S}_d = Ap^d + B$ , where again  $B$  is determined based on the specific RB technique being employed.

### 3.2.2. Ansatz Function to Model the Variance of the RB Success Probability Distributions.

Unlike the mean, the variance of RB success probabilities does not have a clean, functional form (a generalized analytical form can be derived through representation theory). In this scenario, it becomes necessary to define a variance ansatz function whose form is determined by the observed behavior of the variance of RB success probability distribution.

Intuitively, we can consider three different depth-regimes in order to characterize the variance. At low-depth, we expect almost all circuits to run successfully, thus we will have a low variance in our success probabilities. At medium-depth, we are running an exponentially-larger number of circuits with varied gate layers that will result in a higher variance in RB success probabilities. Finally, at high-depth, errors accumulate exponentially in the depth of the circuits, so we expect that success probabilities are clustered around 0, therefore resulting in a lower variance.

Let us now describe the design of the variance ansatz function. To connect these three depth-regimes, we need a growth term to transition between the low-depth regime and the medium-depth regime and a decay term to transition between the medium-depth regime and the high-depth regime; both growth and decay terms will naturally have a dependence on circuit depth. Finally, we must ensure that our variance function is properly bounded. Given that the Beta distribution is defined on  $[0, 1]$ , it follows that its variance is bounded from above by  $\mu(1 - \mu)$  where  $\mu$  is the mean of the distribution. A short proof of this fact is provided below:

*Statement:* For a probability distribution defined on  $[0, 1]$  the variance of such a distribution is upper-bounded by  $\mu(1 - \mu)$

*Proof.* For distributions with support on  $[0, 1]$ ,

$$\int_0^1 x^2 f_X(x) dx \leq \int_0^1 x f_X(x) dx$$

as  $x^2 \leq x$  on  $[0, 1]$ . Therefore,

$$Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \leq \mathbb{E}[X] - \mathbb{E}[X]^2 = \mathbb{E}[X](1 - \mathbb{E}[X])$$

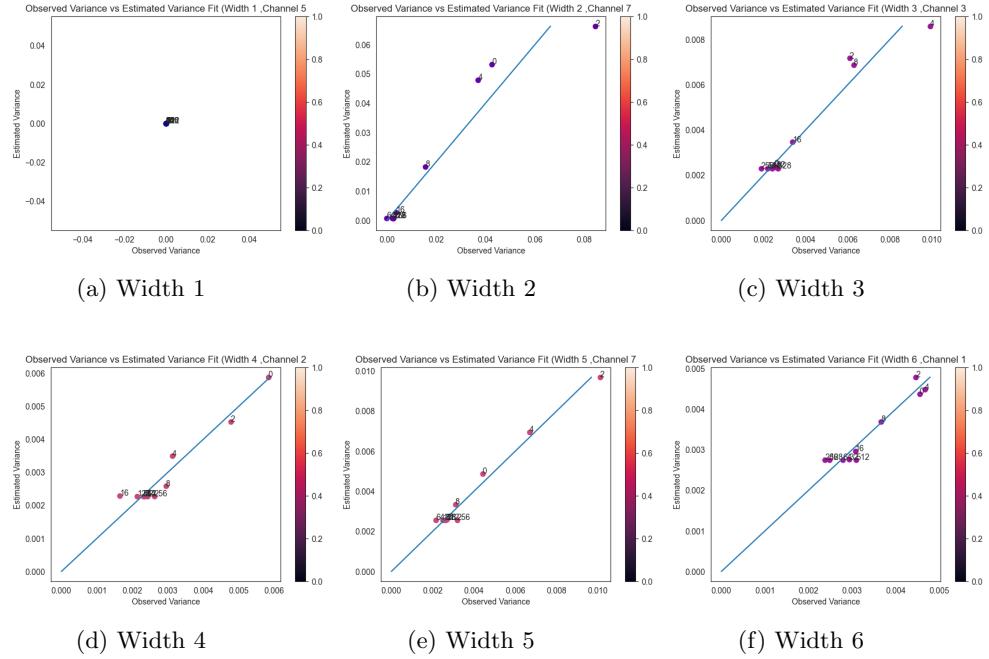
□

With the above proof, we can upper-bound our variance function by  $\mu(1 - \mu)$  for  $\mu = \mathbb{E}[X]$ . Thus, we define our few-parameter variance ansatz function in the following way:

$$Var_{\max,p} = p[\mu(1 - \mu)] \quad \left| \begin{array}{l} G_p(d) = 1 - e^{-p(d+1)} \\ D_p(d) = e^{-p(d+1)} \end{array} \right. \quad (3.4)$$

where  $Var_{\max,p}$  defines a parameterized upper-bound for the variance function,  $G_p$  defines the growth term and  $D_p$  defines the decay term. Putting these together, we can define the following parameterized variance ansatz function:

$$V(d) = Var_{\max,p_0} \cdot (G_{p_1}(d) \cdot D_{p_2}(d)) + p_3 \quad (3.5)$$



**Fig. 3.2: Suitability of Variance Ansatz for Variance of Per-Depth RB Success Probabilities** We compare the closeness of the estimated per-depth RB success probability variance based on Eq. 3.5 with observed per-depth RB success probability variances. On the  $x$ -axis, we plot the observed variance, and on the  $y$ -axis we plot the estimated variance. We also plot the  $y = x$  line as a point of reference for how close our estimates are to the observed variance.

To test the “goodness” of our variance ansatz function, we compare the observed variances of per-depth MRB success probabilities with the results from naive, least-squared error-based fits of our parameterized function (refer to Figure 3.2).

In the future, we look to move away from a variance-ansatz function defined via observation of the behavior of variance as a function of depth. Instead, we seek to use representation theoretic techniques to derive an accurate few-parameter variance ansatz function.

**4. Fitting RB Success Probabilities with Maximum Likelihood Estimation.** Having determined suitable few-parameter functions to model the mean and variance of RB success probability distributions, we use maximum likelihood estimation (MLE) in order to determine the optimal set of parameters that best fit observed RB success probability data. We present two kinds of models; a *fixed-width* model where the mean and variance ansatz functions are only dependent on circuit depth and a *multi-width* model where the mean and variance ansatz functions are dependent on both circuit width and depth. The fixed-width model serves as a replacement for ad hoc curve fitting. The multi-width model is a further improvement that allows us to simultaneously fit data from many RB experiments run on a multi-qubit device.

We present the fixed-width model as a replacement for ad hoc curve fitting and the multi-width model as a further improvement on the fixed-width model so as to fit the

exponential decay curves for RB experiments run on a range of circuit widths.

**4.1. Fixed-Width Model for Fitting RB Success Probabilities.** We now present the components of the *fixed-width* model. RB theory tells us the relationship between circuit depth and the mean of RB circuit success probabilities. Thus, we define the *mean ansatz function* as

$$\bar{S}_d = p_0 p_1^d + B \quad (4.1)$$

where  $B$  is determined by the RB protocol employed. The *variance ansatz function* is defined as above:

$$\text{Var}(\bar{S}_d) = p_2 \cdot \text{Var}_{\max}((1 - e^{-p_3(d+1)})(e^{-p_4(d+1)}) + p_5) \quad (4.2)$$

Finally, to perform MLE, defining a *likelihood function* is necessary.  $S_d$  denotes the  $d$ th entry of an array of success counts  $S$ ,  $N_d$  denotes the number of times a particular circuit at depth  $d$  was run,  $\alpha$  and  $\beta$  are the parameters for the Beta distribution, and  $f_\beta$  is the Beta distribution PDF:

$$L(S, N, \alpha, \beta) = - \sum_{d \in D} \log(f_\beta(S_d | N_d, \alpha, \beta)) \quad (4.3)$$

With the *fixed-width* model defined as above, we perform MLE on likelihood function  $L(S, N, \alpha, \beta)$ —defined as the sum of negative log-likelihoods for the Beta-Binomial distribution—in order to find optimal parameters  $\{p_i\}_{i \in \{0,1,2,3,4,5\}}$ . In Figure 4.1, we present some results with our fixed-width model. In particular, we bring attention to the larger plot where we are plotting the mean and standard deviations of calculating RB error rates and observing how this figure varies with the number of circuits sampled per-depth. We achieve a comparable level of accuracy to pyGSTi’s ad hoc curve-fitting method and do so in a structured and statistically-rigorous manner. This MLE-based fitting technique lends itself nicely, not only to a fixed-width setting, but also in a setting where multiple circuit widths are being benchmarked.

**4.2. Multi-Width Model for Fitting RB Success Probabilities.** As above, we present the components of the *multi-width* model. The *multi-width* model aims to generalize the *fixed-width* model by describing the statistical nature of per-depth RB success probabilities, not only as a function of circuit depth, but also circuit width. We define the *mean ansatz function* as

$$\bar{S}_d = p_0^w p_1^{wd} + B \quad (4.4)$$

The *variance ansatz function* is generalized from our above definition:

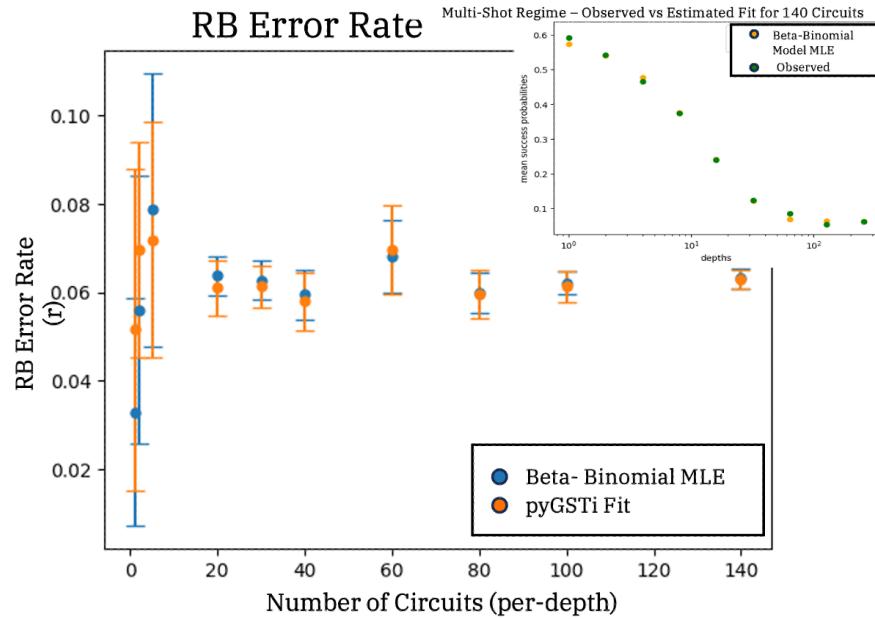
$$\text{Var}(\bar{S}_d) = p_2 \cdot \text{Var}_{\max}((1 - e^{-p_3 w(d+1)})(e^{-p_4 w(d+1)}) + p_5) \quad (4.5)$$

Finally, the *likelihood function* is defined as follows:

$$L(S, N, \alpha, \beta) = - \sum_{w \in W} \sum_{d \in D} \log(f_\beta(S_{w,d} | N_{w,d}, \alpha, \beta)) \quad (4.6)$$

As with the *fixed-width* model, we perform MLE on likelihood function  $L(S, N, \alpha, \beta)$  in order to find an optimal parameter set to fit the MRB success probability data; naturally, this likelihood function also sums over all the benchmarked circuit widths as the goal of the *multi-width* model is, as the name suggests, to simultaneously fit RB success probability

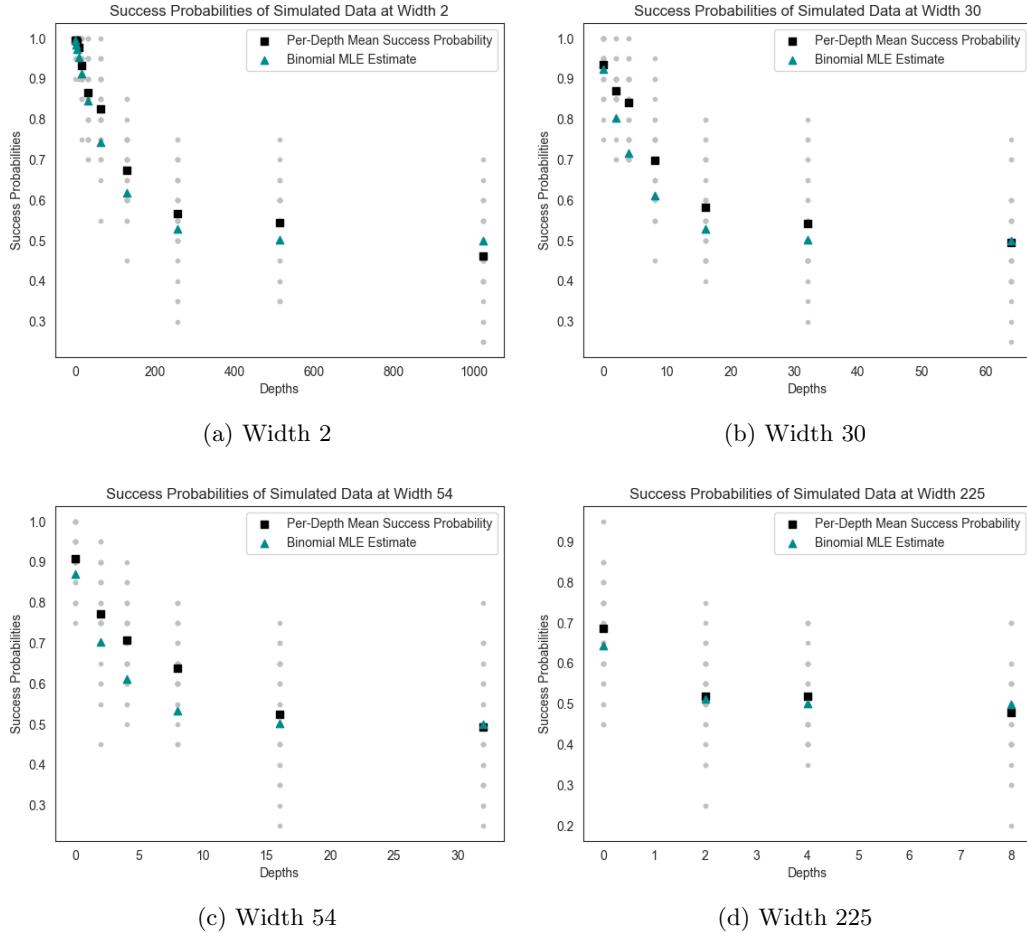
## Mean and Standard Deviation of Estimated RB Error Rate



**Fig. 4.1: Fixed-Width Model Fit with MLE for DRB Success Probabilities** We utilize DRB as our protocol of choice to test the functionality of our fixed-width model. As reference the mean ansatz function for DRB is  $\bar{S}_d = p_0 p_1^d + \frac{1}{2} w$  where  $w$  is 4 (number of qubits benchmarked). For the smaller plot, 140 circuits are sampled per-benchmarked depth for depths 0, 1, 2, 4, 8, 16, 32, 64, 128, and 256. This plot shows the estimated vs observed per-depth mean DRB success probabilities; observe the closeness of our fit. The larger plot shows how the RB error rate varies with respect to the number of circuits sampled. Naturally, the more circuits we sample, the more accurate our estimate becomes. We compare our estimated fit to pyGSTi's built-in exponential curve-fitting method; notice the consistency across depths in the fitting methods.

data across a range of circuit widths. We refer the reader to Figure 4.2 for results on fitting MRB data from a range of widths.

We provide some brief intuition for the form of the mean ansatz function. In accordance with RB theory, the parameter  $p_0$  (more commonly referred to with the variable  $A$ ) represents the effect of SPAM error on the fidelity of a RB circuit. Although  $p_0$  is independent of circuit depth—as SPAM error is only relevant to the beginning and end of an RB circuit—it is naturally dependent on the number of qubits in the circuit. This is due to the fact that the number of qubits determines the number of state preparations and measurements that need to be performed. Thus, we can roughly model SPAM infidelity as some  $p_0^w$  for  $p_0$  being the SPAM error for a single qubit. With regards to the rest of the function as well as the variance ansatz function, we conjecture there is an exponential dependence on circuit width as well as circuit depth wherever there was previously only an exponential dependence on circuit depth for the *fixed-width* model. As this is a conjecture, it remains to be shown that these functions are theoretically sound. Finally, we divide this fidelity decay term by 1/2 to account for the vacuously true case of  $w = 0$  as this would result in probability 1.



**Fig. 4.2: Multi-Width Model Fit with MLE for MRB+DFE Success Probabilities**  
 We use MRB as our protocol of choice to test the functionality of our multi-width model. We use simulated data whereby 30 circuits are sampled per width, depth pair, for a varying range of circuit depths per width, and for circuit widths 1, 2, 3, 4, 5, 7, 9, 13, 17, 23, 30, 40, 54, 72, 95, 127, 169, and 225. This plot shows in gray, the per-depth success probability data, in black, the mean for the gray data points, and the blue triangles representing the estimates using MLE. We observe that the plots somewhat match the trend of observed MRB+DFE success probabilities, however there is room for improvement.

**5. Conclusion.** Having laid the framework for a novel method to fit RB success probability data, we discuss the implications of our work with regards to benchmarking multi-qubit devices with a broader range of RB protocols. Fundamentally, our work leverages the underlying statistical properties of RB success probability data—the Beta distributed-nature of this data—to develop a statistically-inspired fitting method that, not only accurately fits benchmarking data for a fixed circuit width, but can extrapolate device capabilities for a range of circuit widths. Furthermore, we can do so with significantly less data compared to what is prescribed for standard RB protocols. For example, the *fixed-width* (See Figure 4.1)

and *multi-width* (See Figure 4.2) model results we collected were for circuits run for only 10 shots per-depth, as opposed to standard RB protocols which require tens of thousands of shots. Furthermore, this novel fitting method allows us to infer device performance for circuit widths that haven't even been benchmarked. Thus, with this work, we have established the foundation for making existing RB protocols more efficient.

In future work, we seek to strengthen the theoretical underpinnings of our models. Namely, we believe that there is room for improvement in the definition of our mean ansatz function in the context of MRB+DFE (which we hope to generalize to RB+DFE) as well as the variance ansatz function; with regards to the variance ansatz function—as mentioned in Section 3.2.2—we aim to develop a hybridized function that is inspired by representation theory as well as its observed behavior based on empirical evidence. We are also interested in making our data analysis more nuanced by considering which specific  $k$ -qubit subset is being benchmarked in an  $n$ -qubit QIP. Given that qubits on a multi-qubit device are not symmetrically arranged, and therefore suffer from different kinds of noise, we look to take this asymmetry into consideration to provide a more detailed understanding of a device's average noisy behavior.

## REFERENCES

- [1] J. CARRASCO, A. ELBEN, C. KOKAIL, B. KRAUS, AND P. ZOLLER, *Theoretical and experimental perspectives of quantum verification*, PRX Quantum, 2 (2021).
- [2] M. P. DA SILVA, O. LANDON-CARDINAL, AND D. POULIN, *Practical characterization of quantum devices without tomography*, Phys. Rev. Lett., 107 (2011), p. 210404.
- [3] S. T. FLAMMIA AND Y.-K. LIU, *Direct fidelity estimation from few pauli measurements*, Phys. Rev. Lett., 106 (2011), p. 230501.
- [4] J. HINES, M. LU, R. K. NAIK, A. HASHIM, J.-L. VILLE, B. MITCHELL, J. M. KRIEKEBAUM, D. I. SANTIAGO, S. SERITAN, E. NIELSEN, R. BLUME-KOHOUT, K. YOUNG, I. SIDDIQI, B. WHALEY, AND T. PROCTOR, *Demonstrating scalable randomized benchmarking of universal gate sets*, 2022.
- [5] N. L. JOHNSON, S. KOTZ, AND N. BALAKRISHNAN, *Beta distributions*, Continuous univariate distributions, 2 (1994), pp. 210–275.
- [6] D. LEIBFRIED, D. M. MEEKHOF, B. E. KING, C. MONROE, W. M. ITANO, AND D. J. WINELAND, *Experimental determination of the motional quantum state of a trapped atom*, Phys. Rev. Lett., 77 (1996), pp. 4281–4285.
- [7] E. MAGESAN, J. M. GAMBETTA, AND J. EMERSON, *Scalable and robust randomized benchmarking of quantum processes*, Phys. Rev. Lett., 106 (2011), p. 180504.
- [8] E. MAGESAN, J. M. GAMBETTA, AND J. EMERSON, *Characterizing quantum gates via randomized benchmarking*, Phys. Rev. A, 85 (2012), p. 042311.
- [9] E. NIELSEN, J. K. GAMBLE, K. RUDINGER, T. SCHOLTEN, K. YOUNG, AND R. BLUME-KOHOUT, *Gate Set Tomography*, Quantum, 5 (2021), p. 557.
- [10] M. A. NIELSEN, *A simple formula for the average gate fidelity of a quantum dynamical operation*, Physics Letters A, 303 (2002), pp. 249–252.
- [11] A. M. POLLORENO, A. CARIGNAN-DUGAS, J. HINES, R. BLUME-KOHOUT, K. YOUNG, AND T. PROCTOR, *A theory of direct randomized benchmarking*, 2023.
- [12] J. F. POYATOS, J. I. CIRAC, AND P. ZOLLER, *Complete characterization of a quantum process: The two-bit quantum gate*, Phys. Rev. Lett., 78 (1997), pp. 390–393.
- [13] T. PROCTOR, K. RUDINGER, K. YOUNG, E. NIELSEN, AND R. BLUME-KOHOUT, *Measuring the capabilities of quantum computers*, Nature Physics, 18 (2021), pp. 75–79.
- [14] T. PROCTOR, K. RUDINGER, K. YOUNG, M. SAROVAR, AND R. BLUME-KOHOUT, *What randomized benchmarking actually measures*, Phys. Rev. Lett., 119 (2017), p. 130502.
- [15] T. PROCTOR, S. SERITAN, K. RUDINGER, E. NIELSEN, R. BLUME-KOHOUT, AND K. YOUNG, *Scalable randomized benchmarking of quantum computers using mirror circuits*, Phys. Rev. Lett., 129 (2022), p. 150502.

## SCALABLE BENCHMARKING OF QUANTUM CHEMISTRY ALGORITHMS USING CIRCUIT MIRRORING

AIDAN Q. WILBER-GAUTHIER\* AND STEFAN K. SERITAN†

**Abstract.** Current benchmarking methods for measuring progress towards applications on quantum computers often lack scalability and specificity. This limits their ability to produce generalizable metrics of processor performance. Here, we describe a method for creating scalable application-specific benchmarks using mirrored subcircuits that quantify a device’s ability to execute particular subroutines or entire algorithms. We then apply this method to two quantum chemistry subroutines, performing noisy numerical simulations and interpreting their results within the paradigm of volumetric benchmarking. We show how to extract effective error rates and predict full circuit fidelities from the subcircuit data and demonstrate that we can distinguish differences in performance resulting from structural properties of the circuits.

**1. Introduction.** Quantum processors promise to revolutionize computing, possessing the ability to solve problems that are intractable on classical computers. Significant developments have been made in creating algorithms that may soon realize useful applications of quantum computing. Yet, the most interesting applications of quantum devices are stalled by complex and hard-to-predict hardware errors present in them. Many protocols have been developed to characterize such errors and evaluate the performance of quantum processors. Tomographic approaches such as gate set tomography (GST) [3] produce detailed models of device noise, but are currently not scalable beyond a few qubits. Other methods, such as randomized benchmarking (RB) [19] or IBM’s quantum volume benchmark [5], scale to at least a reasonable number of qubits, but provide high-level metrics of device performance that are not indicative of how well a device can execute real-world tasks. Novel benchmarking methods that reconcile scalability with the ability to provide tailored performance metrics are needed to better measure the capability of quantum devices.

As modern quantum processors approach the size necessary for small but potentially useful applications, there is growing interest in application-specific benchmarks that probe a device’s performance on a particular algorithm. Most current methods choose a set of one or more tasks to create a benchmarking suite, aiming to capture especially common or relevant programmatic structures that reflect realistic computations [1, 6, 7, 9, 12, 13, 14, 15]. Other recent works have presented frameworks for creating application-specific benchmarks for arbitrary applications, known as benchmark generators [8, 11]. Benchmarks created using these methods rely on either exponentially scaling classical simulation of the benchmarking circuits, exploitation of some problem-specific property that allows one to construct easily predictable circuits, or some other application-specific strategy for quantifying the quality of the benchmarking circuit outcomes. Scalable benchmarks can only be created using these frameworks if one implements a clever circuit construction or outcome verification strategy, detracting from the generality these methods aim to provide.

Here, we introduce a method for creating scalable volumetric benchmarks for almost any algorithm. From an application circuit, we sample a set of subcircuits of varying widths and depths that is representative of the full circuit. We then estimate the subcircuit fidelities using mirror circuit fidelity estimation (MCFE) [18], which provides an efficient, generalizable method for characterizing the performance of the subcircuits, resulting in application-agnostic scalability. We apply this method to two quantum chemistry application circuits and show how subcircuit fidelities can be used as a proxy for full circuit performance using effective error rates. After establishing the efficacy of these error rates, we demonstrate

---

\*Sandia National Laboratories, aqwilbe@sandia.gov

†Sandia National Laboratories, sserita@sandia.gov

that our benchmark is able to reveal performance differences between the two algorithms resulting from structural properties of the circuits.

**2. Benchmark Generation.** In this section we describe our method for generating application-specific benchmarks. Starting from an application-circuit  $\tilde{C}$ , we first pass the circuit through a minimal compilation procedure, which maps the circuit to our target device’s qubit graph containing only one- and two-qubit gates. The mapped circuit, written as a composition of  $w_C$ -qubit layers  $C = L_{d_C} \cdots L_1$ , is passed to one of our subcircuit selection routines, described in Section 2.1, producing a set  $\{S_C\}$  of subcircuits with varying, user-defined shapes. The subcircuits are then mirrored resulting in a set of mirror circuits  $\{M_C\}$  to run on the target device, after which we use mirror circuit fidelity estimation (MCFE) to quantify the performance of the subcircuits, as described in Section 2.2.

**2.1. Subcircuit Selection.** The goal of subcircuit selection is to generate a set of subcircuits that are representative of the full circuit. The primary difficulty in this task is the presence of two-qubit gates, since one cannot simply choose any range of circuit layers and subset of qubits without producing dangling gates — two-qubit gates where one qubit is in the selected subset and the other is not. In our simple method (Section 2.1.1), we choose to drop dangling gates from the subcircuit, whereas our connected components method (Section 2.1.2) avoids dangling gates entirely. In both cases, the user defines a set of subcircuit shapes (i.e., width-depth pairs) and chooses how many subcircuits to sample for each shape.

**2.1.1. Simple Method.** In our simple selection method, we pick subcircuits iteratively over all the width-depth pairs until the desired number of subcircuits is reached. To pick a subcircuit of width  $w$  and depth<sup>1</sup>  $d$ , we first generate the set of all connected subgraphs of  $w$  qubits on which the circuit is mapped. A starting layer  $L_i$  is chosen uniformly from the set of allowable starting layers  $\{L_1, \dots, L_{d_C-d}\}$  along with a subset of  $w$  connected qubits. Layers are added until the target depth is reached, dropping and counting dangling gates. While this method works well for small full circuit and subcircuit sizes, the generation of the connected qubit subgraphs is combinatorially difficult, limiting its use for circuits using more than a few dozen qubits. The restriction of uniformly sampling connected qubit subgraphs can be lifted to remove this computational bottleneck, but this was not explored in the current work.

**2.1.2. Connected Components Method.** In the connected components method, we consider each possible starting layer  $L_i$ , and partition the qubits into disjoint subsets that are not connected by a two-qubit gate. From the starting layer, we iterate over subsequent layers, unioning subsets when a two-qubit gate is found that connects them. Concurrently, we track the depth of each subset, ensuring that “inactive” layers (i.e., layers without operations on the chosen subset) are not counted. Whenever a  $w$ -qubit subset with depth  $d$  where  $(w, d)$  is a desired shape is found, we record the layers and qubits as a potential subcircuit, since we might find more subcircuits than desired. After our search is finished, we choose a random subset of the subcircuit options from each shape as our final set of subcircuits. In contrast to the simple method, this process is computationally efficient, but it does not always result in a sufficient number of subcircuits. Optionally, we can sample wider subcircuits by combining parallel subcircuits on disjoint sets of qubits or deeper subcircuits by dropping qubits from high-depth, high-width subcircuits, but even these steps do not guarantee the necessary number of subcircuits.

---

<sup>1</sup>For our purposes, we take depth to mean the *physical* depth, or how many unit-time ‘cycles’ the circuit takes to execute on the target device.

**2.2. Mirror Circuit Fidelity Estimation.** Here, we provide a brief overview of circuit mirroring [17], which underpins both mirror randomized benchmarking (RB) [19], a variant of RB that can scale to hundreds of qubits, as well as the mirror circuit fidelity estimation (MCFE) [18] technique that we will use to probe the performance of each subcircuit. Circuit mirroring is based on the simple idea that an error-free quantum circuit should be perfectly reversible, i.e. running the circuit forwards and backwards should return the system back to the initial state. Therefore, any deviation from the initial state must be due to errors in the circuit. If the initial state is known or easy to prepare, then one can directly measure this deviation and thus how noisy the circuit execution was. However, running the “mirrored” (i.e. combined forward and backward) circuit by itself is known as the Loschmidt echo, and is not sufficient to capture all the errors in the system. Thus, the version of circuit mirroring in MCFE uses three additional features: i) randomized compiling to twirl coherent errors and ensure they do not systematically cancel, ii) using ensembles of local random initial states with classical postprocessing to avoid needing complicated  $n$ -qubit initial states, and iii) reference experiments to correct for state preparation and measurement (SPAM) errors.

Calculating the fidelity of subcircuits with MCFE has several advantages that make it useful for creating application-inspired benchmarks. Firstly, it captures both stochastic and coherent errors. This is particularly important in the case of algorithmic circuits, which have more structure than the random circuits used in many other benchmarks and may interact with coherent errors in nontrivial ways. Secondly, it is a scalable technique due to the localized nature of both the random states and the randomized compilation, making the classical processing of the circuits to only scale linearly with number of circuit locations. This allows for the fidelity of even large subcircuits to be estimated efficiently using MCFE.

**3. Quantum Chemistry Applications.** Quantum chemistry has been identified as one possible “killer app” for quantum computing [20], and as such there has been much effort in developing quantum algorithms for chemistry and materials science in recent years. One common use case is ground state energy estimation, where the ground state of some molecular Hamiltonian is prepared on the quantum computer and quantum phase estimation (QPE) is used to read off the corresponding energy. It is usually the case that the ground state can only be prepared approximately with some overlap  $\gamma$  to the true ground state. Since QPE only has a chance to project onto the true ground state proportional to  $\gamma^2$ , it is often worth trying to improve this overlap. In fact, recent resource estimates show that even though accurate state preparation can be expensive, it is almost always worth doing in realistic scenarios [16].

Specifically, the quantum chemistry problem under consideration is the non-relativistic Born-Oppenheimer electronic Hamiltonian given by:

$$H = -\sum_{i=1}^{N_e} \frac{\nabla_i^2}{2} - \sum_{i=1}^{N_e} \sum_{a=1}^{N_A} \frac{\xi_a}{\|R_a - r_i\|} + \sum_{i \neq j=1}^{N_e} \frac{1}{2\|r_i - r_j\|}, \quad (3.1)$$

where  $i, j$  index  $N_e$  electrons with positions  $r$ ,  $a$  indexes  $N_A$  nuclei with atomic charge  $\xi$  and position  $R$ , and  $\|\cdot\|$  denotes the 2-norm. The three terms in the sum are the kinetic energy of the electrons, the nuclear-electron attraction, and the electron-electron repulsion, often denoted  $T$ ,  $U$ , and  $V$ , respectively. Working with the Hamiltonian in this form is known as *first quantization*, and the natural parameters to express the wavefunction in are the positions and momenta of the electrons.

However, one can also introduce a basis set (often Gaussians for molecules and planewaves

for materials) and express the Hamiltonian in *second quantization* instead:

$$H = \sum_{ij}^{N_{\text{orb}}} h_{ij} a_i^\dagger a_j + \sum_{ijkl}^{N_{\text{orb}}} g_{ijkl} a_i^\dagger a_j^\dagger a_k a_l. \quad (3.2)$$

In this formulation,  $ijkl$  index  $N_{\text{orb}}$  molecular orbitals constructed from the basis set,  $a^\dagger$  ( $a$ ) are creation (annihilation) operators, and the  $h_{ij}$  and  $g_{ijkl}$  coefficients are precomputed as integrals of the  $T$ ,  $U$ , and  $V$  terms from Eqn. (3.1) over the molecular orbitals. The convenience of second quantization is that now the natural parameters of the wavefunction are simply the coefficients of the molecular orbitals.

As we will showcase below, the state preparation circuits for first-quantized and second-quantized algorithms can differ in structure significantly. We would like to use application-inspired benchmarks to compare the performance of state preparation circuits in these two competing approaches.

**3.1. Antisymmetrization for First-Quantized State Preparation.** Recent work [21] has shown that fault-tolerant first-quantized algorithms can scale better than second-quantized algorithms. This is primarily due to the fact that there are often many more molecular orbitals needed in second-quantized approaches than there are actual electrons. However, one downside to moving to first quantization is that the trial state now must be explicitly antisymmetrized. As this is a major difference between first-quantized and second-quantized state preparation, we will focus on building a benchmark for the antisymmetrization circuit.

Antisymmetrization can be done efficiently on a quantum computer [2] with four steps:

1. Prepare an auxiliary register in an even superposition of all possible strings of the target length.
2. Sort this auxiliary register and store the outcome of each comparison.
3. Delete any collisions from the auxiliary register.
4. Apply the reverse of the sort to the target register.

The core routine of the antisymmetrization circuit is the sort, which must be done using a sorting network. An example sorting network is shown in Fig. 3.1 and consists of many (potentially parallel) comparator operations performing inequality tests and controlled-SWAPs.

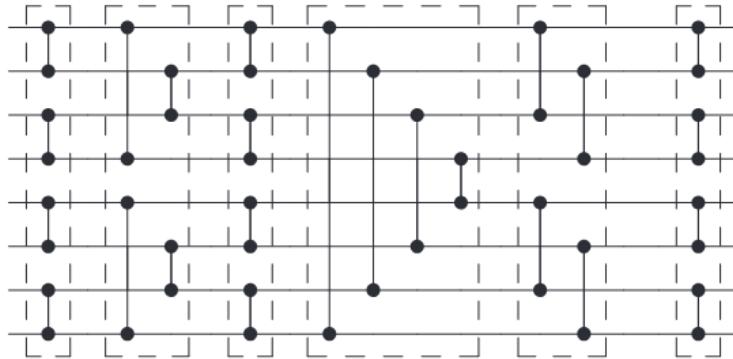


FIG. 3.1. Example bitonic sort on 8 inputs, which is a major component of the antisymmetrization circuit. Reproduced from [2].

**3.2. Block-Encoded Hamiltonians for Second-Quantized State Preparation.** Unlike the first-quantized approach above, the antisymmetrization of the wavefunction in

second quantization is handled by ensuring the proper anticommutation relations of the ladder operators in Eqn. (3.2). Instead, we only have to worry about preparing an accurate trial wavefunction. While there are many strategies for second-quantized state preparation, we will focus here on the near-optimal filter-based approach from Lin and Tong [10]. The key component in the filter-based state preparation approach is “block encoding” of the Hamiltonian in Eqn. (3.2), which we will implement using the linear combination of unitaries (LCU) approach [4]. The LCU approach generally consists of performing controlled rotations to set the  $h_{ij}$  and  $g_{ijkl}$  coefficients in a “selection” register, performing multicontrolled operations controlled by the selection register, and then unsetting the selection register. An example circuit for the  $H_2$  molecule can be found in Fig. 3.2.

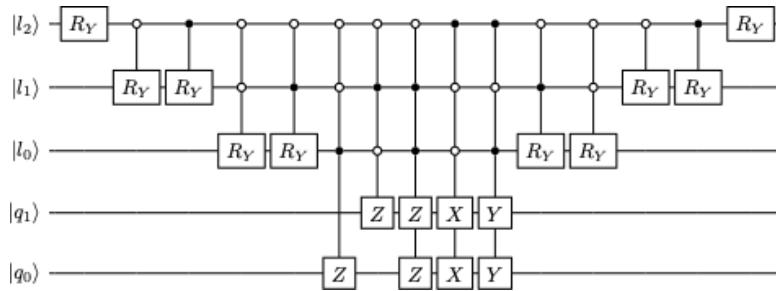


FIG. 3.2. Exemplar LCU circuit for the tapered  $H_2$  molecular Hamiltonian.

Unlike the bitonic sort in Fig. 3.1, one can see that the block-encoded Hamiltonian contains significantly more multicontrolled operations and uses the qubit registers much more inhomogeneously, i.e. there is a clear distinction between the selection register which act as controls and the system register which act as targets.

**4. Results and Discussion.** In this section, we present numerically simulated results of our benchmarks created for 11-qubit antisymmetrization (ASym) and LCU circuits. Both benchmarks were created using  $N_{sc} = 300$  subcircuits per width-depth pair for each width from 2 to 11 and exponentially increasing depths from 2 to 512, excluding widths 8 through 11 for the 512 depth. We used 100 mirror samples per subcircuit and 100 reference circuits for each width. The mirror circuits were simulated using a noise model with one- and two-qubit Hamiltonian and stochastic error generators. The coefficients of the error generators were sampled uniformly from 0 to a maximum strength varied by the error type. The strengths used for the one- and two-qubit Hamiltonian generators were 0.005 and 0.01, respectively, while those for the stochastic error generators were 0.002 and 0.004 for one- and two-qubits.

Figures 4.1(a) and (b) show the mean subcircuit fidelities by subcircuit shape for the ASym and LCU benchmarks. The fidelity serves as an indicator of the circuit’s performance, where a circuit executed perfectly will have a fidelity of 1.0, decreasing towards 0.0 as the circuit is executed less faithfully. As the width and depth of the subcircuits approach the width and depth of the full circuit, we expect that the process fidelity of the subcircuits approaches that of the full circuit. Looking to the volumetric benchmarking (VB) plots, we see that the subcircuit fidelities decrease with increasing size in both benchmarks, as we would expect since larger circuits admit more opportunities for errors to occur. Perhaps the more interesting quality of the plots is whether they reveal differing performance between the two algorithms. Figure 4.1(c) shows the difference between the mean subcircuit fidelities of the two benchmarks, from which we observe that the ASym mirror circuits were executed more faithfully than the LCU ones, with the difference becoming increasingly apparent at larger circuit sizes. The greater prominence of the performance differences are likely due

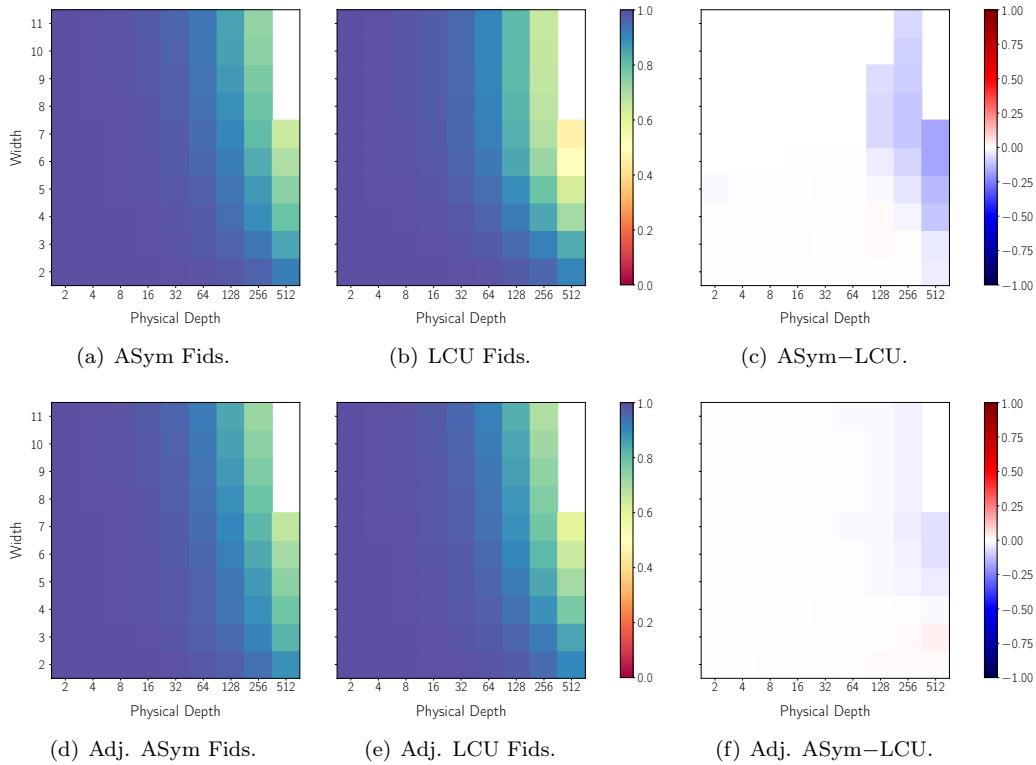


FIG. 4.1. Mean mirror circuit process fidelities by width and physical depth for ASym and LCU benchmarks and difference between mean fidelities between ASym and LCU. (a)-(c) use raw fidelities, whereas (d)-(f) use density-adjusted fidelities (denoted Adj.).

to larger subcircuits capturing more of the structure of the parent circuit, whereas smaller subcircuits are relatively indistinguishable from each other regardless of the circuit they were chosen from. In particular, a  $2 \times 2$  subcircuit may look similar whether sampled from the ASym or LCU circuit, but a  $7 \times 512$  subcircuit from one is unlikely to resemble a subcircuit of the same shape from the other.

**4.1. Density-adjusted fidelities.** Although circuit structure is a plausible explanation for the differences in the subcircuit fidelities between the two benchmarks, there are other relevant factors to consider, namely the gate density of the subcircuits (i.e., the ratio of the gate count to the circuit size). Generally, we would expect that, given two circuits with the same shape and perfect idles but different gate densities, the denser circuit will have a lower fidelity. In real systems, the two-qubit gate density is typically the driver of infidelity since idle and one-qubit gate errors are often significant. As the idle gates are perfect in our numerical simulations, we expect the density to impact the circuit fidelity.

Without accounting for the gate densities, then, we cannot attribute the differences in fidelities between the types of subcircuits to solely structural properties. Moreover, the subcircuits are not guaranteed to have equal densities to their parent circuits, which impacts how representative they are of the full circuit. Attempting to recover a density-agnostic measure of circuit performance, we computed a *density-adjusted fidelity*  $F_{adj}$  for each subcircuit to estimate the fidelity of the subcircuit if it had a target density  $\xi$  rather than its actual density  $\xi_{sc}$ . As a function of the subcircuit’s “raw” or unaltered fidelity  $F_{raw}$ , the

density-adjusted fidelity is given by

$$F_{adj} = (F_{raw})^{\xi/\xi_{sc}}. \quad (4.1)$$

Using these density-adjusted fidelities, we replicated the same VB plots as before, setting the target density to  $\xi = 1/8$  as a proxy for the average gate density of the two circuits, as shown in Figs. 4.1(d)-(f). These density-adjusted plots exhibit similar performance differences to those seen in the raw subcircuit fidelities, although slightly damped, indicating the original performance differences were likely artificially amplified by, but not entirely due to, unequal densities between subcircuits of the same shape.

**4.2. Predicting full circuit fidelity from subcircuits.** For our purposes, whether raw or density-adjusted fidelities are a better metric of performance primarily depends on which ones better align to full circuit performance. To test this, we used MCFE to measure the process fidelity of the full ASym and LCU circuits. Then, we used the benchmarking data to predict the fidelity of the respective full circuit using a simple error rate model, where we assume that the fidelity  $F_C$  of the full circuit  $C$  with size  $s_C = w_C d_C$  can be computed from an “effective” error rate  $\varepsilon_C$  by

$$F_C = (1 - \varepsilon_C)^{s_C}. \quad (4.2)$$

To predict  $F_C$  from a set of  $w$  by  $d$  subcircuits, we first compute an estimator  $\hat{\varepsilon}_{wd}$  of  $\varepsilon_C$  defined as

$$\hat{\varepsilon}_{wd} = 1 - GM \{F_{wd,1}, \dots, F_{wd,n}\}^{1/s}, \quad (4.3)$$

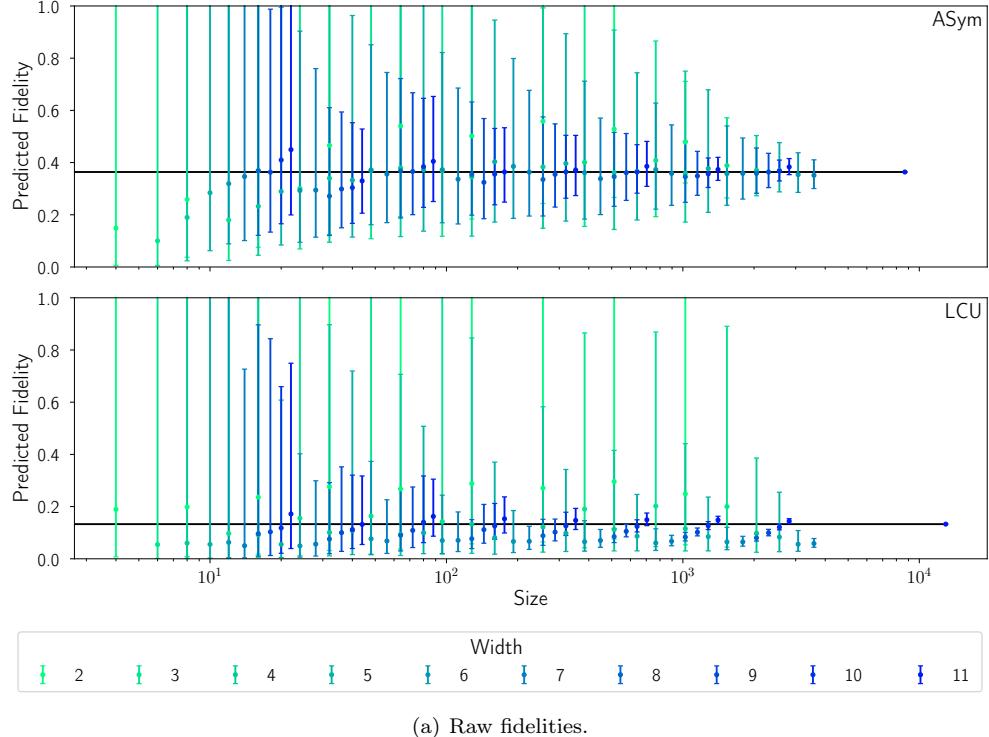
where  $s = wd$  is the size of the subcircuits,  $F_{wd,i}$  is the fidelity of the  $i$ th subcircuit out of the  $N_{sc}$  subcircuits of the particular shape, and GM denotes the geometric mean. The predicted fidelity  $\hat{F}_{wd}$  is then calculated analogously as

$$\hat{F}_{wd} = (1 - \hat{\varepsilon}_{wd})^{s_C}. \quad (4.4)$$

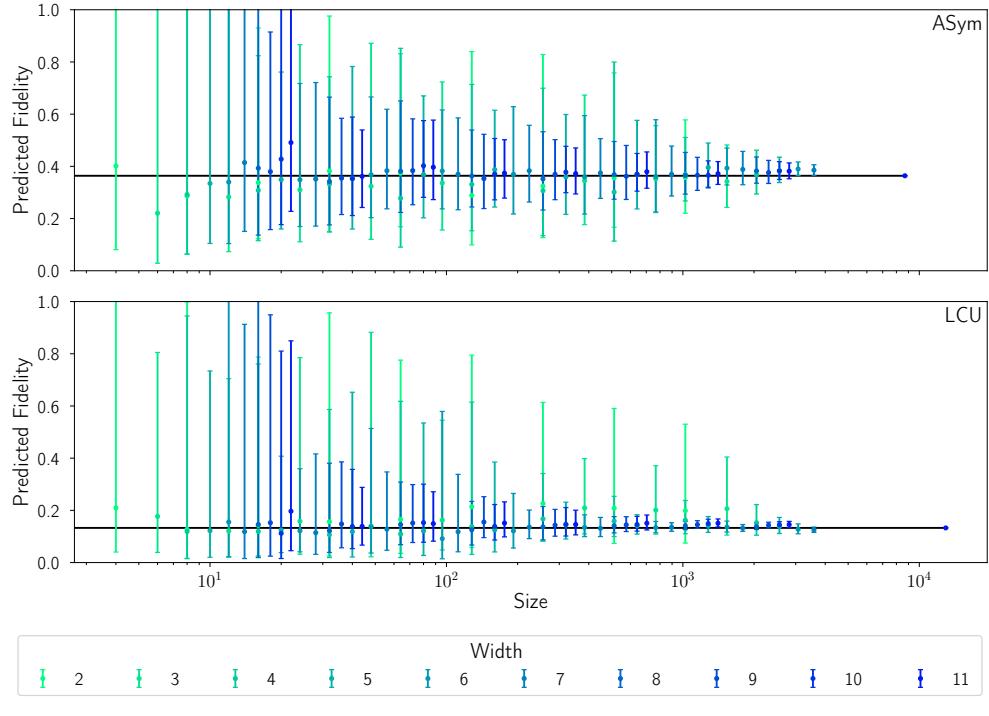
We then tested this model with both the raw and density-adjusted fidelities as shown in Figs. 4.2(a) and (b), respectively. In both figures, the error bars indicate the expected range of fidelities calculated from the geometric standard deviation, which is given by

$$\sigma_{GM} = \exp \sqrt{\frac{1}{N_{sc}} \sum_{i=1}^{N_{sc}} \left( \ln \frac{\hat{F}_{wd,i}}{\hat{F}_{wd}} \right)}, \quad (4.5)$$

where  $\hat{F}_{wd,i}$  is the fidelity prediction based on the  $i$ th  $w$  by  $d$  subcircuit. The upper and lower error bars are then given by  $\hat{F}_{wd} \cdot \sigma_{GM}$  and  $\hat{F}_{wd}/\sigma_{GM}$ , respectively (analogous to the arithmetic mean plus or minus the typical standard deviation). Since we wanted to test how well the subcircuits could predict the full circuit fidelity independently for each application, the target density used for computing the density-adjusted fidelities was set to the respective densities of the full circuits rather than a common one. For the ASym circuit, both the raw and density-adjusted fidelity predictions similarly tended toward the measured full circuit fidelity, though the spread of predictions was generally smaller using the density-adjusted fidelities as indicated by the error bars. For the LCU circuits, however, the raw subcircuit fidelities resulted in significantly worse predictions than the density-adjusted fidelities. The predictions using the raw fidelities did not result in an obvious consensus even at large subcircuit sizes, while the predictions using the density-adjusted fidelities show a



(a) Raw fidelities.



(b) Density-adjusted fidelities.

FIG. 4.2. Full circuit fidelity predictions by subcircuit size (width  $\times$  depth) for ASym (top) and LCU (bottom) circuits. Predictions using raw fidelities are shown in (a) and those using density-adjusted fidelities are shown in (b). Colors indicate width of subcircuits. Solid black line indicates actual full circuit fidelity.

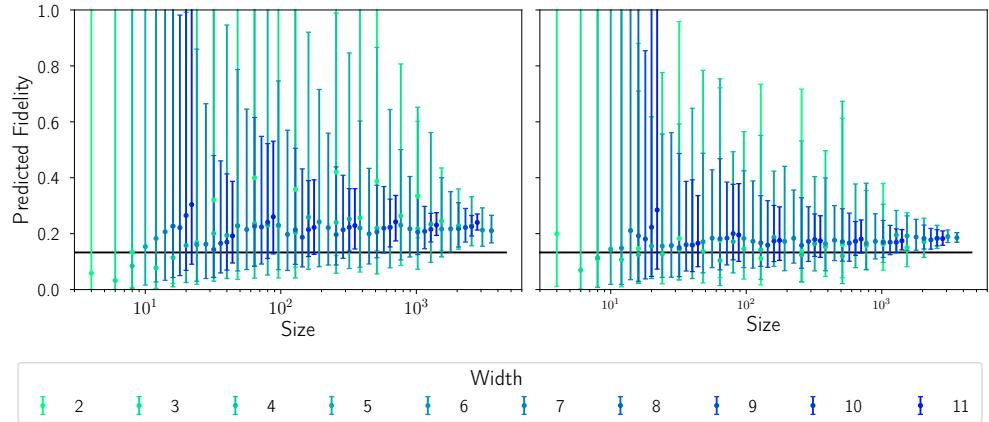


FIG. 4.3. Full circuit fidelity predictions for LCU circuit by subcircuit size (width  $\times$  depth) using ASym subcircuit data. Predictions using raw fidelities are shown on the left and density-adjusted fidelities on the right (target density set to LCU full circuit density). Colors indicate width of subcircuits. Solid black line indicates actual LCU full circuit fidelity.

much clearer convergence towards the full circuit fidelity. Notably, the mean density of the ASym subcircuits was within 0.5% of the full circuit density, while the mean density of the LCU subcircuits was more than 11% greater than the full circuit density, explaining why the raw ASym predictions worked relatively well whereas the raw LCU predictions were largely inaccurate.

Paying particular attention to the effect of the subcircuit width on the accuracy of predictions, we observe that high-width subcircuits resulted in significantly better predictions than those from low-width subcircuits of similar size, both in terms of the mean predicted value and the spread of predictions. Such results coincide with the well-established phenomenon that measured one- and two-qubit error rates — often used as metrics in periodic device calibration — are poor predictors of real device performance.

**4.3. Cross-predicting circuit performance.** The relative efficacy of the subcircuits in predicting the full circuit fidelities demonstrates that subcircuit fidelities, particular density-adjusted ones, are a sufficient proxy for measuring the performance of the full circuit. However, one might consider whether they reveal authentic distinctions in a device’s capability to run the full circuit due to structural differences in the circuits, or if the different results are simply a consequence of unequal circuit depth and density. If the latter claim were true, we would expect that, using our fidelity prediction model, we could predict the full circuit fidelity of the LCU circuit from the ASym subcircuits, or vice versa. In hopes of rejecting such a possibility, we attempted to do exactly this. Using ASym subcircuit fidelities to find the effective error rate, we computed the same predictions as before, but using the width, depth, and density (where applicable) of the LCU circuit, the results of which are depicted in Fig. 4.3. Using both the raw and density-adjusted fidelities, we see that the predictions do not converge to the full circuit fidelity, implying that we have identified performance differences that cannot be attributed to superficial properties of the circuits.

From this attempt at cross-predicting circuit performance, Fig. 4.3 demonstrates that the ASym data predict a greater fidelity than the LCU circuit actually has; that is, the full circuit fidelity predictions tend towards 0.2 from the ASym data while the true LCU circuit fidelity is closer to 0.15. More quantitatively, we can examine the differences in performance by comparing the effective error rates between the two circuits, given in Table 4.1. For each

TABLE 4.1  
*Actual ( $\varepsilon_C$ ) and estimated ( $\hat{\varepsilon}_{raw}$  and  $\hat{\varepsilon}_{adj}$ ) effective error rates for ASym and LCU circuits.*

	$\varepsilon_C$	$\hat{\varepsilon}_{raw}$	$\hat{\varepsilon}_{adj}$
ASym	$1.163 \times 10^{-4}$	$1.213(276) \times 10^{-4}$	$1.185(126) \times 10^{-4}$
LCU	$1.561 \times 10^{-4}$	$1.731(350) \times 10^{-4}$	$1.498(133) \times 10^{-4}$
Difference	$3.98 \times 10^{-5}$	$5.66 \times 10^{-5}$	$3.61 \times 10^{-5}$

circuit, the effective error rate was computed as

$$\varepsilon_C = 1 - (F_C)^{1/s_C}, \quad (4.6)$$

where  $s_C$  is the size of the full circuit and  $F_C$  is the fidelity estimate found using MCFE, which we note is consistent with Eqn. (4.2). To estimate the error rate from the subcircuit fidelities, we computed the error rate for each shape as in Eqn. (4.3) and then took a simple average over those error rates, resulting in the estimated error rates  $\hat{\varepsilon}_{raw}$  and  $\hat{\varepsilon}_{adj}$ , corresponding to estimates using raw and density-adjusted fidelities. It is worth mentioning that a more intelligent scheme could be used to average these error rates as we know that higher width or higher size subcircuits generally yield for accurate predictions; however, a simple arithmetic mean is used here to enable straightforward analysis and error bars.

The results in Table 4.1 coincide with much of what we have already seen in the fidelity predictions, namely that estimates using subcircuit data produce close approximations for the directly-measured full circuit error rates, with higher accuracy using the density-adjusted fidelities. A novel result yields from comparing the error rates for the different circuits, revealing significantly lower error rates for the ASym circuit. Notably, we find that the estimated error rates from density-adjusted subcircuit fidelities closely mirror the actual error rates, even in the relative difference between the two circuits, indicating that subcircuit data can accurately quantify performance differences between application circuits.

**5. Conclusions.** In this work, we described a general method for creating scalable benchmarks from any application circuit using mirrored subcircuits and applied it to two quantum chemistry application circuits. We showed using simulated results that subcircuit fidelities can accurately predict full circuit fidelities, giving credence to the notion of using an effective error rate as a mechanism for comparing the performance of different circuits. Crucially, we found that the fidelity predictions using subcircuit data are specific to the parent circuit, even when accounting for properties such as the width, depth, and gate density, indicating that our benchmarking scheme is able to distinguish differences in performance resulting from structural properties of the circuits rather than superficial ones.

These results demonstrate that subcircuit techniques for benchmarking provide a scalable proxy for full circuit performance, but several open questions remain. Whether these results can be replicated using a real device instead of a simulated noise model is still unclear. Moreover, our methods for subcircuit selection are imperfect, with the simple method suffering from computational complexity and the connected components method failing to guarantee a sufficient number of subcircuits. We also have yet to examine the differences in results between subcircuit selection methods, which could have significant consequences on how well a generated benchmark mirrors the performance of the application circuit and how we interpret the benchmarking data. Even without regard to subcircuit selection methods, more work is needed to assess the resilience of our fidelity prediction and error rate model, which currently lacks a strong analytical foundation. More sophisticated models may provide more accurate results or allow us to bound the error on our predictions.

## REFERENCES

- [1] M. BENEDETTI, D. GARCIA-PINTOS, O. PERDOMO, V. LEYTON-ORTEGA, Y. NAM, AND A. PERDOMO-ORTIZ, *A generative modeling approach for benchmarking and training shallow quantum circuits*, npj Quantum Information, 5 (2019), p. 45.
- [2] D. W. BERRY, M. KIEFEROVÁ, A. SCHERER, Y. R. SANDERS, G. H. LOW, N. WIEBE, C. GIDNEY, AND R. BABBUSH, *Improved techniques for preparing eigenstates of fermionic Hamiltonians*, npj Quantum Information, 4 (2018), p. 22.
- [3] R. BLUME-KOHOUT, J. K. GAMBLE, E. NIELSEN, K. RUDINGER, J. MIZRAHI, K. FORTIER, AND P. MAUNZ, *Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography*, Nature Communications, 8 (2017), p. 14485.
- [4] A. M. CHILDS AND N. WIEBE, *Hamiltonian Simulation Using Linear Combinations of Unitary Operations*, Quantum Inf. Comput., 12 (2012), pp. 0901–0924.
- [5] A. W. CROSS, L. S. BISHOP, S. SHELDON, P. D. NATION, AND J. M. GAMBETTA, *Validating quantum computers using randomized model circuits*, Physical Review A, 100 (2019).
- [6] P.-L. DALLAIRE-DEMERS, M. STEČHLY, J. F. GONTHIER, N. T. BASHIGE, J. ROMERO, AND Y. CAO, *An application benchmark for fermionic quantum simulations*, 2020.
- [7] Y. DONG AND L. LIN, *Random circuit block-encoded matrix and a proposal of quantum LINPACK benchmark*, Physical Review A, 103 (2021).
- [8] J. R. FINŽGAR, P. ROSS, L. HÖLSCHER, J. KLEPSCH, AND A. LUCKOW, *QUARK: A framework for quantum computing application benchmarking*, in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), 2022, pp. 226–237.
- [9] A. LI, S. STEIN, S. KRISHNAMOORTHY, AND J. ANG, *QASMBench: A low-level qasm benchmark suite for nisq evaluation and simulation*, 2022.
- [10] L. LIN AND Y. TONG, *Near-optimal ground state preparation*, Quantum, 4 (2020), p. 372.
- [11] T. LUBINSKI, S. JOHRI, P. VAROSY, J. COLEMAN, L. ZHAO, J. NECAISE, C. H. BALDWIN, K. MAYER, AND T. PROCTOR, *Application-oriented performance benchmarks for quantum computing*, 2023.
- [12] S. MARTIEL, T. AYRAL, AND C. ALLOUCHE, *Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way*, IEEE Transactions on Quantum Engineering, 2 (2021), pp. 1–11.
- [13] A. J. MCCASKEY, Z. P. PARKS, J. JAKOWSKI, S. V. MOORE, T. D. MORRIS, T. S. HUMBLE, AND R. C. POOSER, *Quantum chemistry as a benchmark for near-term quantum computers*, npj Quantum Information, 5 (2019), p. 99.
- [14] K. MESMAN, Z. AL-ARS, AND M. MÖLLER, *QPack: Quantum approximate optimization algorithms as universal benchmark for quantum computers*, 2022.
- [15] D. MILLS, S. SIVARAJAH, T. L. SCHOLTEN, AND R. DUNCAN, *Application-motivated, holistic benchmarking of a full quantum computing stack*, Quantum, 5 (2021), p. 415.
- [16] S. PATHAK, A. E. RUSSO, S. K. SERITAN, AND A. D. BACZEWSKI, *Quantifying t-gate-count improvements for ground-state-energy estimation with near-optimal state preparation*, Phys. Rev. A, 107 (2023), p. L040601.
- [17] T. PROCTOR, K. RUDINGER, K. YOUNG, E. NIELSEN, AND R. BLUME-KOHOUT, *Measuring the capabilities of quantum computers*, Nat. Phys., 18 (2022), pp. 75–79.
- [18] T. PROCTOR, S. SERITAN, E. NIELSEN, K. RUDINGER, K. YOUNG, R. BLUME-KOHOUT, AND M. SAROVAR, *Establishing trust in quantum computations*, arXiv, (2022).
- [19] T. PROCTOR, S. SERITAN, K. RUDINGER, E. NIELSEN, R. BLUME-KOHOUT, AND K. YOUNG, *Scalable randomized benchmarking of quantum computers using mirror circuits*, Physical Review Letters, 129 (2022).
- [20] M. REIHER, N. WIEBE, K. M. SVORE, D. WECKER, AND M. TROYER, *Elucidating reaction mechanisms on quantum computers*, Proc. Natl. Acad. Sci. U.S.A., 114 (2017), pp. 7555–7560.
- [21] Y. SU, D. W. BERRY, N. WIEBE, N. RUBIN, AND R. BABBUSH, *Fault-Tolerant Quantum Simulations of Chemistry in First Quantization*, PRX Quantum, 2 (2021), p. 040332.

## ARE HARD QUANTUM PROBLEMS ALSO HARD CLASSICALLY?

JUSTIN YIRKA\*, JOHN KALLAUGHER†, AND OJAS PAREKH‡

**Abstract.** A report on work done by Justin Yirka as an intern in summer 2023, studying quantum complexity theory related to optimization. Major topics included the Quantum Max Cut problem, hardness of finding optimal product states of local Hamiltonians, and defining interesting quantum constrained optimization problems.

The main result from the summer is a full classification of the complexity of estimating product states of families of 2-local Hamiltonians based on the allowed interaction terms.

*Acknowledgments.* Section 3 is primarily material from a paper in-preparation with John Kallaugher, Ojas Parekh, Kevin Thompson, and Yipu Wang.

**1. Introduction.** We consider several problems in quantum computing and complexity theory. In particular, we are interested in problems in Hamiltonian complexity, analogous to the study of classical constraint-satisfaction problems.

The quantum analogue to the NP-complete  $k$ -MAXSAT problem is the  $k$ -LOCAL HAMILTONIAN problem ( $k$ -LH). We are given a matrix on  $n$  qubits as a sum of matrices which act on at most  $k$ -qubits (and act as Identity on the others), and are asked to estimate its minimum eigenvalue.  $k$ -LH is the canonical QMA-complete problem, a quantum analogue of NP.

First, in Section 2, we focus on a restriction of  $k$ -LH known as QUANTUM MAX-CUT (QMC) which is also known to be QMA-hard, with the goal of developing a simpler proof that the problem is NP-hard. Here, QMA is the standard quantum analogue of the classical complexity class NP and  $\text{NP} \subseteq \text{QMA}$ . Proving NP-hardness is strictly weaker than proving QMA-hardness, but given that current proofs require opaque methods from perturbation theory, it would be exciting to find a simple, direct proof. In particular, QMC is a natural analogue of the classical problem MAX-CUT, and while we know MAX-CUT is reducible to QMC by the QMA-hardness result, we do not have a direct reduction. Here, we report on one idea for developing a simpler proof which we conclude appears unlikely to succeed.

Second, in Section 3, we investigate the following: if a family of Hamiltonians is such that finding optimal quantum solutions is QMA-hard, then is that same family is such that finding optimal classical, product-state solutions is NP-hard? A product-state is a quantum state with no entanglement. In other words, it is a tensor product of single-qubit states. Unlike general quantum states, product states have short classical descriptions. For this and other reasons, product states have become an important ansatz in the quantum computing literature, including in the study of approximation and optimization. We prove this conjecture is true.

Third, in Section 4, we investigate a variant of the problem in the previous section. An informal statement of the theorem proved in the previous section can be misinterpreted: what exactly is a “family” of Hamiltonians? In particular, while we prove examples of QMA-hard families of Hamiltonians generated from certain local interactions have NP-hard product states, we also find that more general sets of Hamiltonians which are QMA-hard do *not* necessarily have hard product states.

Finally, in Section 5, we consider defining interesting quantum constrained optimization problems. In classical computer science, simple objectives are known to produce complex problems. Sometimes, these objectives are combined with simple constraints to produce

---

\*The University of Texas at Austin and Sandia National Laboratories, jkyirka@sandia.gov

†Sandia National Laboratories, jmkall@sandia.gov

‡Sandia National Laboratories, odparek@sandia.gov

interesting, complex problems. For example, the VERTEX COVER problem is to cover all edges in a graph using as few vertices as possible. Similar problems include INDEPENDENT SET and BIN PACKING. These problems are useful beyond complexity theory and have become workhorses in developing and testing classical optimization techniques, an incredibly successful contribution of classical computer science. In contrast, interesting quantum constrained optimization problems are yet to be defined. We have analogs to constraint satisfaction problems, e.g. 3-SAT, but have yet to take the next step. Here we report on some initial ideas related to defining such problems.

We finish with a plan for future work in Section 6.

**2. Simpler proof that Quantum Max-Cut is NP-hard.** In this section, we study the QUANTUM MAX-CUT problem.  $X, Y$ , and  $Z$  refer to the single-qubit Pauli operators, and we use the convention of denoting tensor products similarly to standard multiplication, e.g.  $X_1 Z_2$  instead of  $X \otimes Z$ .

**DEFINITION 2.1** (QUANTUM MAX-CUT (QMC)). *Given parameters  $b, a > 0$  such that  $b - a \geq 1/\text{poly}$  and a Hamiltonian  $H = \sum_{i < j}^n w_{ij} H_{ij}$  acting on  $n$  qubits where all  $w_{ij} \geq 0$  and each  $H_{ij} = I - X_i X_j - Y_i Y_j - Z_i Z_j$ , decide whether  $\lambda_{\max}(H)$  is at least  $b$  or at most  $a$ .*

It's clear that the above can be viewed as a graph problem, where we set the weights  $w_{ij}$  equal to the edge weights of some undirected graph. QMC is in QMA, since it is a restriction of the problem  $k$ -LH. QMC is QMA-hard because of two facts. First, 2-LH restricted to the anti-ferromagnetic Heisenberg model (i.e. the same type of Hamiltonian as QMC but without Identity) is QMA-complete [8], and we can reduce a minimization problem to a maximization one by converting the input  $H$  to  $\|H\|I - H$ .

Note that  $\text{SWAP} := F = \frac{1}{2}(I + XX + YY + ZZ)$ , where SWAP is the 2-qubit unitary operation which exchanges the states of the qubits it acts on. So, we see that QMC considers Hamiltonians of the form  $\sum w_{ij}(2I - F)$ . The Heisenberg model is of the form  $\sum w_{ij}(XX + YY + ZZ)$  for  $w_{ij} \in \mathbb{R}$ , and the antiferromagnetic Heisenberg model means  $w_{ij} \geq 0$ . We say  $F$  is in the Heisenberg model and generally can ignore identity terms: implicitly, we drop the identity term and shift all eigenvalues/parameters by a constant. This can be very convenient for analysis—but also very annoying if we want to compute exact weights.

What about classical MAX-CUT? It is equivalent to computing the maximum eigenvalue of  $\sum_{i < j} \frac{w_{ij}}{2}(I - ZZ) = \sum_E \frac{w_{ij}}{4} w_{ij}(I - ZZ)$ . So if we were to redefine QMC to be a different family of Hamiltonians with just the  $Z$  terms, or if we restrict it to estimating the maximum energy of a *classical basis state* against the Heisenberg model, then it simulates classical MAX-CUT.

Of course, we know classical MAX-CUT can be reduced to general QMC since QMC is QMA-complete. But, the QMA-hardness proof of Piddock and Montanaro [8] is complicated. It requires the work of Cubitt and Montanaro [4] proving  $\{XX + YY\}$  is QMA-hard, which itself depends on proving other models are hard as in [2], and this depends on work by Kempe, Kitaev, and Regev [6], which has an extensive intro to the perturbation math, but is still difficult. This prompts us to ask whether there a simpler way to reduce classical MAX-CUT to QUANTUM MAX-CUT?

To attempt to answer this question, a natural approach is to study the reductions of [4] and [8] when applied to an instance of unweighted classical MAX-CUT, instead of to an arbitrary QMA-hard problem. Hopefully, we can avoid incorporating parts of the constructions which are only necessary for QMA-hardness. In particular, we will focus on the construction of [4] which simulates Hamiltonians which are sums of  $\{X, Z, XX, ZZ\}$  using sums of  $XX + YY$  interactions, i.e. the XY-model. Then, we will use the construction of [8] to reduce that to the antiferromagnetic Heisenberg model.

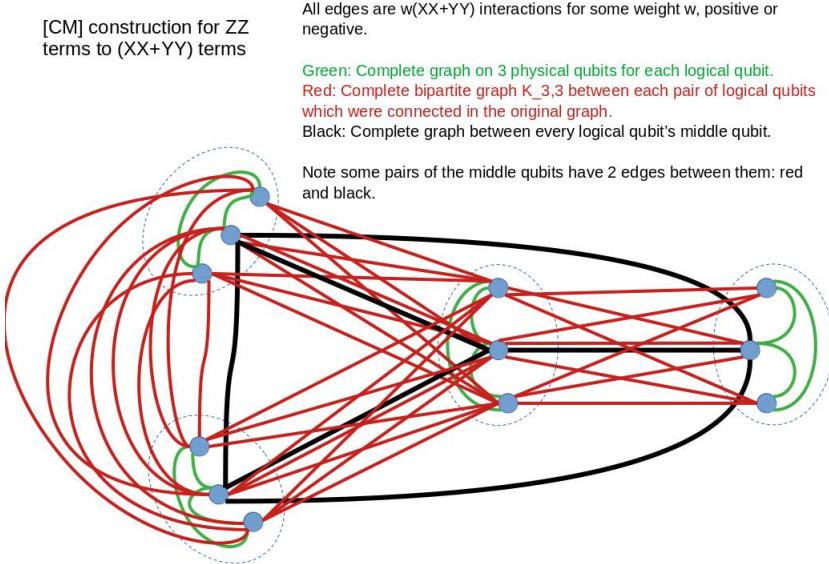


FIG. 2.1.

For the sake of brevity, we skip the details of the analysis, and proceed directly to stating some of our observations.

**2.1. Observations.** See Fig. 2.1 for the final interaction graph produced by the construction of [4] to simulate MAX-CUT, specifically beginning with the (3,4)-lollipop graph. This construction then would need to be run through the reduction of [8] to put the Hamiltonian into the XYZ model. We include this figure to demonstrate that unfortunately, the construction does not reduce classical MAX-CUT to a “simple” problem.

Importantly, the analysis of the Hamiltonian with this interaction graph still requires techniques from perturbation theory. We are able to avoid some details since we do not need QMA-hardness, but some details still remain. Perturbation gadgets necessitate most of the complicated analysis of prior works, so this is disappointing.

The figure is only a visual impression. We can make some more precise observations. For an instance of MAX-CUT on  $n$  vertices and  $m$  edges, when we apply both the [4] reduction and the [8] reduction, we get an instance of QMC with:

- $3n + 2 \left(\frac{1}{2}n(n+5) + 8m\right) = 3n + n(n+5) + 16m = n(n+8) + 16m$  vertices.
- $3 \left(\frac{1}{2}n(n+5) + 8m\right) = \frac{3}{2}n(n+5) + 24m$  edges.

Both of these are  $\Theta(n^2)$ . That means the final interaction graph is sparse, as it has  $\Theta(n^2)$  vertices and  $\Theta(n^2)$  edges, far fewer than the possible  $O(n^4)$  possible edges.

Let’s next think about the degree of each vertex. After applying the same reductions, we find that for a graph of  $n$  vertices and  $m$  edges, and  $d_i$  is the degree of vertex  $i$ , the final QMC Hamiltonian will be such that  $2n$  qubits have degree  $\Theta(d)$ ,  $n$  have degree  $\Theta(n+d)$ , and  $\Theta(n^2+m)$  qubits have degree  $O(1)$ . So even if the original graph was  $d$ -regular, the final Hamiltonian is very non-regular.

We can also consider groups of qubits, similar to idea of “clusters” described by Harrow and Brandao [3]. Unfortunately, there does not seem to be a good structure. Finally, we

can calculate rough order of magnitude estimates of the edge weights and observe they vary from  $O(1)$  on some edges to  $O(|E|)$  on others. More precision in the estimates of the edge weights appear to require very long calculations.

Finally, recall the motivation for this entire section. We had hoped to develop a proof that QMC is NP-hard that is simpler than the existing proof that QMC is QMA-hard. But, nothing in the observations we make directly above suggests, as far as we see, a way to remove more of the constructions of [4, 8] than we already have. In particular, we are not even able to avoid the use of perturbation theory. However, we note that analyzing the details of these reductions may be useful when studying QMC for other purposes, including the work in the next section.

**3. Product states from 2-local Hamiltonian terms.** We study whether families of Hamiltonians such that  $k$ -LH is QMA-hard also have product states which are NP-hard to optimize over. In [4], Cubitt and Montanaro classified the hardness  $k$ -LH when restricted to any family of 2-local Hamiltonian terms. We formalize restricting  $k$ -LH to a family of terms below.

**DEFINITION 3.1 ( $\mathcal{F}$ -LH).** *For  $\mathcal{F}$  any fixed set of 2-local Hamiltonian terms, define  $\mathcal{F}$ -LH as the problem  $k$ -LH with the additional promise that any input is of the form  $\sum_i \alpha_i H_i$  for  $\alpha_i \in \mathbb{R}, H_i \in \mathcal{F}$ .*

The main result of [4] which we are concerned with is the below.

**THEOREM 3.2** (Theorem 7 of [4]). *For  $\mathcal{F}$  any fixed set of 2-local Hamiltonian terms:*

- *If every matrix in  $\mathcal{F}$  is 1-local, then  $\mathcal{F}$ -LH is in P.*
- *Otherwise, if there exists a single-qubit unitary  $U$  such that  $U$  locally diagonalizes all elements of  $\mathcal{F}$  (i.e.  $U^{\otimes 2} H U^\dagger \otimes 2$  is diagonal), then  $\mathcal{F}$ -LH is NP-complete.*
- *Otherwise, if there exists a single-qubit unitary  $U$  such that for each 2-qubit  $H_i \in \mathcal{F}$ ,  $U^{\otimes 2} H_i U^\dagger \otimes 2 = \alpha_i Z^{\otimes 2} + A_i \otimes I + I \otimes B_i$  for some  $\alpha_i \in \mathbb{R}$  and 1-local Hermitian matrices  $A_i, B_i$ , then  $\mathcal{F}$ -LH is StoqMA-complete.*
- *Otherwise,  $\mathcal{F}$ -LH is QMA-complete.*

Note that Theorem 3.2 classifies *all* sets  $\mathcal{F}$ —in particular, the final case is a “catch-all” for sets which don’t belong to other cases.

In this work, we begin with sets for which  $k$ -LH is hard, but instead of further studying the complexity of estimating the optimal eigenstate of a Hamiltonian, we are interested in the complexity of estimating the optimal product state.

**DEFINITION 3.3** (Product state). *A state which is the tensor product of individual qubits:  $|\psi\rangle = \bigotimes_{i=1}^n |\psi_i\rangle$ .*

Informally, we are interested in the problems  $k$ -LH and  $\mathcal{F}$ -LH restricted to product states.

**DEFINITION 3.4** (PRODLH). *Given a  $k$ -local Hamiltonian  $H = \sum_i^m H_i$  on  $n$  qubits with  $m = \text{poly}(n)$  and all  $\|H_i\| = \text{poly}(n)$ , and two real parameters  $b - a \geq 1/\text{poly}(n)$ , decide:*

- *If there exists a product state  $|\psi\rangle$  such that  $\langle\psi| H |\psi\rangle \leq a$ , then output YES*
- *If for all product states  $|\psi\rangle$  it holds that  $\langle\psi| H |\psi\rangle \geq b$ , then output NO*

*promised one of these is the case. Define  $\mathcal{F}$ -PRODLH analogously to  $\mathcal{F}$ -LH.*

Our main theorem is the below.

**THEOREM 3.5.** *For any fixed set of 2-local Hamiltonian terms  $\mathcal{F}$ , if the problem  $\mathcal{F}$ -LH is NP-complete, StoqMA-complete, or QMA-complete, then the problem  $\mathcal{F}$ -PRODLH is NP-complete.*

Given that Theorem 3.2 of [4] shows all families fall into one of the above three cases or else is in P, we can state the following equivalent result.

**COROLLARY 3.6.** *For any fixed set of 2-local Hamiltonian terms  $\mathcal{F}$ , the problem  $\mathcal{F}\text{-LH}$  is in  $\text{P}$  if and only if  $\mathcal{F}\text{-PRODLH}$  is in  $\text{P}$ .*

In order to prove our theorem, we do the most straightforward thing and follow the proof of Theorem 3.2 from [4] to identify what techniques and constructions can be used for analyzing product states instead of eigenstates.

The statement of Theorem 3.2 gives a clear description of all  $\mathcal{F}$  such that  $\mathcal{F}\text{-LH}$  is NP-complete or StoqMA-complete. For the remaining sets  $\mathcal{F}$  which are QMA-complete, we require more structure than is given in the theorem statement. In the proof by [4] of their dichotomy theorem, they characterize the QMA-complete sets  $\mathcal{F}$  in more detail. We collect their results as a single lemma here.

**LEMMA 3.7.** *For a fixed set of 2-local Hamiltonian terms  $\mathcal{F}$ , the problem  $\mathcal{F}\text{-LH}$  is QMA-complete if and only if conjugating all terms in  $\mathcal{F}$  by  $U \otimes U$  for single-qubit unitary  $U$  and taking linear combinations of terms is sufficient to construct one of the following: (1)  $XX + \beta YY + \gamma ZZ + AI + IA$  for some real  $\beta, \gamma$ , at least one of them nonzero, and some single-qubit Hermitian  $A$ ; (2)  $XX + YY$ ; (3)  $XX + YY + AI + IA$  for some single-qubit Hermitian  $A$ ; (4)  $XX + \gamma ZZ$  for some real nonzero  $\gamma$ ; (5)  $XX + \gamma ZZ + AI + IA$  for some real nonzero  $\gamma$  and single-qubit Hermitian  $A$ ; or (6)  $XZ - ZX + AI - IA$  for some single-qubit Hermitian  $A$ .*

For brevity, we do not include details of the above lemma here.

As suggested by the statement of the lemma, a crucial tool is the fact that for any single-qubit unitary  $U$ , the complexity of  $\mathcal{F}\text{-LH}$  is unaffected by conjugating each element of  $\mathcal{F}$  by  $U^{\otimes n}$ . This is because conjugating all terms in  $\mathcal{F}$  by  $U \otimes U$  is equivalent to conjugating any Hamiltonian  $H = \sum_i \alpha_i H_i$  constructed from terms  $H_i \in \mathcal{F}$  by  $U^{\otimes n}$ , which does not change eigenvalues. Crucially for our purposes, because the conjugation is by products of single-qubit unitaries, it also does not change the spectrum of possible expectation values of product states. This and other important facts are review in the appendix.

We are now ready to prove our main theorem.

*Proof. (of Theorem 3.5.)* If  $\mathcal{F}\text{-LH}$  is NP-complete or StoqMA-complete, then Theorem 3.8 with Theorem 3.9 prove that  $\mathcal{F}\text{-PRODLH}$  is NP-complete.

If  $\mathcal{F}\text{-LH}$  is QMA-complete, then  $\mathcal{F}$  falls into one of the six cases described above in Theorem 3.7. We show in Theorem 3.11 that for the first through fifth cases,  $\mathcal{F}\text{-PRODLH}$  is NP-complete. Theorem 3.10 shows this for the sixth case.

The above shows the forward direction of the theorem. For the reverse direction, according to Theorem 3.2 the only sets  $\mathcal{F}$  for which we have not shown  $\mathcal{F}\text{-LH}$  is NP-hard and  $\mathcal{F}\text{-PRODLH}$  is NP-complete are  $\mathcal{F}$  such that every element is 1-local. In that case, both problems are in  $\text{P}$ .  $\square$

We now state the lemmas required in the proof of the main theorem. Theorem 3.8 is analogous to Theorem 3.7, transforming terms in some cases into a simplified form. In Theorem 3.9, we show how to encode the objective function of the classical NP-complete problem MAX-CUT into the expectation of product states against Hamiltonians constructed from that simple form. Together, those two lemmas resolve the cases where  $\mathcal{F}\text{-LH}$  is NP- or StoqMA-complete.

For the sake of brevity in this summary article, we do not include the proofs of the next two lemmas. They are the simplest cases, so are somewhat less interesting than others.

**LEMMA 3.8.** *For any fixed set of 2-local Hamiltonian terms  $\mathcal{F}$ , if the problem  $\mathcal{F}\text{-LH}$  is NP-complete or StoqMA-complete, then conjugating all terms in  $\mathcal{F}$  by  $U \otimes U$  for single-qubit unitary  $U$  and taking linear combinations of terms is sufficient to construct a term of the form  $\alpha ZZ + \beta(IC + CI)$  for some  $\alpha \neq 0$ , some  $\beta$ , and some single-qubit Hermitian  $C$ .*

**LEMMA 3.9.** *If  $\mathcal{F}$  contains a term  $H = \alpha ZZ + \beta(IC + CI)$  for  $\alpha \neq 0$ , real  $\beta$ , and some single-qubit Hermitian  $C$ , then  $\mathcal{F}\text{-PRODLH}$  is NP-complete.*

Now we move on to the lemmas necessary for the cases when  $\mathcal{F}$ -LH is QMA-complete. The statements of Theorems 3.10 and 3.11 are similar to Theorem 3.9 just above, and their proofs begin similarly as well. In particular, the following two proofs reduce to an objective function involving multi-dimensional vectors, rather than the simple  $\pm 1$  labels of standard MAX-CUT. In Theorem 3.12, we prove that this objective function is indeed NP-complete to optimize over.

**LEMMA 3.10.** *If  $\mathcal{F}$  contains a term  $H = XZ - ZX + \delta(AI - IA)$  for some single-qubit Hermitian  $A$  and real  $\delta$ , then  $\mathcal{F}$ -PRODLH is NP-complete.*

For the sake of brevity in this summary article, we do not include the proof of Theorem 3.10.

**LEMMA 3.11.** *If  $\mathcal{F}$  contains a term  $H = XX + \beta YY + \gamma ZZ + \delta(AI + IA)$  for real  $\beta, \gamma$  with at least one of  $\beta, \gamma$  nonzero and some 1-qubit Hermitian  $A$  with any real  $\delta$ , then  $\mathcal{F}$ -PRODLH is NP-complete.*

*Proof.* As in the previous proofs, we begin by using a symmetrization gadget to remove the 1-local terms. Given four qubits, define  $G = H_{ab} + H_{cd} - H_{bd} - H_{ac}$ . This is a rectangle with positive horizontal edges and negative vertical edges. This gadget is equivalent to

$$G = (X_a - X_d)(X_b - X_c) + \beta(Y_a - Y_d)(Y_b - Y_c) + \gamma(Z_a - Z_d)(Z_b - Z_c).$$

Now we consider see how  $G$  interacts with product states on four qubits. Define  $\rho^a, \rho^b, \rho^c, \rho^d$  as in the previous proofs. We find that

$$\begin{aligned} \text{Tr}[G\rho^a\rho^b\rho^c\rho^d] &= (x^a x^b - x^a x^c - x^b x^d + x^a x^c) + \beta(y^a y^b - y^a y^c - y^b y^d + y^a y^c) \\ &\quad + \gamma(z^a z^b - z^a z^c - z^b z^d + z^a z^c) \end{aligned}$$

With a little thought, this is equivalent to

$$(r^b - r^c)^T \begin{pmatrix} 1 & & \\ & \beta & \\ & & \gamma \end{pmatrix} (r^a - r^d).$$

For convenience, label the transformation/weight matrix as  $W$ .

Now, consider an arbitrary graph. As in the previous proofs, associate a “vertex qubit” with each vertex and construct a gadget  $G^{ij}$  for each edge  $i, j$ , such that two of its vertices are vertex qubits and two of its vertices are ancilla qubits. The vertex qubits may be shared among several gadgets, while the ancilla qubits are part of only one gadget.

Let qubits  $a$  and  $d$  in each gadget be the vertex qubits. If we fix the states of the vertex qubits and minimize the expectation on a single gadget, the minimum is achieved when  $r^b = -\frac{W(r^a - r^d)}{\|W(r^a - r^d)\|}$  and  $r^c = -r^b$ , which yields an expectation of

$$-2 \|W(r^a - r^d)\|$$

Given an arbitrary graph, substituting our gadget for every edge constructs a Hamiltonian such that the minimum expectation of any product state is equal to

$$2 \min_{v_i \in S^2} \sum_{\substack{ij \in E \\ i < j}} -\|Wv_i - Wv_j\|. \tag{3.1}$$

Similarly to the previous proof, the objective function here is a sum of distances rather than *squared* distances as in the definition of RANK- $k$ -MAXCUT. And here, the distances are warped by a weight matrix.

The remainder of the proof is deferred to Theorem 3.12 below. To meet the conditions of the below lemma, observe the following. At least one of  $\beta, \gamma$  are nonzero and they may be any real number. We may multiplicatively scale  $W$  so all weights have magnitude at most 1, and one of them exactly equals 1. Next, only the magnitudes of the weights, not the signs, matter because our objective function leaves the ancilla qubits  $b$  and  $c$  in each gadget free to flip the sign as necessary. So, we can treat all of the weights as positive. Finally, without loss of generality we may assume the entries of  $W$  are ordered as  $1 \geq \beta \geq \gamma$ .  $\square$

**LEMMA 3.12.** *Fix a matrix  $W = \text{diag}(1, \beta, \gamma)$  for some constants  $1 \geq \beta \geq \gamma \geq 0$ . Given some parameter  $\alpha$  and a graph  $G = (V, E)$ , it is NP-complete to decide whether  $\Delta(G) \leq \alpha$ , for*

$$\Delta(G) := \min_{v_i \in S^2} \sum_{i,j \in E, i < j} -\|Wv_i - Wv_j\|.$$

*Proof.* We reduce from the 3-coloring problem. We have two constructions, one for when the three weights are equal and one for all other cases. The construction and proof for the case  $1 = \beta = \gamma$  will be given later, and we begin with our construction for all other cases.

Consider an arbitrary instance of 3-COLORING on a graph  $G = (V, E)$  on  $n$  vertices and  $m$  edges and construct  $H = (V', E')$  by replacing every edge with a 3-clique, i.e. a triangle  $K_3$ . That is, for each edge  $ij \in E$ , we add a vertex  $k_{ij}$  and edges  $ik_{ij}$  and  $jk_{ij}$  to form  $H$ .

Let  $\Gamma = \Delta(K_3)$ , the minimum objective value on a 3-clique. We claim  $G$  is 3-colorable if and only if the value  $\Delta(H) \leq m\Gamma$ .

$\Rightarrow$ : Suppose  $G$  is 3-colorable. Let  $\{u, v, w\}$  be any fixed set of three vectors that achieve  $\Gamma$  on  $K_3$ . Then we may assign one of the three vectors  $u, v, w$  to each color, such that every vertex in  $G$  is assigned a vector and no adjacent vertices have the same vectors. We copy those vectors to the vertices of the graph  $H$ , and for each constructed 3-clique  $i, j, k_{ij}$  in  $H$  we assign to  $k_{ij}$  the vector in  $\{u, v, w\}$  not assigned to  $i$  or  $j$ . By construction, this assignment will yield a value of  $\Gamma$  on each of the  $m$  3-cliques  $i, j, k_{ij}$ , so  $\Delta(H) \leq m\Gamma$ .

$\Leftarrow$ : Suppose there exists a set of vectors achieving  $\Delta(H) \leq m\Gamma$ . Since the minimum value on any  $K_3$  is  $\Gamma$ , the vectors must achieve the minimum on each of the  $m$  3-cliques in  $H$  corresponding to edges in  $G$ .

We will first consider when  $\gamma = 0$ , so the only nonzero weights are 1 and  $\beta \leq 1$ . Consider  $\Delta(K_3)$ . The objective on a 3-clique is equivalent to fixing three points on an ellipse so as to minimize the negative of the perimeter of the resulting inscribed triangle. The ellipse has horizontal semi-major axis 1 and vertical semi-minor axis  $\beta$ .<sup>1</sup> Henceforth, we will just refer to the problem of finding a *maximum* perimeter triangle in the ellipse, understanding this is equivalent to achieving the minimum value  $\Gamma$ .

To proceed, we use an observation related to billiard trajectories, which are inscribed polygons such that at each vertex, the angle of incidence equals the angle of reflection. They are relevant here because any inscribed polygon of maximum perimeter is known to be a billiard trajectory [9]. For a given ellipse, let  $P$  be the maximum perimeter of any inscribed triangle. Given any one point fixed on the ellipse, there exists a unique pair of points on the ellipse such that the resulting triangle achieves the maximum perimeter  $P$  (see [7] or 17.6.6 of [1]).

Now, consider any 3-clique  $i, j, k_{ij}$  in  $H$  corresponding to an edge in  $G$ . Let  $v_i, v_j, v_k$ , respectively, be the vectors assigned to the vertices. As we stated earlier, these vectors form a triangle that achieves the maximum perimeter on an ellipse with semi-axes 1 and  $\beta$ . If

---

<sup>1</sup>A semi-axis is half the length of a line from the center to the ellipse, similar to a radius.

this clique shares a vertex with another 3-clique  $a, j, k_{aj}$ , and so shares at least one vector  $v_j$ , then by our observation above we have that the vectors assigned to  $a, j, k_{aj}$  must also be  $v_i, v_j, v_k$  (not necessarily in that order) to also achieve the maximum perimeter. Extending this argument to the full connected graph shows that every constructed 3-clique in  $H$  must be assigned the same three vectors. And because each constructed 3-clique shares at most one vertex, no adjacent vertices are assigned the same vector. Therefore, we can assign colors to each vector, yielding a 3-coloring of  $H$  and thus of  $G$ .

The above argument concludes the proof when there are only two nonzero weights. Now, we consider when the third weight  $\gamma$  is also non-zero (but still not  $1 = \beta = \gamma$ ). Above, the objective on a 3-clique corresponded to an ellipse. Here,  $\Delta(K_3)$  corresponds to a three-dimensional ellipsoid with semi-axes  $1, \beta$ , and  $\gamma$ . Minimizing the objective is equivalent to finding three points on the surface of the ellipsoid such that the perimeter of the resulting inscribed triangle—drawn through the interior of the ellipsoid—is maximized.

Any triangle inscribed in an ellipsoid is also inscribed in a two-dimensional ellipse equal to a cross-section. Observe that there always exists a cross-section with its two semi-axes equal to the largest two of the ellipsoid. The semi-axes of an ellipse fully determine the maximum perimeter of a triangle inscribed in the ellipse. Given two ellipses with semi-axes  $A, B$  and  $C, D$  such that  $A \leq B$  and  $C \leq D$ , the maximum perimeter of a triangle inscribed in the second ellipse is clearly greater than in the first. From this we conclude that a triangle achieving maximum perimeter in the ellipsoid must exist in a cross-section with semi-axes equal to the two largest semi-axes of the ellipsoid (cf. [5, Theorem 4]).

When the weights  $1, \beta, \gamma$  are all distinct or  $1 = \beta > \gamma$ , the above observation reduces the problem to a single two-dimensional plane, so a three-coloring follows from our previous argument.

In the case that  $1 > \beta = \gamma$ , then the maximum-perimeter triangle must be in a plane that contains the  $x$ -axis such that the cross-section is an ellipse with major-axis 1 and minor-axis  $\beta = \gamma$ . Now, consider any 3-clique  $i, j, k_{ij}$  in  $H$  corresponding to an edge in  $G$  and assigned vectors  $v_i, v_j, v_k$  which form a maximum-perimeter triangle. Suppose this clique shares a vertex with another 3-clique  $a, j, k_{aj}$ , and so shares at least one vector  $v_j$ . First, suppose that none of these points is  $\pm(1, 0, 0)$  on the  $x$ -axis. Then there is a unique plane which contains the point  $v_j$  and the  $x$ -axis, so both clique's triangles must be in the same plane. As we saw before, extending this argument shows that all of the 3-cliques must be assigned vectors in the same plane, and a 3-coloring follows as in the two-dimensional case. Second, suppose one of the points is on the  $x$ -axis. Now, if  $v_j$  is the point on the  $x$ -axis, then the two cliques may use different planes which have identical cross-sections but are rotated about the  $x$ -axis. So, all 3-cliques must be assigned the same set of three vectors forming identical triangles *up to* a rotation about the  $x$ -axis. To 3-color  $H$  and thus  $G$ , we assign a color to points on the  $x$ -axis, to points with positive  $y$ -coordinate, and to points with negative  $y$ -coordinate (and arbitrarily but consistently when  $y = 0$ ).

This concludes the proof for all possible weights except when  $1 = \beta = \gamma$ .

We omit the proof for the case  $1 = \beta = \gamma$  for the sake of brevity. Due to the increased numbers of symmetries, it requires modifying the previous construction to use 4-cliques instead of 3-cliques as well as a global “sink” vertex.  $\square$

**4. Product state hardness in other Hamiltonians.** In this section, we continue studying the problem of showing whether Hamiltonians which are QMA-hard have NP-hard product states. Now, we consider the problem  $k$ -LH restricted to a set  $\mathcal{S}$ . By this we mean the input is restricted to the set  $\mathcal{S}$ . This is in contrast to the problem  $\mathcal{F}$ -LH in the previous section, which places restrictions on how the set  $\mathcal{S}$  is construction, namely that a set of small

terms  $\mathcal{F}$  is used to construct the family of larger Hamiltonians  $\mathcal{S}$  through arbitrary addition and weighting. Here, we just have that  $\mathcal{S}$  can be any infinite set of  $n$ -qubit Hamiltonians.

We show that it is possible to construct a set of QMA-hard 3-local Hamiltonians such that finding the optimal product states of the Hamiltonians is *not* NP-hard. We prove for  $k \geq 4$  first, then sketch how to reduce to 3-local in a later section. This result raises the question of differences between specifying terms versus full Hamiltonians, as well as 2-local versus higher-locality Hamiltonians.

**THEOREM 4.1.** *There exists an infinite set of 4-local Hamiltonians  $\mathcal{S}$  such that  $k$ -LH restricted to  $\mathcal{S}$  is QMA-hard but estimating the optimal product state is easy.*

*Intuition.* For any local Hamiltonian, we can encode its action on the  $|0\rangle / |1\rangle$  basis term-by-term using two of the Bell states, which requires doubling the number of qubits. We can use a penalty Hamiltonian to penalize the other two Bell states, so the overall Hamiltonian preserves the original minimum eigenvalue. Then, we consider product states. On each pair of physical qubits encoding a logical qubit, we penalize the Bell states  $|\psi^\pm\rangle \propto |01\rangle \pm |10\rangle$ . So, every such pair of physical qubits must be orthogonal to  $|\psi^\pm\rangle$  or else suffer a large penalty. Note that  $|\psi^-\rangle \propto |vv^\perp\rangle - |v^\perp v\rangle$  for every  $|v\rangle$ , so any product state with a pair which is not symmetric will have some overlap with  $|\psi^-\rangle$  and be penalized. As for symmetric states, any state containing  $|01\rangle$  or  $|10\rangle$  will have some overlap with  $|\psi^+\rangle$ , so only states spanned by  $|00\rangle$  and  $|11\rangle$  are ‘safe’. And finally, since we demand product states, the qubits cannot be in a superposition of the two safe states. The only *product states* which are orthogonal to both  $|\psi^\pm\rangle$  are  $|00\rangle$  and  $|11\rangle$ .

Once we’ve seen that the penalty Hamiltonian forces states to be a product of  $|00\rangle$  and  $|11\rangle$ , we use the fact that the rest of the Hamiltonian, the “computational” part, is spanned by states  $|\phi^\pm\rangle$ . This state has equal overlap with  $|00\rangle$  and  $|11\rangle$ , so any state which is a product of any combination of these states will have the same expectation. Finally, we show that expectation is efficient to calculate, since it’s just proportional to the trace of each term.

We omit the proof for the sake of space.

The above proof sketch begins with an arbitrary QMA-hard  $k$ -local Hamiltonian and constructs a  $2k$ -local Hamiltonian. So, it provides an example set of QMA-hard  $k$ -local Hamiltonians with easy product states for any  $k \geq 4$ . This leaves open sets of 2- and 3-local Hamiltonians.

To construct a set of QMA-hard 3-local Hamiltonian with easy product states, begin with a Hamiltonian restricted to an interaction graph on a 2-dimensional grid, which is known to be QMA-complete. Color each qubit in a checkboard pattern. We’ll encode all the ‘white’ qubits using the Bell basis as above, and leave the ‘black’ qubits alone. Now, every interaction between nearest-neighbor logical qubits is an interaction between one ‘white’ and one ‘black’ logical qubit, which is a total of 3 physical qubits.

First, similar to the previous construction, we can show this preserves the minimum eigenvalue, so the Hamiltonians remain QMA-hard. Second, we want to argue that computing the optimal product state can be done efficiently. It’s not quite as simple as before, but we can still do it. For any ‘white’ logical qubit which we encoded in the Bell basis, a similar argument as before can show that assigning  $|00\rangle$  or  $|11\rangle$  will result in the same expectation against any 3-local term acting on it, and will do better than any other possible state. For any ‘black’ qubit, which we did not encode in the Bell basis, consider the up to four 3-local terms through which it interacts with its neighbors. After assigning  $|00\rangle / |11\rangle$  arbitrarily to each of its white neighbors, we can optimize by brute-force to find the state of the black qubit which is optimal against the four interaction terms. Repeating this for every black qubit, we can compute the optimal product state efficiently. Therefore, the product states cannot be NP-hard.

**5. Quantum Vertex Cover.** The general  $k$ -LH problem asks to estimate the minimum eigenvalue of a  $k$ -local Hamiltonian. This is similar to determining satisfiability of a Boolean formula or solving Max-SAT, which are forms of optimization problems. But, there are many other types of interesting problems classically. In this section, we study constrained optimization problems including constrained versions of  $k$ -LH.

For example, the classical (minimum) Vertex Cover problem asks us, given a graph  $G = (V, E)$ , to determine the smallest subset  $S \subseteq V$  such that every edge in  $E$  is incident to  $S$ . This problem is NP-complete. We can phrase this as an optimization problem:

$$\min_{x \in \{0,1\}^n} \text{HW}(x) \quad \text{such that} \quad x_i \vee x_j = 1 \text{ for all } i, j \in E.$$

On the left is the objective and on the right is the constraint. An interesting feature of this type of problem is that on their own, the objective and constraint are each easy to satisfy: the objective is optimized by the all-0 string and the constraint is trivially satisfied by the all-1 string. The difficulty of the problem comes from the combination.

A tempting but incorrect direction is to define the problem in any number of ways which requires optimizing over some QMA-complete Hamiltonian. But, this means the objective function would be very fine-tuned and complex. We would prefer to find a problem with a simple objective and simple constraints which together produce complexity, as is possible classically.

First, we consider VERTEX COVER. Observe that classical VC is equivalent to computing  $\min_{\rho} \sum_{i \in V} \text{Tr}[C_i \rho]$  such that  $\text{Tr}[\tilde{H}_{ij} \rho] \leq \epsilon$  for all  $i, j \in E$  where  $C_i = (I - Z_i)/2$ ,  $\tilde{H}_{ij} = |00\rangle\langle 00| = (I - Z_i) \otimes (I - Z_j)/4 = I - H_{ij}$  for rank-3  $H_{ij}$ , and  $\epsilon = 0$ .

So to generalize to a quantum version of the problem, maybe we should just adjust  $C, H, \epsilon$ . This may be a promising direction, but appears difficult. Interestingly though, we can define another, classical, problem which is sort of a middle-ground with the quantum problem.

Define Probabilistic-VC defined as  $\min_{\rho} \mathbb{E}_{x \sim \rho}(|x|)$  such that for all  $i, j \in E$  we have  $\Pr_{x \sim \rho}(x_i \vee x_j) \geq 1 - \epsilon$ , where  $\rho$  is a classical, diagonal distribution. As far as we can tell, this problem has not been studied. But, it is exactly what we are considering for the quantum problem, just restricting the density matrix  $\rho$  to be classical. We have a sketch of an argument that, at least for  $\epsilon = 0$ , this problem is in NP. Based on the number of constraints, we can define a linear program with a solution with rank of  $\rho$  upperbounded by  $|E|$ . It's unclear if the problem is hard, though.

In contrast to this new problem which we are considering, there is another variant that is well-understood. Consider the classical Fractional Vertex Cover problem. Here, for every vertex  $i$ , assign a weight  $p_i \in [0, 1]$  such that every edge  $i, j \in E$  must have  $p_i + p_j \geq 1$ , and try to minimize the sum of the weights  $p_i$ . Note that this is a linear program and can be solved in polynomial time. (The integer program for a standard VC problem is 0-1 valued, and relaxing to a linear program only outputs an approximation, specifically the solution to the fractional problem!) Changing  $p_i + p_j \geq 1 - \epsilon$  will still yield a linear program, so it's still polynomial time. But, this is distinct from our Probabilistic VC problem. In particular, the problem we're describing considers all diagonal  $\rho$ , which allows for correlations between the  $p_i$ , up to 2nd moments, while the Fractional Vertex Cover problem has all weights/probabilities assigned independently as in a product distribution.

It appears plausible that studying this new, classical Probabilistic-VC problem could shed light on defining quantum problems. Analyzing Probabilistic-VC seems difficult because of the presence of distributions, and quantum mixed states would force us to confront the same difficulties when defining and analyzing quantum problems.

**6. Future work.** This article includes partially-completed work in Section 3, which should soon be finished.

The work described in this article suggests the following problems for future study:

1. How do the results of Section 3 change if we consider  $\mathcal{F}$ -LH where  $\mathcal{F}$  is a set of  $k$ -local terms for  $k > 2$ ?
2. We were unable to define quantum constrained optimization problems with interesting complexity. Do such problems exist? What are they?
3. Convexity plays an enormous role in the study of optimization. Can knowledge from convex optimization help to understand the role of quantum mixed states in the complexity of quantum problems?

#### REFERENCES

- [1] M. BERGER, M. COLE, AND S. LEVY, *Geometry II*, Universitext, Springer Berlin Heidelberg, 2009.
- [2] J. D. BIAMONTE AND P. J. LOVE, *Realizable hamiltonians for universal adiabatic quantum computers*, Physical Review A, 78 (2008), p. 012352.
- [3] F. G. BRANDAO AND A. W. HARROW, *Product-state approximations to quantum states*, Communications in Mathematical Physics, 342 (2016), pp. 47–80.
- [4] T. CUBITT AND A. MONTANARO, *Complexity classification of local Hamiltonian problems*, SIAM Journal on Computing, 45 (2016), pp. 268–316.
- [5] V. DRAGOVIĆ AND M. RADNOVIĆ, *Periodic ellipsoidal billiard trajectories and extremal polynomials*, Communications in Mathematical Physics, 372 (2019), pp. 183–211.
- [6] J. KEMPE, A. KITAEV, AND O. REGEV, *The complexity of the local hamiltonian problem*, Siam journal on computing, 35 (2006), pp. 1070–1097.
- [7] G. LION, *Variational aspects of Poncelet’s theorem*, Geometriae Dedicata, 52 (1994), pp. 105–118.
- [8] S. PIDDOCK AND A. MONTANARO, *The complexity of antiferromagnetic interactions and 2d lattices*, Quantum Information & Computation, 17 (2017), pp. 636–672.
- [9] S. TABACHNIKOV, *Geometry and billiards*, vol. 30, American Mathematical Soc., 2005.

### III. Machine Learning

Articles in this section discuss the application of machine learning techniques to problems in computational physics and mathematics. This includes the design of neural network architectures, the efficient use of data for training and validation, or the use of machine learning to provide insights into materials, high-energy applications, atmospheric phenomena, and structural mechanics.

1. *Dalman* and *Barone* use sensor fusion to improve unsupervised learning approaches for the classification of turbulent flow data.
2. *Davis*, *Geraci* and *Motamed* leverage Fourier feature networks for multi-fidelity surrogate modeling.
3. *Furrik*, *Wood* and *Hensley* develop a machine-learned interatomic potential for nickel-platinum catalysts.
4. *Huynh*, *Meissner*, *Bochev* and *Kuberry* obtain inverse Sobolev-type inequalities for tanh neural networks to facilitate asymptotic convergence.
5. *Islas Quinones* and *Maupin* validate the use of surrogate machine learning models to capture model form error for uncertainty quantification.
6. *Pawar*, *Bochev* and *Owen* develop a neural ODE surrogate for the flux solution vector in coupled transmission problems.
7. *Polifrone*, *Bayat*, *Goff* and *Xu* study grain boundary segregation in alloys using machine-learned interatomic potentials.
8. *Meissner*, *Huynh*, *Kuberry* and *Bochev* leverage elliptical regularity to improve physics-inspired neural networks for Stokes equations.
9. *Mullen* and *Sikorski* investigate the effect of organizing training data on machine-learned interatomic potentials.
10. *Ngangmeni* and *Maupin* utilize random forest models to identify model form error for uncertainty quantification.
11. *Snyder*, *Tezaur* and *Wentland* investigate domain decomposition-based coupling of physics-informed neural networks via the Schwarz alternating method.
12. *Vera Cruz* and *Sikorski* develop machine-learned interatomic potentials for ultra-high-temperature ceramic composites.
13. *Villatoro*, *Geraci* and *Schiavazzi* showcase the effectiveness of coordinate encoding in multi-fidelity neural network models.

S.K. Seritan  
B.W. Reuter

November 28, 2023

## AN UNSUPERVISED LEARNING SENSOR FUSION APPROACH FOR CLASSIFICATION OF TURBULENT FLOW DATA

BENJAMIN DALMAN\* AND MATTHEW BARONE†

### **Abstract.**

In this work, unsupervised approaches for the classification of turbulent flow data are explored. First, the common usage of models in this field is discussed, and several issues in the common usage of the models are identified. Solutions to these issues are then proposed, in the form of a Bayesian filtering approach which probabilistically incorporates multiple sources of data to improve confidence in a result. A sensor is proposed for use this method, and tested on the turbulent channel flow case.

### **1. Introduction.**

**1.1. Unsupervised clustering approaches in fluids.** Unsupervised learning approaches have shown past success in automatically categorizing various turbulent flow fields into regions that are roughly consistent with prior knowledge [1, 2, 7]. These approaches can utilize different machine learning (ML) techniques (Gaussian mixture models, k-means clustering, self-organizing maps, etc), as well as different approaches for hyperparameter selection (feature selection, optimal number of clusters, etc). Barone et al. [1] utilized Gaussian mixture models (GMMs) and an algorithmic approach to selecting hyperparameters (called feature subset selection) to classify four different turbulent flow cases, identifying 19 unique clusters across all four cases shown. Callaham et al. [2] also utilized GMMs with sparse-principal component analysis (SPCA), but relied on a dominant-balance physics-based approach to guide the selection of features, and the SPCA approach to select an appropriate number of clusters. Callaham et al. [2] showcased their method on a several different flow cases.

Wu et al. [11] presented the use of self-organizing maps to automatically detect the transition between laminar and turbulent domains. They discuss the use of various “detector flow variables” which are later used to compare with their own results. Common issues with these traditional methods are discussed. Saetta and Tognaccini [9] also demonstrated the use of GMMs, and contrasted them to the “deterministic” methods for flow classification in shockwave-boundary layer interaction (SW-BLI) cases. Manual searches were conducted to determine useful features to use for classification in different cases. The optimal number of clusters to use was determined in advance for each case, but the choices were justified by comparing the Bayes Information Criterion (BIC) values from the models with different numbers of clusters. Lyne [5] used a k-means clustering approach to identify coherent structures in turbulent flows. Reproducibility was ensured by running the training 1000 times and selecting the model with the minimum intra-class variance. Silhouette analysis and intra-class variance were used as metrics to determine optimal number of clusters.

Although these approaches have seen some success, there remains several major issues. For one, the non-deterministic nature of the unsupervised learning process can lead to different results from run to run (a repeatability issue). A second problem (or an extension of the first) is that the common solution to the repeatability issue relies on heuristic measurements. Although they are generally acceptable, the performance of these heuristics can vary from case to case which limits their applicability in the general sense. A final issue is that there is currently no way to actually guide the unsupervised clustering results to a particular result if the clustering of some points is known in advance. This work proposes a solution these

---

\*University of Southern California, dalman@usc.edu. Financial support for this work provided in part by the PSAAP-III program, DOE/NNSA Contract DE-NA0003993, dalman@usc.edu

†Sandia National Laboratories, mbarone@sandia.gov

issues, by introducing a method to combine the results of multiple unsupervised models, as well as more established traditional flow sensors.

**1.2. Determinism and Repeatability in Gaussian Mixture Models.** Gaussian mixture models have become a popular choice in recent studies attempting to automatically classify turbulent flow data. GMMs assume that the data can be represented as a mixture of Gaussian distributions, and then attempt to automatically form clusters in the feature space to best fit this assumption. In this way, GMMs account for the means of each cluster (as is done in k-means clustering), and the covariances of clusters as well. The fitting process for GMMs is done using the iterative statistical “expectation-maximization” (EM) algorithm, which is guaranteed to converge to a local optimum. The initialization process for this EM algorithm can be done either completely randomly, or by utilizing the results from a simpler clustering algorithm as a starting point. Obviously for a highly nonlinear clustering problem, different initializations can result in vastly different converged results. GMMs in this work are implemented using the GaussianMixtureModel package in scikit-learn for python [8].

Various studies have taken different approaches to ensure repeatable and consistent results for different GMM initializations. Callaham et al. [2] did not discuss any specific measures taken to ensure repeatability from their GMMs; however in their publically available sample code they used a constant seed for their GMM initialization. Although seeding of the random parameters can ensure consistency from run to run (if done properly), it does little to ensure repeatability of the experimental results - i.e. how can we be certain this result is not just a lucky seed?

Other studies have taken additional steps to enable repeatability of results. Barone et al. [1] performed 40 initializations of each GMM, and kept only the model with the “best” results. Saetta & Tognaccinni [9] performed 100 independent initializations and reported that the percentage of flow classified in the viscous region remained constant. Lyne [5] performed 1000 initializations of each model and kept the best, selected by intra-class variance. All of these approaches are vastly superior to making no effort to ensure reproducibility, however they can yet be improved. In particular, having independent initializations with no variation is possible only for some problem-specific cases (where the model converges to the same optimum each time), and for Saetta & Tognaccinni [9] it is likely possible due to using only two clusters which simplifies the clustering problem. Additionally, the methods used by Barone et al. [1] and Lyne [5] can be replicated in other contexts to ensure reproducible results, but rely heavily on a definition of the “best” model initialization result. Both approaches use different definitions for the “best” model, and although they both produce acceptable results for the given cases there is once again no generalizability. In short, although many studies see value in the ability of these unsupervised clustering approaches to remove manual human input, they trade that problem specific knowledge for general heuristic criterions which have no guarantee to work for all cases.

To examine the influence of some of these factors on a simple clustering problem, we examine the case of a turbulent channel flow at several different Reynolds numbers. Turbulent channel flow comes from Lee & Moser [4]. Gaussian mixture models are used for clustering, and the feature set used is the components of the turbulent kinetic energy (TKE) equation, normalized by the dissipation term. Five clusters are chosen to match with results found by Barone et al. [1]. Further discussion of the other methods for selecting optimal hyperparameters (features and number of clusters in this case) is covered further down. In all cases, the GMMs are trained with a convergence tolerance of  $10^{-10}$  and a maximum number of iterations of 1000, matching the values used by Saetta & Tognaccinni [9] (although default values were also found to be sufficient for most cases). In order to verify that utilizing some heuristic approach for “best” could at least ensure reproducibility,

results from 100 model initializations are combined into an ensemble average. For each point, the amount of times it was classified into a given cluster is counted, after which a “classification fraction” is determined measuring the likelihood of a point being classified into a cluster using identical features and hyperparameters.

Results from GMM classifications of the channel flow case using TKE equation features are shown in Fig 1.1, for two different types of GMM initialization methods (k-means and random). In both cases, an ensemble of 100 models was used, and for each ensemble model 20 individual models were generated and only the “best” was kept (best was measured by the BIC criterion).

We can see that although the average results for both sets are similar (the same general clusters are consistently identified), the k-means initialization seems to result in the most consistent results (lowest spread), while the random initialization still has a noticeable spread. This likely indicates that the k-means initialization will tend to favor similar clusterings, while the random initialization will better explore the complete cluster space. Additionally, as the random initialization explores more of the space, it also returns “best” models (by the BIC metric) that can be less physically representative of our understanding of the system. When we compare the results for k-means initialization in Fig. 1.1a to results for random initialization in Fig. 1.1b, we see that the demarcation between the viscous sub-layer and the buffer layer (clusters 2 and 4 in the figures) changes from  $y^+ = 4$  to  $y^+ = 2$ . Our domain knowledge would suggest the demarcation should occur around  $y^+ = 5$ , so in this case the random initialization performs worse. We expect that this difference is hidden by the k-means initialization case because of the reduced exploration of the problem space for that method. This case is an example of where our heuristic for picking the “best” model fails to identify what an expert would consider to be the best, and also serves to show that even with fairly repeatable settings (in Fig. 1.1a) we can still see some spread in our results.

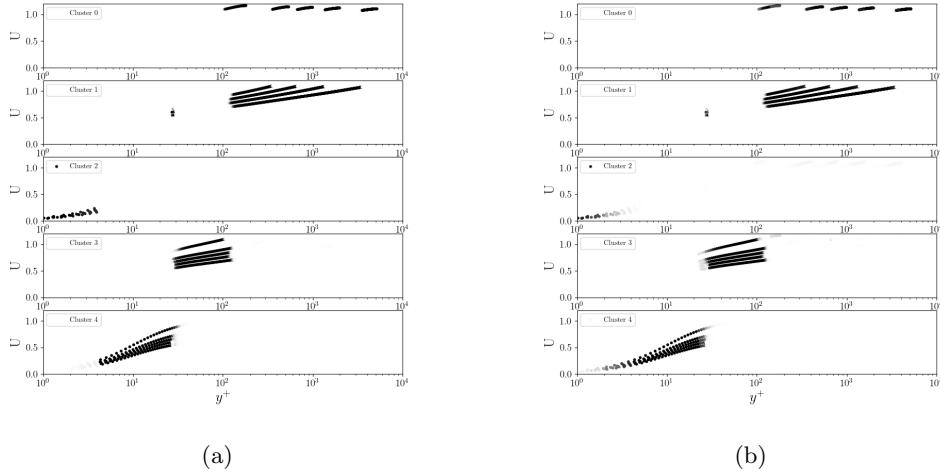


Fig. 1.1: Repeatability of the GMMs when initialized with k-means initialization(a), and random initialization (b). Points are colored darker if they were selected by more of the 100 models generated in the ensemble.

There is clearly some amount of randomness in GMMs (and likely unsupervised learning

more generally) that must be addressed in some manner in order to ensure repeatable results. This randomness can shift results of things such as hyperparameter selection when choosing an optimal number of clusters. In some cases this randomness can simply be resolved by expert domain knowledge. One of the most common methods to reduce the randomness is to initialize a model many times, and keep only the “best”. This begs the question though - is this “best” model actually giving us the “best” classifications? If we closely examine Fig. 1.1, we can see issues in the classifications from our models. For one, cluster 1, which is nominally the upper log-layer, includes a set of 5 extraneous points from the elbow between the buffer and log-layers. The mis-identification of the viscous sub-layer to buffer layer divide has also already been mentioned. Clearly there is some connection between these areas in the feature space such that the GMM continues to find these connections. If we want to produce both repeatable predictions, and good predictions we may need to look at other areas for ideas.

**1.3. Sensor Fusion via Bayesian Filtering.** While clearly unsupervised clustering approaches in general, and Gaussian mixture models in particular, are powerful and widely used for their ability to automatically categorize previously unseen data into categories which roughly correspond with human intuition, there are still several issues with the common approaches. For one, the generalizability and repeatability of the method rely heavily on heuristic approaches, which themselves can sometimes only apply in certain circumstances. Second, even if repeatability of the model is achieved, the use of heuristic approaches will often result in a “good but not great” result for the clustering (i.e. a local optimum, but not necessarily the best clustering). This can be seen in Fig. 1.1. In this work, we examine solutions to both of these issues by drawing inspiration from the fields of control theory and robotics.

In mobile robotics, a fundamental problem is that of “localization”, where a robot must estimate its current location in state space based only on its prior knowledge of its location, its internal model for estimating changes to location based on control inputs, and sensor measurements. The key idea here is realizing that repeated estimates from a single source will only reinforce the biases of that source; however by combining information from multiple sources one can achieve a much better prediction. In the field of supervised learning, there is an analogous concept called “boosting”, whereby several poorly performing classifiers are combined to produce a better result [10].

The seminal paper describing robot localization techniques was written by Fox et al. [3], and it describes several of the key methods developed for robotic applications, all based on the concept of a Bayesian filter. The key algorithm for this approach is comprised of two steps:

- First, a prediction is performed, which updates the estimate for the state of the system based on the prior state estimation and the internal dynamics model
- Second, a sensor measurement is taken, and then the prediction is updated based on the likelihood that the given sensor measurement would return given our current prediction

The report is structured as follows. In the introduction (section 1), an overview of the topic covered by this work is provided, as well as discussion of the approaches typically undertaken in the field. Section 2 lays out the mathematical foundation for the method proposed. Section 3 discusses a major potential hurdle for implementation, and the method by which it is addressed here. Section 4 presents some initial results from the approach in this paper. Finally, section 5 summarizes the work covered in this paper.

**2. Modeling Assumptions and Equations.** Let us begin by defining the state vector for each point using a variable  $\mathbf{x}^k$  which we will refer to as the “belief” value of point

$\mathbf{x}$ , where  $\mathbf{x}^k \in \mathbb{R}^c$  and  $c$  is the number of clusters we are using. In the mobile robotics localization context, the belief vector holds the probability distribution for the robots current state (location, velocity, etc). We will use it similarly - with the belief value of each data point representing our current best guess as to its actual clustering. Each element of  $\mathbf{x}^k$  represents the probability that point  $\mathbf{x}^k$  is in the respective cluster. The superscript  $k$  indicates that the state vector is for pseudotime  $k$ , where “pseudotime” in this context acts as a way to track the iterations of a numerical method as it is applied repeatedly to some data at a constant physical time until it achieves sufficient convergence. The belief value of a given point can then evolve through pseudotime as such:

$$\mathbf{x}^k = \mathbf{f}^k(\mathbf{x}^{k-1}) + \mathbf{w}^k \quad (2.1)$$

where  $\mathbf{f}^k$  is a model representing the evolution of the state space/belief vector through pseudotime, and  $\mathbf{w}_k$  is some known zero-mean noise in the measurement of the state. For our applications, the state noise term ( $\mathbf{w}^k$ ) will depend on the initialization of the clustering method and the actual distribution of data in the feature space. Since each vector  $\mathbf{x}^k$  is a belief distribution, we will also require that:

$$\|\mathbf{x}^k\|_1 = \sum_{i=1}^c x_i^k = 1 \quad (2.2)$$

so that the total probability distribution at any time always equals one (and the implicit assumption is that each value is non-negative). Additionally we can model the inputs from some sensor as:

$$\mathbf{z}^k = \mathbf{h}^k(\mathbf{x}^k) + \mathbf{v}^k \quad (2.3)$$

where  $\mathbf{z}^k$  are some sensor measurements collected at pseudotime  $k$ ,  $\mathbf{h}^k$  is the sensor model which determines sensor outputs as a function of the state, and  $\mathbf{v}^k$  is some known zero-mean noise in the sensor (which is uncorrelated with the noise in the state measurement, and the exact distribution of which is typically dependent on the specific sensor used).

A typical implementation of a Bayesian Filter would then follow the steps:

$$p(\mathbf{x}^k | \mathbf{z}^{1:k-1}) = \int p(\mathbf{x}^{k-1} | \mathbf{z}^{1:k-1}) p(\mathbf{x}^k | \mathbf{x}^{k-1}) d\mathbf{x}^{k-1} \quad (2.4)$$

$$p(\mathbf{x}^k | \mathbf{z}^{1:k}) = \frac{p(\mathbf{x}^k | \mathbf{z}^{1:k-1}) p(\mathbf{z}^k | \mathbf{x}^k)}{\int p(\mathbf{x}^k | \mathbf{z}^{1:k-1}) p(\mathbf{z}^k | \mathbf{x}^k) d\mathbf{x}^k} \quad (2.5)$$

where eqn. 2.4 is referred to as the prediction step, and eqn. 2.5 is the correction step. The individual terms of the filter are identified as follows: the term  $p(\mathbf{x}^{k-1} | \mathbf{z}^{1:k-1})$  is the prior belief,  $p(\mathbf{x}^k | \mathbf{x}^{k-1})$  is the transition likelihood,  $p(\mathbf{z}^k | \mathbf{x}^k)$  is the sensor evaluation, and the denominator of the correction step is used to renormalize the new belief.

To this point, we have only been concerned with the belief distributions of a single point in the flow field (for which we use the vector  $\mathbf{x}^k$  for the full belief distribution, or the vector element  $x_i^k$  if we are discussing the belief that the point belongs to cluster  $i$ ). However, when we are dealing with the belief values of a whole flow field, we will denote the flow

field belief matrix with upper case bold values  $\mathbf{X}^k$ , or for individual point  $i$  and cluster  $j$  we would write  $X_{ij}^k$ . Under most situations, the rows of matrix  $\mathbf{X}^k$  are unrelated, however they can be connected via the choice of an appropriate sensor model if desired.

How to model each of these terms will be discussed in the context of flow classification over the following subsections.

**2.1. Prior Belief.** Given a set of  $a$  data points, each with  $b$  features, we denote the features matrix  $\mathbf{A} \in \mathbb{R}^{axb}$ , and we can represent the unsupervised clustering of those points as:

$$\mathbf{X}^k = \mathbf{g}^k(\mathbf{A}) \quad (2.6)$$

where  $X_{ij}^k$  once again represents the probability of data point  $i$  belonging to cluster  $j$ , and  $\mathbf{g}^k(\cdot)$  is the unsupervised model (i.e. some Gaussian mixture model). Note that both the model and the beliefs estimate are denoted with the superscript  $k$  to denote that the output is dependent on the iteration (this is not yet technically accurate as so far the outputs do not relate to each other through time, but we will use it for now), but the features matrix is independent of pseudotime. As has been established previously in this paper the results of the clustering  $\mathbf{X}^k$  can be dependent on the random initialization of  $\mathbf{g}^k(\cdot)$ . As such, we can represent the unsupervised model as the combination of a “true” (or ideal) model, and some noise term. So the full equation will then be  $\mathbf{X}^k = \mathbf{g}^k(\mathbf{A}) = f(\mathbf{A}) + \mathbf{w}^k$ . The problem with this implementation is clear however - for the unsupervised model we have no way of knowing how much impact the noise term has on the final results for any individual prediction.

Additionally, by considering the model results in pseudotime we gain the advantage of being able to directly measure the residual of the belief through pseudotime as:

$$r_b = \frac{1}{a} \|\mathbf{X}^k - \mathbf{X}^{k-1}\|_{fro} \quad (2.7)$$

where the *fro*-norm denotes the “Frobenius” matrix norm. By exposing the residual in this manner, we can directly estimate the convergence of the model  $\mathbf{g}^k(\cdot)$  in the limit as  $k \rightarrow \infty$ . Under the current formulation, the model is unlikely to converge as each individual model is independent of all other models, so the residual will essentially be a measure of the noise. In the next subsection, a modification of our belief matrix will be discussed that will increase the utility of its residual.

**2.2. Transition Likelihood Model.** The transition likelihood model, given in eqn. 2.4 as  $p(\mathbf{x}^k | \mathbf{x}^{k-1})$ , is a model to predict the likelihood of going from the state in  $\mathbf{x}^{k-1}$  to all the possible states  $\mathbf{x}^k$ . In typical Bayesian filter applications, we are concerned with tracking a moving object under uncertain conditions, and the transition likelihood model reflects this. For our purposes primary application, we evaluate the change in pseudotime to an application which is static (noise is introduced mainly by the GMM initialization), and we want to find eventual convergence. As such, we evaluate the transition likelihood of each point in a cumulative average as:

$$\bar{\mathbf{x}}^k = \frac{(k-1)\bar{\mathbf{x}}^{k-1} + \mathbf{x}^k}{k} \quad (2.8)$$

where  $\bar{\mathbf{x}}^k$  is the running average likelihood of a point being in a given state, based on the transition likelihood. In this way the cumulative average belief of the entire field can be

denoted  $\bar{\mathbf{X}}^k$ , and the residual only considers the change in the cumulative average belief of the field.

**2.3. Sensor Evaluation.** Several different possible sensor models are described within this section, all of which approach the problem of relating data to GMM measurements. Other possible sensors surely exist; however this represents a subset of the sensors for which a strong analogy to the robot localization problem could be made, and thus were identified as candidates to be tested for flow classification.

**Uniform sensor:**

The first sensor implemented is a uniform belief sensor - it takes no inputs and simply returns as a measurement a uniform belief field across all clusters. In mathematical terms, the uniform belief distribution for a data point is simply:

$$\omega_i = \frac{1}{c} \quad (2.9)$$

where  $\omega_i$  is the discrete belief weight in cluster  $i$  of a given data point (recalling that  $c$  is the number of clusters).

The purpose of the uniform belief sensor is simply to demonstrate that a sensor can be implemented in a non-intrusive way.

**Buffer layer sensor:**

Another potential sensor type developed for the turbulent channel flow case is one we have named the “buffer layer sensor”. The buffer layer sensor incorporates specific knowledge about the flow to be studied in order to accurately estimate at least one of the clusters. In this way we incorporate some heuristic, but only in specific cases where the heuristic is known to apply to that case.

The buffer layer sensor used in this work does the following steps. A first pass is made over all points, and any point where the ratio of production of turbulent kinetic energy (TKE) to dissipation of TKE exceeds 1.1 is marked. This value to identify the buffer layer was chosen somewhat arbitrarily, however it tends to be fairly accurate that it will only occur in the buffer layer of a flow field (for simple wall bounded flows at least - obviously for more complex cases this situation can also occur in other regions). After each point satisfying this cutoff is identified, we set the weighted belief values of these points as follows:

$$\omega_i = \begin{cases} v, & \text{if } i = k \\ (1.0 - v)/c, & \text{if } i \neq k \end{cases} \quad (2.10)$$

where  $k$  is the cluster that we have identified as the buffer layer cluster. The sensor can be developed with any value of  $v$  in the range  $v \in (1.0/c, 1.0)$ . In practice, values of  $v$  result in very high rates of convergence in the identified region. The belief value of all points not identified as part of the buffer layer are set to uniform - in this way the sensor is only minimally intrusive to the model.

**3. Cluster to cluster correlations.** Correlating clusters from one Gaussian mixture model to another is a non-trivial problem. Even when two models are trained using the same feature set and data, due to the random initialization process the final clusters may end up different which can result in changes to the cluster feature means. The problem becomes even more challenging when correlating clusters from models trained on the same dataset, but a different feature space (this particular problem has not yet been attempted, but will be soon hopefully).

We will define the problem as follows for our circumstances: for a given trained GMM, the matrix of feature means for each cluster will be denoted  $\mathbf{M}_{ij} \in \mathbb{R}^{cxb}$ . So each row of that matrix  $M[i, :]$  corresponds to the vector of feature means for cluster  $i$ . Equivalently, the feature means from a second GMM could be denoted  $\mathbf{N}_{ij}$  (for now we will only consider the case with identical features and number of clusters, although this can be extended in the future). Mathematically, we can now write the problem of finding the optimal cluster correlation between two clusters as:

$$\mathbf{a}_{opt} = \arg \min_{\mathbf{a}} \sum_{i=1}^c \|\mathbf{M}[i, :] - \mathbf{N}[a_i, :]\|_2 \quad (3.1)$$

In words, we are looking to find the optimal ordering of the rows in  $\mathbf{N}$ , to minimize the sum of the euclidean norms of the difference between the two GMM feature mean rows. The optimal ordering vector  $\mathbf{a}$  is restricted here to possible index values of the rows of  $\mathbf{N}$ , or  $[0, c) \in \mathbb{Z}$ . Additionally, for our problem we require each cluster to correlate uniquely to only one other cluster, so repeated values are not permitted in the optimal ordering vector  $\mathbf{a}$ . To solve this, we perform a first pass where each value in  $\mathbf{a}$  is initially inserted based on the cluster in  $\mathbf{N}$  which minimizes the norm (allowing for repeated values). A second pass targets only the duplicate values in  $\mathbf{a}$ , and greedily replaces the repeated value which will cause the smallest increase in the total norm with an as of yet unused cluster until all the values of  $\mathbf{a}$  are unique.

This problem (as defined) is a integer optimization problem, and closely related to the “reorderable matrix problem” [6]. Although Mäkinen & Siirtola [6] don’t provide a rigorous proof that reorderable matrices are NP-complete, they show similarities between various sub-problems and known NP-complete problems. As such, there is likely no known general algorithm to solve this problem in polynomial time for all cases. For the moment, we will simply state that tests of the robustness of this approach for the channel flow case have found that the clusters are completely correctly correlated roughly 65% of the time, and only completely mis-correlated 5% of the time. The remaining cases are split between cases where most clusters are correctly correlated but one pair is swapped (mostly correct correlations), and cases where the GMM produces an anomalous clustering and this method still does well even though there’s no true result.

**4. Results.** For the plots in this section, we automatically identified the clusters by cluster mean  $y^+$  values, and plotted them from lowest to highest. This enforces a consistent ordering between clusters in the plots.

**4.1. No sensor/uniform sensor.** The results with the uniform sensor tend to be identical to the results from simply averaging many sensor predictions. This is working as intended for a sensor designed to not impact the results. Typically we can achieve fairly consistent predictions after 100 iterations, but there is a very small rate when things are noticeably different, so for this section cases are run for around 300 iterations. The results from this sensor are shown in Fig. 4.1. We see the same general issues we identified previously (with less averaging) - the model performs well the majority of the time, but occasionally individual runs will get different clusters (partially due to misclassification, and partially due to true changes in clusters), so the cumulative averaging process smooths this out. Although the results obtained here (with essentially just an ensemble average of model predictions) are not significantly better than in Fig. 1.1, they are also not significantly worse. Most notably, these results are produced without the reliance on any heuristic to identify a “best” result, and using a random initialization of the models.

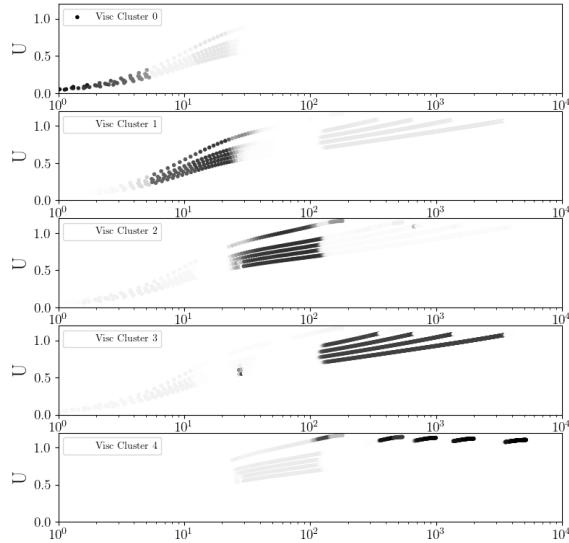


Fig. 4.1: Uniform sensor results (essentially the same as no sensor, or previous averages) after 300 iterations

**4.2. Buffer Layer Sensor.** If the buffer layer sensor is well tuned (with appropriate values to identify the buffer layer), the cluster of interest converges almost immediately. Results are shown in Fig. 4.2. It can be seen that the results are fairly similar to the results with the uniform sensor case (Fig. 4.1), with the major exception that the turbulent buffer layer (viscous cluster 1 in the plot) is classified with essentially no uncertainty. That is to say, the points of interest in the buffer layer appear only in the buffer layer, and there is no mis-classification or overlap for these points.

**5. Conclusions.** In this work, methods to improve the robustness and repeatability of unsupervised learning approaches are presented. Work began by surveying the field of unsupervised learning approaches as they are currently being used for turbulent flow problems. Several problems were identified with methods that are commonly being used in the current literature. A methodology for remedying several of these issues is then developed, based on the approach used in mobile robot localization which applies Bayesian filters. Several different types of sensors are discussed and developed in the context of flow fields. Results are presented for the unsupervised clustering of a channel flow.

Future work will explore several avenues. Additional types of sensors are discussed, which could improve the methods as presented to date. Additionally, testing of this approach on cases beyond simple channel flow will be important to validating the utility of the developed method. Finally, future work could explore extensions to this method, to allow it to work on time-dependent data, to help in other sections of hyperparameter selection, or even to function as an “online” model that is able to classify a new data point without resorting to re-training.

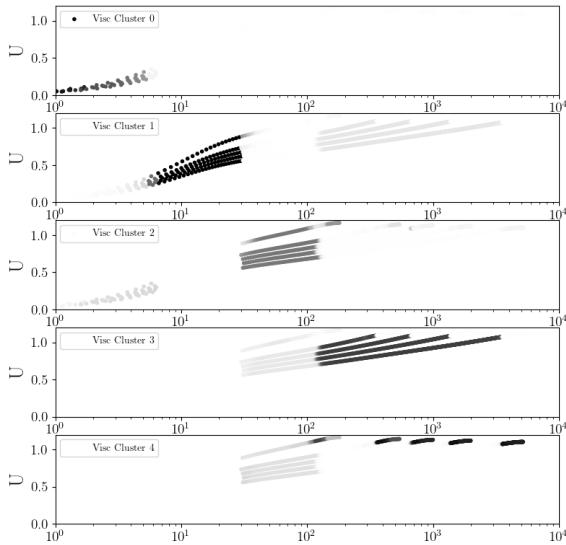


Fig. 4.2: Results with a buffer layer sensor after 300 iterations

### References.

- [1] M. BARONE, J. RAY, AND S. DOMINO, *Feature selection, clustering, and prototype placement for turbulence datasets*, AIAA Journal, 60 (2022), pp. 1332–1346.
- [2] J. L. CALLAHAM, J. V. KOCH, B. W. BRUNTON, J. N. KUTZ, AND S. L. BRUNTON, *Learning dominant physical processes with data-driven balance models*, Nature communications, 12 (2021), p. 1016.
- [3] V. FOX, J. HIGHTOWER, L. LIAO, D. SCHULZ, AND G. BORRIELLO, *Bayesian filtering for location estimation*, IEEE pervasive computing, 2 (2003), pp. 24–33.
- [4] M. LEE AND R. D. MOSER, *Direct numerical simulation of turbulent channel flow up to*, Journal of fluid mechanics, 774 (2015), pp. 395–415.
- [5] LYNE, JOHN, *Unsupervised learning for coherent structure identification in turbulent channel flow*, Master's thesis, 2023.
- [6] E. MÄKINEN AND H. SIIRTOLA, *Reordering the reorderable matrix as an algorithmic problem*, in International Conference on Theory and Application of Diagrams, Springer, 2000, pp. 453–468.
- [7] K.-E. OTMANI, G. NTOUKAS, O. A. MARIÑO, AND E. FERRER, *Toward a robust detection of viscous and turbulent flow regions using unsupervised machine learning*, Physics of Fluids, 35 (2023).
- [8] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT, AND E. DUCHESNAY, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.
- [9] E. SAETTA AND R. TOGNACCINI, *Identification of flowfield regions by machine learn-*

- ing, AIAA Journal, 61 (2023), pp. 1503–1518.
- [10] R. E. SCHAPIRE, *The strength of weak learnability*, Machine learning, 5 (1990), pp. 197–227.
- [11] Z. WU, J. LEE, C. MENEVEAU, AND T. ZAKI, *Application of a self-organizing map to identify the turbulent-boundary-layer interface in a transitional flow*, Physical review fluids, 4 (2019), p. 023902.

## MULTI-FIDELITY SURROGATE MODELING WITH FOURIER FEATURES NETWORKS

OWEN DAVIS\*, GIANLUCA GERACI†, AND MOHAMMAD MOTAMED‡

**Abstract.** In studying quantities of interest related to real world physical systems we often make use of computationally complex mathematical models, and in recent years, there has been growing interest in understanding how to optimally leverage an ensemble of mathematical models that all approximate the quantity of interest, but at varying cost and accuracy. That is, we assume have a high-fidelity model that is expensive to evaluate but very accurate along with an ensemble of lower-fidelity models that are cheaper to compute, less accurate, and correlated with the high-fidelity model. In this work, we assume that these various mathematical models give rise to a set of multi-fidelity training data comprised of a small number of high-fidelity data and a much larger number of lower-fidelity data, and we leverage this data to train a neural network based surrogate model that is both cheap to evaluate and as accurate as the high-fidelity model. Our novel contribution involves the development of a multi-fidelity modeling strategy that leverages Fourier features networks, a type of feedforward neural network that uses complex exponential activation functions. We motivate the use of this variety of neural network in a multi-fidelity context noting that they can be highly interpretable, that their training cost has the potential to scale favorably with problem dimension, and that they admit useful theoretical guarantees that give plausible estimates on generalization error in a sparse training data regime.

**1. Introduction.** In many science and engineering applications we leverage computationally complex mathematical models to study and make predictions about a quantity of interest (QoI) related to a real world physical system. The predictions often depend on user specified or experimentally determined parameters with varying levels of uncertainty. Estimating optimal values for these parameters and/or quantifying how the uncertainty propagates to the model predictions is a computationally expensive outer-loop task that requires many evaluations of the underlying mathematical model.

In many real world applications evaluations of the true QoI are so expensive or difficult to obtain that the use of single fidelity approaches is unfeasible; however, at least in principle, one could resort to multi-fidelity methods. In this work, we assume that we have an expensive and accurate high-fidelity approximation of the QoI (which may be the true QoI) along with an ensemble of cheaper to compute, but less accurate lower-fidelity approximations that are correlated with the high-fidelity QoI. The main idea is to leverage the relationship between approximations of different fidelity, shifting the computational burden from the expensive high-fidelity approximation to the cheaper lower-fidelity approximations.

Given this multi-fidelity information, existing strategies for completing this parameter estimation or uncertainty quantification (UQ) task fall into two general categories: (i) Monte Carlo (MC) sampling and (ii) surrogate modeling. In the sampling-based approaches the various approximations of the QoI are strategically evaluated over many different parameter values, and then statistics are computed from the resulting data set. In general, the majority of the evaluations are conducted on the cheaper lower-fidelity approximations, and the main idea is to leverage the low-fidelity approximations to reduce the variance of the high-fidelity estimator. Some examples of multi-fidelity sampling based approaches include multi-level MC [9], multi-index MC [14], multi-fidelity MC [31], and approximate control variates [11]. In the surrogate modeling approach, the approximations of varying fidelity are strategically evaluated over the desired parameter range to generate a multi-fidelity data set, which is used to train a cheap-to-evaluate surrogate model mapping from the uncertain parameter(s) to the high-fidelity QoI. Generally, the training set is composed of a small number of high-

---

\*University of New Mexico, Sandia National Laboratories, ondavis@sandia.gov

†Sandia National Laboratories, ggeraci@sandia.gov

‡University of New Mexico, motamed@unm.edu

fidelity data and a much larger set of lower-fidelity data. Once trained, statistics about the QoI can be extracted, in some cases directly from the surrogate model, or in all cases by using the cheap-to-evaluate surrogate in conjunction with MC sampling. Some examples include multi-fidelity Gaussian processes (MF-GP) [19, 7], multi-fidelity polynomial chaos expansions (MF-PCE) [28], co-kriging [33], and multi-fidelity Bayesian Networks [23].

The computational cost of surrogate modeling is generally dominated by the cost of generating training data and fitting the surrogate model to that data. Hence for surrogate modeling to be cost-effective we must be able to train the surrogate to approximate the QOI to the desired tolerance on much less training data than what would be required to complete the outer-loop task with MC sampling directly. In general, for QoIs of low to moderate dimension and moderate to high regularity conventional strategies are cost effective. However, for very high-dimensional and very-low-regularity QoIs conventional surrogate modeling techniques can become very computationally expensive. Hence developing cost effective surrogate models in this problem regime is of particular interest to the scientific community.

In recent years deep neural networks have been increasingly successful in approximating complex high-dimensional target quantities [10], and because of this many speculate that they could be successfully used as surrogate models for these low-regularity high-dimensional QoIs. However, deep neural networks generally require lots of training data, and so the primary challenge is leveraging a neural network to learn the mapping between the lower- and high-fidelity approximations on sparse high-fidelity training data.

There is considerable past work on this problem, most of which focuses on feedforward supervised learning in the bi-fidelity case where we have one high-fidelity approximation and one low-fidelity approximation. In [6] the low-fidelity data is assumed to be linearly correlated with the high-fidelity data and this correlation is elucidated through a learned multiplicative and additive correction on the low-fidelity model. This method is limited by the assumption that the correlation between models is linear, which cannot be expected in general. Several works in the literature address this concern. In [32] the multiplicative correction term is replaced by a learned, possibly non-linear, function of the low-fidelity data, and in [25, 13] the additive correction is further absorbed into this unknown non-linear function, and this general mapping is learned by a single neural network. In [24, 23, 13] this general non-linear function is split into linear and non-linear components which are learned by separate neural networks. The idea is lean on the linear component which is generally easier to learn on sparse data. In [4] the authors show empirically that if the discrepancy between the high and low-fidelity model is small in uniform norm relative to the high-fidelity model, then it is advantageous to reformulate the correlation between data types as a possibly non-linear residual function that targets this discrepancy. In [13] two additional methods are proposed. In the first, the low-fidelity model outputs are considered latent variables in the neural network that learns from the sparse high-fidelity data and are inserted in an intermediate weight layer of the network. The second proposed method takes advantage of the connection between neural networks and Gaussian processes [12, 21, 27]; this method is named GPmimic and aims to replicate the action of a Gaussian process without incurring the same prohibitive computational cost for high-dimensional problems. In [22] the authors use a transfer learning approach to train a surrogate from multifidelity data. A deep neural network is initially trained on a data set comprised of mostly low-fidelity data. This pre-trained network is then fine tuned on the sparse high-fidelity data. To ensure transfer learning in the fine tuning step the parameters in the first several layers of the network are frozen, and only the trainable parameters in the last couple layers of the network are allowed to vary.

There are three high-level drawbacks common to almost all existing neural network

based surrogate modeling strategies. The first is the lack of theoretical guarantees. Rigorous results on neural network approximation are generally limited to universal approximation theorems, which only show the existence of desired neural networks in the limit of infinite training data [34, 5, 3, 26, 2]. While universal approximation theorems are necessary and important in justifying the use of neural networks, they rarely provide direct insight into their performance on sparse training data.

Second, neural networks have notable predictive uncertainty with respect to many factors including the number and distribution of training data and the hyper-parameters used in the stochastic gradient based optimization algorithms generally used to train them. There have been several works that attempt to address this issue including [35, 39], which work to develop a unified framework for UQ in machine learning. One of the primary challenges is that most of the uncertain parameters are chosen before network training, so in principle, completing the UQ task requires training the neural network numerous times over many different choices of training data and hyper-parameters. For deep neural networks, which are generally required for complex high-dimensional target functions, this can become computationally prohibitive. One strategy that has been used to alleviate this computational cost is MC dropout [8] in which the neurons of the network are randomly switched off during training and/or testing with some chosen probability. During training, applying dropout is a form of regularization that can prevent over-fitting whereas during evaluation, MC dropout has been shown to be a form of Bayesian approximation from which uncertainty estimates can be derived. Although MC dropout is a powerful tool when it comes to UQ for machine learning, it has its own disadvantages. Generally, the obtained uncertainty estimate is highly dependent on the probability with which neurons are dropped, which is a chosen hyper-parameter. Furthermore, it is observed in [29] that to get reasonable single predictions from the neural network (as opposed to only a reasonable mean prediction), dropout needs to be used during training and testing. This is not ideal since using dropout during training is not benign, and can be intrusive to other network hyper-parameters. For example, using dropout during training leads to a highly oscillatory training loss curve, which excludes the possibility to develop a robust adaptive learning rate based on monitoring training loss.

The final high-level drawback is the lack of interpretability in neural networks. This exacerbates the issue with predictive uncertainty, since at the end of training it is usually impossible to intuitively understand the distribution of network parameters. Moreover, the lack of interpretability makes developing an informed initialization for network parameters specific to a given target function somewhat intractable.

In order to address these drawbacks we choose to develop a multi-fidelity modeling strategy based on Fourier features networks introduced for single-fidelity training data in [16, 17]. We develop these ideas further in Section 3, but note here that Fourier features networks are highly interpretable in a multi-fidelity context on account of using a Fourier approximation basis, have theoretical guarantees that give plausible estimates on network generalization error in a sparse data regime, and admit a flexible Markov Chain Monte Carlo based training procedure that has the potential to scale favorably with problem dimension. In Section 2 we provide a rigorous description of this surrogate modeling task on multi-fidelity training data, while in Section 3 we introduce single-fidelity Fourier features networks and motivate them for multi-fidelity problems. In Section 4 we develop our method for leveraging Fourier features with multi-fidelity training data and, in Section 5, we implement the developing multi-fidelity Fourier features method on a benchmark problem from the literature, and use the results to guide a discussion on improvements to the developed method and other future work in Section 6.

**2. Mathematical problem description.** Let  $\mathbf{x} \in X \subset \mathbb{R}^n$  be a space-time variable,  $\boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d$  be a parameter vector, and  $\mathbf{u} : (\mathbf{x}, \boldsymbol{\theta}) \in X \times \Theta \mapsto \mathbf{u}(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^m$  the solution to a system of parametric ODEs/PDEs, parameterized by  $\boldsymbol{\theta} \in \Theta$ . Assume our QoI is a function  $Q : \Theta \rightarrow \mathbb{R}$  defined as a functional  $\mathcal{F}_Q$  acting on  $\mathbf{u}$ ,

$$Q : \boldsymbol{\theta} \in \Theta \subset \mathbb{R}^d \mapsto Q(\boldsymbol{\theta}) \in \mathbb{R}, \quad Q(\boldsymbol{\theta}) = \mathcal{F}_Q(\mathbf{u}(\mathbf{x}; \boldsymbol{\theta})),$$

and satisfying

$$\|Q\|_{L^\infty(\Theta)} < \|\hat{Q}\|_{L^1(\mathbb{R}^d)} < \infty,$$

where  $\hat{Q}$  is the Fourier transform of  $Q$ . Such a requirement is not very restrictive. The right inequality asserts that  $Q$  has a well defined Fourier transform and, in examining the left inequality, we note that the corresponding non-strict inequality  $\|Q\|_{L^\infty(\Theta)} \leq \|\hat{Q}\|_{L^1(\mathbb{R}^d)}$  is always true following from the triangle inequality combined with the definition of the inverse Fourier transform,

$$\|Q\|_{L^\infty(\Theta)} = \left\| \int_{\mathbb{R}^d} \hat{Q}(\boldsymbol{\omega}) e^{i2\pi\boldsymbol{\omega} \cdot \boldsymbol{\theta}} d\boldsymbol{\omega} \right\|_{L^\infty(\Theta)} \leq \int_{\mathbb{R}^d} |\hat{Q}(\boldsymbol{\omega})| d\boldsymbol{\omega} = \|\hat{Q}\|_{L^1(\mathbb{R}^d)}.$$

Furthermore, the corresponding strict inequality is empirically satisfied for most target functions that are high-dimensional, defined outside the unit hypercube, and/or have oscillatory or irregular characteristics.

Now, suppose we need numerous evaluations of  $Q(\boldsymbol{\theta})$  over many different values  $\boldsymbol{\theta}$  as in an uncertainty quantification or parameter estimation task. Because evaluating  $Q$  requires solving the system of differential equations, such a task is computationally prohibitive. Hence we seek to construct a cheap-to-evaluate surrogate model satisfying the accuracy constraint

$$\|Q - \tilde{Q}\|_{L^2(\Theta)} < \varepsilon_{TOL}, \tag{2.1}$$

for some chosen tolerance  $\varepsilon_{TOL} < 1/2$ .

Assume further that we have an ensemble of  $M$  approximations of  $Q$  of varying cost and accuracy. We have an expensive-to-compute high-fidelity approximation  $Q_{HF}$  satisfying  $\|Q - Q_{HF}\|_{L^2(\Theta)} < \varepsilon_{TOL}$  along with an ensemble of  $M-1$  cheaper to compute lower-fidelity approximations  $\{Q_{LF_j}\}_{j=1}^{M-1}$  correlated with  $Q_{HF}$ . We make no limiting assumptions about the source of these low-fidelity approximations. For example, they could result from solving the underlying differential equations on a discretizations of varying refinement, using a reduced order model, simplifying the underlying physics of the model, or taking an asymptotic approximation. Moreover, the lower-fidelity approximations are not assumed to be ordered by their approximation quality of  $Q_{HF}$  for a given cost. However, for this study we do assume that all models have the same parameterization (the same input and output space); the generalization to the common case of models with dissimilar parameterization is left for future studies, and it could be obtained following similar multi-fidelity approaches like [38].

We assume that the varying approximations give rise to multi-fidelity training data of the following structure,

- High-fidelity data:  $\{(\boldsymbol{\theta}^{(m)}, Q_{HF}(\boldsymbol{\theta}^{(m)}))\}_{m=1}^{N_{HF}}$
- Lower-fidelity data:  $\{(\boldsymbol{\theta}^{(m)}, Q_{LF_j}(\boldsymbol{\theta}^{(m)}))\}_{m=1}^{N_{LF_j}}, j = 1, \dots, M-1$ ,

where the high-fidelity data is sparse relative to the lower-fidelity data ( $N_{HF} \ll N_{LF_j}$  for all  $j = 1, \dots, M-1$ ). Our goal is then to train a neural network based surrogate model  $\tilde{Q}$

on this multi-fidelity training data that satisfies (2.1). There are many different varieties of neural networks that could be used for this task, but in this work we consider using Fourier features networks, which are introduced and motivated for multi-fidelity problems in the ensuing sections.

**3. Fourier features networks.** Following [16, 17], a Fourier features network of width  $K$  and depth  $L$  is a feed-forward residual network [15] with  $L$  hidden blocks, where each block consists of two layers. The first block has  $K$  neurons in the first layer and 1 neurons in the second and all subsequent blocks have  $2K$  neurons in the first layer and 2 neurons in the second. These networks further are endowed with the Fourier features activation function  $\sigma(\boldsymbol{\theta}, \boldsymbol{\omega}) = e^{i\boldsymbol{\theta} \cdot \boldsymbol{\omega}}$ . We represent such a network by the collection of its trainable parameters  $\Phi = (\mathbf{b}, \mathbf{b}', \boldsymbol{\omega}, \boldsymbol{\omega}')$  that are indeed amplitudes  $\mathbf{b}, \mathbf{b}' \in \mathbb{C}^K$  and frequencies  $\boldsymbol{\omega} \in \mathbb{R}_{\geq 0}^{Kd}$  and  $\boldsymbol{\omega}' \in \mathbb{R}_{\geq 0}^K$ . The network realizes the function  $f_{\Phi}(\boldsymbol{\theta}) = z_L(\boldsymbol{\theta})$  where  $z_L$  results from the recurrence:

$$z_1(\boldsymbol{\theta}) = \Re \left( \sum_{j=1}^K b_{1j} \sigma(\boldsymbol{\omega}_{1j}, \boldsymbol{\theta}) \right), \quad (\text{Approximates } Q)$$

$$z_{\ell}(\boldsymbol{\theta}) = z_{\ell-1}(\boldsymbol{\theta}) + \underbrace{\Re \left( \sum_{j=1}^K b_{\ell j} \sigma(\boldsymbol{\omega}_{\ell j}, \boldsymbol{\theta}) \right) + \Re \left( \sum_{j=1}^K b'_{\ell j} \sigma(\boldsymbol{\omega}'_{\ell j}, z_{\ell-1}) \right)}_{(\text{Approximates } Q - z_{\ell-1})}, \quad \ell = 2, \dots, L.$$

For clarity, an example of a Fourier features residual network with  $(K, L) = (2, 3)$  is pictured in Figure 3.1. Each step of the recurrence sums the approximation of the QoI from the

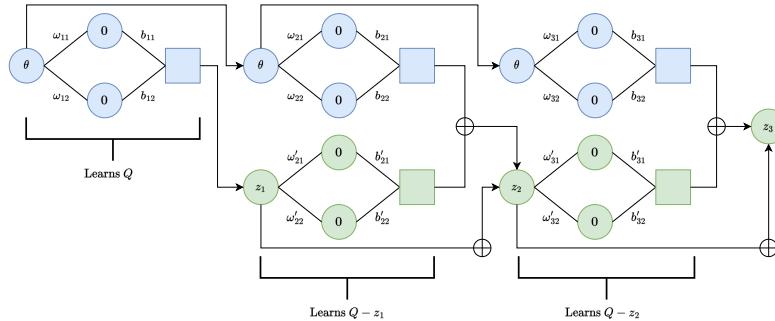


FIG. 3.1. A Fourier features residual network with  $(K, L) = (2, 3)$

previous layer and a learned correction. The correction is decomposed into a standard term (blue) that depends on the native input  $\boldsymbol{\theta}$  and a depth term (green) that depends on the output of the previous layer  $z_{\ell-1}$ . The latter term is the means by which this variety of network takes advantage of the compositional power associated with deep neural networks.

There are three notable attributes of Fourier features networks that to the best of the authors' knowledge distinguish them from other types of neural networks in the literature and make them an attractive choice for multi-fidelity problems.

The first is their interpretability. The relationship between high- and lower-fidelity models is often described as the high-fidelity model containing additional high-frequency information not present in the lower-fidelity models. Such a description is highly interpretable

in a polynomial context, where higher frequency information corresponds to higher order terms, and in a Fourier context, where we explicitly work in the frequency domain. This work focuses on developing neural network based surrogate modeling strategies, and *for neural networks a non-polynomial activation function (approximation basis) is required to preserve density of neural network approximates in even simple function spaces such as the space of continuous functions* [34]. Hence for neural network surrogate modeling with multi-fidelity information the Fourier features activation provides a unique level of interpretability.

Fourier features residual networks also offer advantageous flexibility in their training procedure. While most neural networks are trained using a stochastic gradient descent based algorithm, Fourier features networks can benefit from a layer by layer training algorithm [17]; at each layer  $\ell = 1, \dots, L$  the frequencies  $\omega_\ell = \omega_{\ell 1}, \dots, \omega_{\ell k}$  are assumed to be i.i.d. random variables with common distribution  $p_\ell(\omega)$ . Then the trainable parameters result by solving the nested optimization problem

$$\min_{p_\ell} \mathbb{E}_{\omega_\ell} \left[ \min_{\mathbf{b}_\ell, \mathbf{b}'_\ell} \left\{ \mathbb{E}_{\theta} [|f_\Phi(\theta) - Q(\theta)|^2] + \lambda_\ell \|\mathbf{b}_\ell, \mathbf{b}'_\ell\|_2^2 \right\} \right]. \quad (3.1)$$

We direct the reader to [17] for a derivation and discussion of the nested optimization problem (3.1). The outer minimization seeks an optimal distribution of frequencies  $p_\ell^*(\omega_\ell)$ . Once found, these frequencies are fixed, and then the inner convex optimization problem is solved for the amplitudes  $\mathbf{b}, \mathbf{b}'$ . Note here that we have neglected to mention the frequencies  $\omega'$  associated with the depth term. In [17] the authors choose to sample these frequencies just once from a standard normal distribution and fix them throughout training. The impact of such a choice will be the subject of future work, but at present we aim for consistency with the current single-fidelity Fourier features method in the literature and retain this procedure. Additionally, when numerically computing the solution to (3.1), the expectations are replaced by the empirical mean over the available data samples. Using the nested optimization strategy (3.1) is motivated by two observations. First, there exists an analytic optimal distribution of frequencies  $p_\ell^* = |\hat{Q} - \hat{z}_{\ell-1}| / \|\hat{Q} - \hat{z}_{\ell-1}\|_{L^1}$  at each layer [16], and second, the inner optimization is convex, so it can be solved robustly via several different solution techniques depending on the ratio  $N/K$  where  $N$  is the number of training data and  $K$  is the width of the network (number of basis terms). For example, when  $N/K \gg 1$  a standard least squares optimization is sufficient. While an analytic optimal distribution of frequencies is available at each layer, it usually cannot be efficiently computed, and so the authors in [16] instead propose the *Adaptive Fourier features algorithm*, which uses an adaptive Metropolis-Hastings algorithm to approximately sample from the optimal distribution. The pseudo-code for this algorithm is included in Appendix A.

It was shown in [37, 18] that, although the standard Metropolis-Hastings algorithm can scale more favorably with problem dimension than standard MC approaches, it still suffers from the curse of dimensionality. Nevertheless, there is a considerable literature, e.g., see [37, 40, 1], that develops Metropolis-Hastings type algorithms specifically tailored to sampling from high-dimensional distributions; we anticipate that these existing methods could be used together with the current Adaptive Fourier features algorithm such that it more effectively combats the curse of dimensionality. We leave this for future work, and focus here instead on leveraging the existing Adaptive Fourier features algorithm in a multi-fidelity context. Additionally, note that in this algorithm the Metropolis acceptance criteria for the frequencies is based on the computed amplitudes. This will be important as we develop our multi-fidelity Fourier features method in Section 4.

The final and perhaps most important aspect of Fourier features residual networks are the following theoretical error bounds, which to the best of the authors' knowledge are not currently available for any other type of neural network.

**THEOREM 3.1** (Generalization Error in One Layer Fourier Features Network [16]). *Consider a target function  $Q$  and denote by  $\hat{Q}$  its Fourier transform. Then, there exists a one layer Fourier features network  $f_{\Phi}$  of width  $K$ , and a constant  $C \in \mathbb{R}$  such that*

$$\|Q - f_{\Phi}\|_{L^2(\Theta)}^2 \leq C \frac{\|\hat{Q}\|_{L^1(\mathbb{R}^d)}^2}{K}. \quad (3.2)$$

**THEOREM 3.2** (Generalization Error in Deep Fourier Features Residual Network [17]). *Consider a target function  $Q$  and denote by  $\hat{Q}$  its Fourier transform. Then there exists a Fourier features residual network  $f_{\Phi}$  of depth  $L$ , width  $K = \mathcal{O}(L^2)$ , and a constant  $C \in \mathbb{R}$  such that*

$$\|Q - f_{\Phi}\|_{L^2(\Theta)}^2 \leq C \frac{\|Q\|_{L^\infty(\Theta)}^2}{KL} \left( 1 + \ln \left( \frac{\|\hat{Q}\|_{L^1(\mathbb{R}^d)}}{\|Q\|_{L^\infty(\Theta)}} \right) \right)^2. \quad (3.3)$$

We remark that the network aspect ratio requirement  $K = \mathcal{O}(L^2)$  is needed for the proof of Theorem 3.2 in [17], but empirically we have observed that the condition can be relaxed. Overall, Theorems 3.1 and 3.2 indicate that, if the size of the target function can be approximated from the training data measured as  $\|\hat{Q}\|_{L^1(\mathbb{R}^d)}$  in the case of 1-layer networks or as  $\|Q\|_{L^\infty(\Theta)}$  in the case of deep networks, then for a given error tolerance we can estimate the necessary complexity of the approximating network. Then, taking the plausible assumption that the complexity of a neural network is at least weakly related to the amount of training data needed to power it, for example taking the number of trainable parameters proportional to the number of training data, we can view these error bounds as providing a heuristic on the amount of training data necessary to achieve a given tolerance. This is very useful in a multi-fidelity context where one of the central issues is training data sparsity. We also note here the bound in Theorem 3.2 for deep networks is generally smaller than the bound in Theorem 3.1 for one layer networks. Furthermore, the advantage for deep networks is particularly large in the case that  $\|Q\|_{L^\infty(\Theta)} \ll \|\hat{Q}\|_{L^1(\mathbb{R}^d)}$ . Now that single fidelity Fourier features networks have been introduced and motivated, we move to adapting them to a multi-fidelity supervised learning problem.

**4. Bi-fidelity Fourier features networks.** For simplicity, we restrict our focus to the bi-fidelity case where we have one high-fidelity approximation  $Q_{HF}$  and one low-fidelity approximation  $Q_{LF}$ ; these approximations give rise to  $N_{HF}$  high-fidelity and  $N_{LF}$  low-fidelity data, and we assume  $N_{HF} \ll N_{LF}$ . Furthermore, in this preliminary work, we restrict our focus to 1-layer networks.

There are two complementary goals that guide the development of our bi-fidelity Fourier features method.

- (i) The low-fidelity approximation should be used to reduce the size of the target function learned on sparse high-fidelity data.
- (ii) The cheap low-fidelity approximation should be used to discover some portion of the frequency information (basis terms) necessary to represent the high-fidelity QoI.

Goal (i) is rigorously supported by Theorem 3.1. If the size of the target quantity can be made small, then it can be approximated to a given tolerance by a network of lower complexity, which can plausibly be learned on sparser training data. In a bi-fidelity context, this size reduction can often be accomplished by using a discrepancy model where we learn the difference  $Q_{HF} - Q_{LF}$ , and then recover the high-fidelity quantity by adding back

the low-fidelity contribution. Discrepancy modeling has long been integrated into surrogate modeling strategies, including those based on neural networks. For example, in [4] it is empirically observed for ReLU networks that learning a residual function that targets the discrepancy between models yields a large performance benefit when  $Q_{HF} - Q_{LF}$  is small relative to  $Q_{HF}$ , and Fourier features networks make this empirical observation mathematically rigorous by way of Theorems 3.1 and 3.2. Goal (ii) can be accomplished in many different ways, some of which will be discussed in Section 6, but for this preliminary work we assume that the low- and high-fidelity approximations have some amount of strictly overlapping frequency information. We then leverage the relatively abundant low-fidelity training data to extract this overlapping frequency information thereby relying less heavily on the sparse high-fidelity training data.

Given these two overarching goals the bi-fidelity method proceeds as follows.

**Step 1:** Approximate the low-fidelity model by a 1-layer Fourier features network

$$Q_{LF} \approx z_{LF}(\boldsymbol{\theta}) = \Re \underbrace{\left( \sum_{j=1}^{K_{LF}} b_j^{LF} \sigma(\omega_j^{LF}, \boldsymbol{\theta}) \right)}_{\text{Approximates } Q_{LF}}, \quad (4.1)$$

where  $K_{LF} \in \mathbb{N}$  is the network width, and the parameters  $b_j^{LF}$  and  $\omega_j^{LF}$  are tuned by the adaptive Fourier features algorithm.

**Step 2:** Assume that the high-fidelity model can be well approximated by a Fourier features network of one of the following forms

$$Q_{HF}(\boldsymbol{\theta}) \approx z_{LF}(\boldsymbol{\theta}) + \Re \underbrace{\left( \sum_{j=1}^{K_{HF}} b_j^{HF} \sigma(\omega_j^{HF}, \boldsymbol{\theta}) \right)}_{\text{Approximates } Q_{HF} - z_{LF}} + \Re \left( \sum_{j=1}^{K_{LF}} b_j^{corr} \sigma(\omega_j^{LF}, \boldsymbol{\theta}) \right) \quad (4.2)$$

$$Q_{HF}(\boldsymbol{\theta}) \approx \Re \underbrace{\left( \sum_{j=1}^{K_{HF}} b_j^{HF} \sigma(\omega_j^{HF}, \boldsymbol{\theta}) \right)}_{\text{Approximates } Q_{HF}} + \Re \left( \sum_{j=1}^{K_{LF}} b_j^{corr} \sigma(\omega_j^{LF}, \boldsymbol{\theta}) \right). \quad (4.3)$$

In both proposed forms of the high-fidelity approximation the general strategy is the same. The low-fidelity frequencies discovered in step 1 are assumed to also be present in the high-fidelity model, but with corrected corresponding amplitudes  $\mathbf{b}^{corr}$  that need to be learned. Additionally, we assume that there are  $K_{HF}$  high-fidelity frequencies  $\omega^{HF}$  not present in the low-fidelity model that need to be discovered on the sparse high-fidelity data along with their corresponding amplitudes  $\mathbf{b}^{HF}$ . The only difference between the two proposed forms of the high-fidelity model is that in Equation (4.3) the trainable parameters are tuned to approximate  $Q_{HF}$  directly, whereas in Equation (4.2) we use a discrepancy model where the trainable parameters are tuned to approximate  $Q_{HF} - z_{LF}$ .

Unlike the low-fidelity frequencies, the high-fidelity frequencies cannot be accurately obtained by the adaptive Fourier features algorithm. The issue is the sparse high-fidelity data. Recall that the Metropolis-Hastings algorithm used to approximately sample from the optimal frequency distribution has an acceptance criteria based on the computed amplitudes. On sparse high-fidelity data these amplitudes are often inaccurate leading to errors in the

sampled frequencies. As an alternative, we simultaneously learn the high-fidelity frequencies, their corresponding amplitudes, and the corrected amplitudes for the low-fidelity frequencies using a 1-layer Fourier features network built in Pytorch [30] and trained using the Adam optimizer [20]. An example network diagram is included in Figure 4.1.

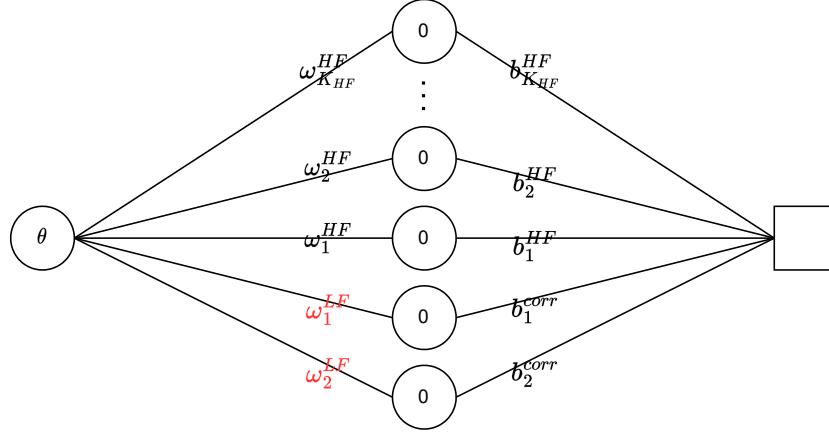


FIG. 4.1. Example network used to tune high-fidelity trainable parameters with low-fidelity informed initialization. Red parameters are fixed, and black parameters are trainable.

It is well known that the effectiveness of stochastic optimization algorithms, such as the Adam optimizer used to train this network, are highly dependent on their initialization [36]. Leveraging the interpretability of Fourier features networks in a multi-fidelity context, we develop the following low-fidelity informed initialization. First, in line with Goal (ii), the low-fidelity frequencies learned in Step 1, shown in red in Figure 4.1, are fixed as input weights in the network and are not updated during training. The remaining  $K_{HF}$  input weights are indeed the additional high-fidelity frequencies that need to be learned by the network. Motivated by the assumption that the high-fidelity model contains important frequency information that is larger in magnitude than any frequency information contained in the low-fidelity model, the additional high-fidelity frequencies are initialized as a random uniform distribution in the range  $[0, c\omega_{max}^{LF}]$  where  $c \geq 0$  is a chosen hyper-parameter and  $\omega_{max}^{LF} := \max\{|\omega_j^{LF}| : j = 1, \dots, K_{LF}\}$ . Additionally, the learned corrected amplitudes  $\mathbf{b}^{corr}$  corresponding to the low-fidelity frequencies are initialized as their value from Step 1.

The choice of the number of additional high-fidelity frequencies (basis terms)  $K_{HF}$  that need to be learned on sparse high-fidelity training data is theoretically important, but unfortunately intractable a-priori. If  $K_{HF}$  is chosen too small, then the network will be unable to accurately represent the high-fidelity QoI. However, choosing  $K_{HF}$  overly large (larger than the number of training samples) leads to a network of high-complexity that may not be able to learn on sparse training data, and results in an under-determined problem with respect to the frequency parameters.

To address this issue we introduce the following loss functions where the loss function (4.4) corresponds to the direct high-fidelity model (4.3) and loss function (4.5) corresponds to the high-fidelity discrepancy model (4.2)

$$\mathcal{L}(\boldsymbol{\Phi}) = \frac{1}{N_{HF}} \sum_{m=1}^{N_{HF}} |f_{\boldsymbol{\Phi}}(\boldsymbol{\theta}_m) - Q_{HF}(\boldsymbol{\theta}_m)|^2 + \lambda_{L^2} \|\boldsymbol{\Phi}\|_2^2 + \lambda_{L^1} \|(\mathbf{b}^{HF}, \mathbf{b}^{corr})\|_1 \quad (4.4)$$

$$\mathcal{L}_{discrep}(\Phi) = \frac{1}{N_{HF}} \sum_{m=1}^{N_{HF}} |f_\Phi(\theta_m) - (Q_{HF}(\theta_m) - z_{LF}(\theta_m))|^2 + \lambda_{L^2} \|\Phi\|_2^2 + \lambda_{L^1} \|(\mathbf{b}^{HF}, \mathbf{b}^{corr})\|_1. \quad (4.5)$$

The loss is the standard mean squared error on the high-fidelity training set augmented by an  $L^2$  regularization on all network parameters  $\Phi$  with regularization constant  $\lambda_{L^2}$  and an  $L^1$  regularization only applied to the amplitudes  $\mathbf{b}^{HF}, \mathbf{b}^{corr}$  with regularization constant  $\lambda_{L^1}$ . The  $L^2$  regularization helps to avoid over-fitting while the  $L^1$  regularization is used to promote network sparsity and help with feature detection. Ideally, this allows  $K_{HF}$  to be chosen relatively large while keeping effective network complexity low, and simultaneously encourages the network to learn a sparse and interpretable representation of the high-fidelity QoI. In the following section we apply our developed bi-fidelity Fourier features method to a benchmark bi-fidelity modeling problem.

**5. Numerical results.** Consider the following benchmark bi-fidelity modeling task [24] where the high and low-fidelity models are defined as

$$Q_{LF}(\theta) = \sin(8\pi\theta), \quad Q_{HF}(\theta) = (x - \sqrt{2})(Q_{LF}(\theta))^2, \quad \theta \in [0, 1]. \quad (5.1)$$

We construct a surrogate model  $\tilde{Q}$  for  $Q_{HF}$  using three different strategies.

1. **MF:** A multi-fidelity model trained with the bi-fidelity Fourier features method using the loss function  $\mathcal{L}(\Phi)$  defined in Equation (4.4)
2. **MFD:** A multi-fidelity discrepancy model trained with the bi-fidelity Fourier features method with the discrepancy loss function  $\mathcal{L}_{discrep}(\Phi)$  defined in Equation (4.5)
3. **SF:** A single-fidelity Fourier features network trained only on the sparse high-fidelity data using the Adam optimization algorithm with loss function  $\mathcal{L}(\Phi)$  defined in Equation (4.4)

For this problem we estimate  $\|Q_{HF}\|_{L^1} \approx 1.65$  and  $\|\hat{Q}_{HF} - \hat{Q}_{LF}\|_{L^1} \approx 2.68$ , so we expect no accuracy improvement in using the multi-fidelity discrepancy model MFD over the standard multi-fidelity model MF. We further assume that the cost associated with constructing a surrogate model using MF, MFD, or SF is dominated by the cost of its training data.

As such, we train each surrogate on 13 different bi-fidelity training data sets  $S_{N_{HF}}$  defined by the number  $N_{HF}$  of high-fidelity samples. These data sets are summarized in Table 5.1. We

$S_{N_{HF}}$	$S_{10}$	$S_{12}$	$S_{14}$	$S_{16}$	$S_{18}$	$S_{20}$	$S_{25}$	$S_{30}$	$S_{40}$	$S_{50}$	$S_{60}$	$S_{70}$	$S_{80}$
$N_{HF}$	10	12	14	16	18	20	25	30	40	50	60	70	80
$N_{LF}$	50	50	50	50	50	50	50	50	50	50	50	50	50

TABLE 5.1  
Summary of considered bi-fidelity training sets  $S_{N_{HF}}$  for various values of  $N_{HF}$

assume that the cost to generate low-fidelity data is negligible compared to high-fidelity data, and so the cost of the training set is defined as the number of high-fidelity data samples it contains. The data sets  $S_{60}$ ,  $S_{70}$ , and  $S_{80}$  contain more high-fidelity data than low-fidelity data. This does not occur in practice, and these data sets were only considered to show full convergence of the SF surrogate. To evaluate our surrogate performance we use the mean squared error (MSE) (5.2) on an independent test set of  $N_{test} = 500$  samples

$$MSE = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} |Q_{HF}(\theta^{(j)}) - \tilde{Q}(\theta^{(j)})|^2. \quad (5.2)$$

For all training sets the high and low-fidelity training sample inputs  $\theta$  are taken to be uniformly spaced in  $[0, 1]$ . As such, the low-fidelity data samples in each training set are identical, and with  $N_{LF} = 50$ , this low-fidelity data is relatively abundant. We find that with this training data the low-fidelity model can be reliably approximated using the adaptive Fourier features algorithm to a tolerance of  $1e-5$ . Furthermore, since the low-fidelity model is a simple sinusoid, this approximation can be accomplished with  $K_{LF} = 1$ . We conduct this approximation of the low-fidelity model with

the adaptive Fourier features algorithm just once and use the very same extracted low-fidelity information in each implementation of the bi-fidelity Fourier features method on each training set. As a result, this numerical example primarily evaluates how accurately the network in Step 2 of the developed bi-fidelity Fourier features method learns the high-fidelity trainable parameters subject to various data sparsity conditions.

The training of this neural network via the Adam optimization algorithm is subject to the selection of several hyper-parameters, and since it is hard to know a-priori how to choose them, it is important to quantify the predictive uncertainty in the trained surrogate model with respect to these uncertain hyper-parameters. We find empirically that certain aspects of neural network training such as learning rate and number of training epochs are best handled adaptively. The learning rate adaption is done by monitoring the training loss and reducing the learning rate when a plateau is detected. Similarly, we implement an adaptive stopping criteria that monitors the training loss and stops when it plateaus or increases for more than a user defined number of epochs. Additionally, although the appropriate width of a neural network is generally unknown, to facilitate a fair comparison between the three tested surrogate models, we fix the network width across all models and all training sets at 21. The remaining hyper-parameters for which we have considerable uncertainty are the regularization constants  $\lambda_{L^1}$ ,  $\lambda_{L^2}$ , the initial learning rate, and the random seed used in the stochastic training algorithm. We train and test the three models over relatively large ranges of these uncertain hyper-parameters, which are summarized in Table 5.2. Our overarching goal is to quantify the predictive uncertainty in the surrogate with respect to reasonable hyper-parameter choices, and what is reasonable generally depends at least weakly on the amount of training data used in the network. For example, for a fixed architecture, if we have sparser training data, then we are more susceptible to over-fitting and a larger  $L^2$  regularization constant is often optimal. However, for more abundant training data the same regularization constant is often a-priori inappropriate. To elucidate what may be a reasonable hyper-parameter range and quantify predictive uncertainty within that range, we implement the following procedure. In total, each network is trained on 800 unique hyper-parameter combinations. From this ensemble of 800 train/test runs we collect the 100 most accurate results using the computed mean squared error on an independent test set of 500 samples as our metric. We then report the point-wise mean and standard deviation of this ensemble. This procedure allows us to use the same large hyper-parameter ranges for every model and every training set and simultaneously balances removing overly naive hyper-parameter choices and offering uncertainty estimates over a fairly large set of unique hyper-parameter combinations.

$\lambda_{L^1}$	$[10^{-14}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$
$\lambda_{L^2}$	$[10^{-14}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]$
initial learning rate	$[0.5, 0.1]$
random seed	$[101, 679, 875, 1131]$

TABLE 5.2  
*Hyper-parameters used in surrogate training*

Figure 5.1 shows results for surrogate construction on training set  $S_{16}$ . The top two plots correspond to the MF surrogate and the bottom two plots correspond to the SF surrogate. The results for the MFD surrogate are very similar to those for the MF surrogate and hence have been omitted. The left two plots show the true high-fidelity QoI (black solid line) along with the mean surrogate prediction and point-wise uncertainty estimates (red for MF and magenta for SF). The mean squared error in the MF surrogate's average prediction is  $\mathcal{O}(10^{-3})$ , which is an order of magnitude smaller than that for the SF surrogate, and the same is true for the relationship between the surrogates' predictive uncertainties.

The right two plots indicate how close the the two surrogates are to learning the optimal frequencies and amplitudes necessary to represent the high-fidelity QoI over their best 10 train/test runs. The three most important frequencies present in the high-fidelity QoI along with their corresponding amplitudes are pictured in red. In blue, we have the high-fidelity frequencies learned by

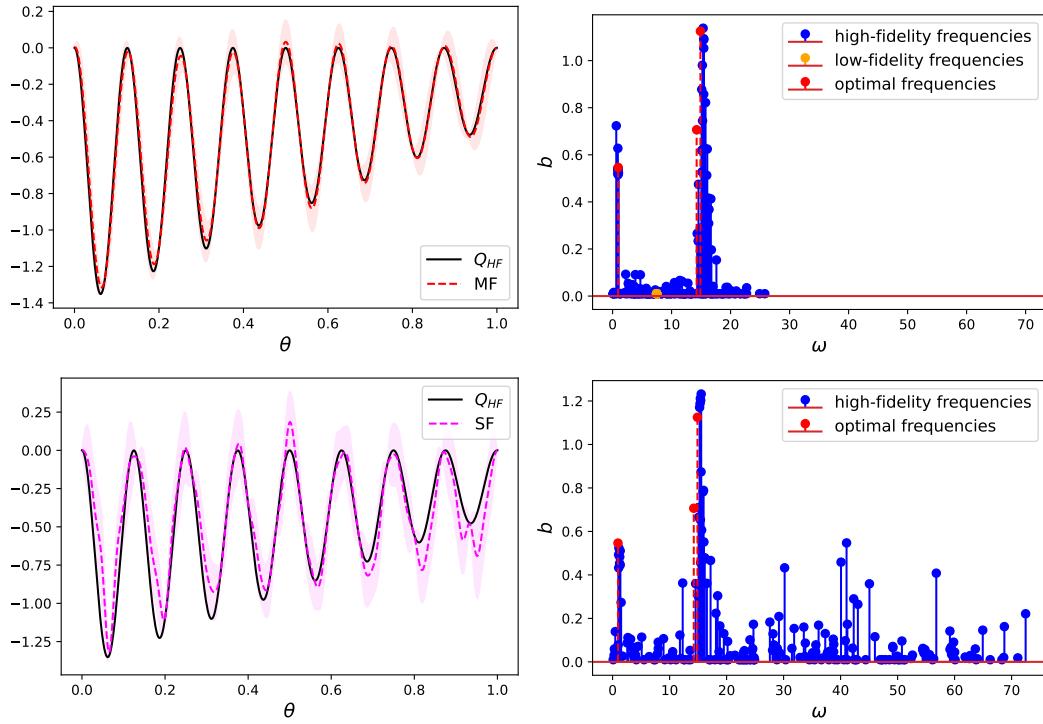


FIG. 5.1. MF surrogate results (top left), MF learned frequencies with corresponding amplitudes (top right), SF surrogate results (bottom left), and SF learned frequencies with corresponding amplitudes (bottom right) for training data set  $S_{16}$

the network, and for the MF surrogate (top right) we additionally plot the fixed low-fidelity frequency along with its learned corrected amplitude in orange. From these right two plots we see that for close to optimal hyper-parameter choices both the SF and MF surrogates approximately learn the optimal frequencies and amplitudes. However, the MF surrogate provides a much sparser, more interpretable representation of the high-fidelity QoI by successfully learning near zero amplitudes for non-optimal frequencies. This is not the case for the SF surrogate which learns many non-zero amplitudes for frequencies that are far from optimal.

Additionally, notice that the MF surrogate consistently learns a near zero corrected amplitude corresponding to the fixed low-fidelity frequency in orange. This indicates that the network was not able to leverage the low-fidelity frequency information directly, and that the performance increase we see for the MF surrogate over the SF surrogate is primarily the result of the low-fidelity informed initialization. This observation can be understood rigorously by recalling that  $Q_{LF}$  appears as a squared quantity in the expression for  $Q_{HF}$ . Considering the complex exponential form of the low-fidelity model, this squaring operation corresponds to multiplying the optimal low-fidelity frequency by 2. Indeed, examining the top right plot in Figure 5.1, we see that the most important frequency in the high-fidelity model (the red frequency with the largest amplitude) is approximately 2 times the low-fidelity frequency in orange. In this particular problem, allowing a learned linear transformation of the low-fidelity frequency, as opposed to assuming that the low-fidelity frequency is exactly present in the high-fidelity model, would allow the low-fidelity frequency information to be leveraged more effectively. Moreover, the non-linear mapping between the low- and high-fidelity model outputs corresponds to just a linear mapping in frequency space, which could likely be learned accurately on sparser data. For a general problem, learning a unique linear transformation applied to each low-fidelity frequency would be computationally prohibitive, but this example does suggest

that combining Fourier features networks with a learned transformation of low-fidelity frequency space could be beneficial. This observation additionally highlights how Goal (i) and (ii) from Section 4 can be complementary. Indeed, for a general problem where the low- and high-fidelity data are potentially related by a non-linear mapping, optimally leveraging low-fidelity frequency information that is informative to the high-fidelity approximation is helpful if not necessary to force the quantity learned on sparse high-fidelity data to be small in uniform norm.

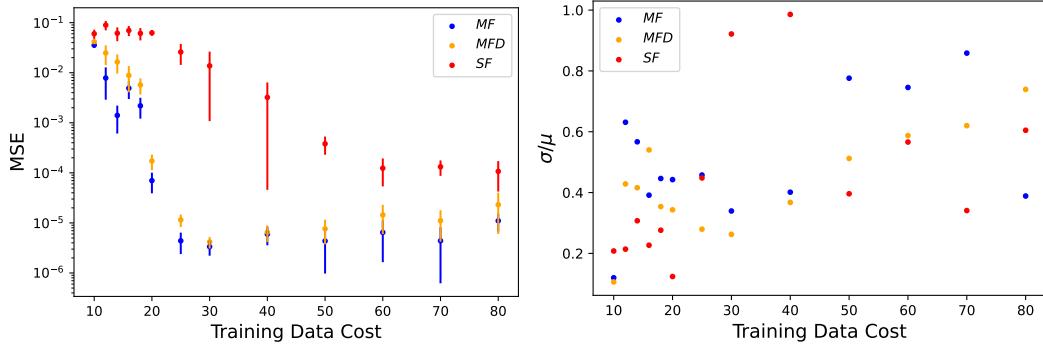


FIG. 5.2. Average mean squared error over 100 unique hyper-parameter combinations as a function of training data cost (left) and standard deviation over average mean squared error as a function of training data cost (right) for MF surrogate (blue), MFD surrogate (orange), and SF surrogate (red)

In Figure 5.2 we present convergence and uncertainty results for each of the surrogate methods over all training data sets  $S_{N_{HF}}$ . The left plot shows average mean squared error with one standard deviation over 100 train/test runs on the 100 best hyper-parameter combinations as a function of training data cost. These 100 best hyper-parameter combinations are surrogate and training data set specific and are determined by the method discussed earlier in this section. We see that both the MF and MFD surrogates converge much faster than the SF surrogate as a function of training data cost. Recall that the cost to generate low-fidelity data was assumed to be negligible, so training data cost is exactly the number of high-fidelity data samples present in the training set. If we instead assume that the cost of low-fidelity data is non-negligible, this would correspond to a rightward shift in the data points for MF and MFD in Figure 5.2, where the magnitude of the shift increases with increasing low-fidelity data cost. Hence the utility of the multi-fidelity surrogates over the single-fidelity surrogate will always be dependent on the relative cost of generating low- and high-fidelity data. The stagnating and noisy behavior in the left plot of Figure 5.2 for higher training data costs is likely due to a combination of factors including the stochastic nature of the training algorithm, being in a convergent regime, insufficient training data, and plotting the average mean squared error over many hyper-parameter combinations as opposed to only the best result. We draw no definitive conclusions concerning the comparative relationship between surrogate performance on these training data sets of higher cost. Rather, we focus on the very clear performance benefit of the multi-fidelity surrogates on training sets of lower cost. Additionally, since the mean squared error is plotted on a logarithmic scale the error bars are somewhat misleading, so we also provide the plot on the right which shows the standard deviation over the average mean squared error as a function of training data cost. The plot indicates that the surrogates are well converged with the mean and standard deviation being of the same order.

**6. Conclusion.** In this work we considered the general problem of surrogate modeling with neural networks and multi-fidelity training data. For this task we considered Fourier features networks. In Section 3 we introduced the single-fidelity version of Fourier features networks and motivated their use in a multi-fidelity context. We reiterate here that this variety of neural network is highly interpretable in a multi-fidelity context, has a flexible Markov Chain Monte Carlo based training algorithm that could likely be modified to scale well with problem dimension, and enjoys unique theoretical guarantees that outline a plausible relationship between generalization error and

the available number of training data.

Continuing, in Section 4 and Section 5 respectively we developed a bi-fidelity 1-layer Fourier features method and applied it to a benchmark bi-fidelity supervised learning problem. Our primary goals in the development of this method were (i) leverage the low-fidelity model to reduce the size of the quantity learned on sparse high-fidelity data, and (ii) leverage the cheap low-fidelity data to discover frequency information (basis terms) that are also necessary to well approximate the high-fidelity model. We attempted to accomplish Goal (i) through discrepancy modeling, and we worked to satisfy Goal (ii) by assuming there was some explicitly overlapping frequency information between the low- and high-fidelity models, and that the range of frequencies learned for the low-fidelity model could be used to provide a useful low-fidelity informed initialization for the neural network learning the high-fidelity parameters. While both high-level goals remain motivating, the tested numerical example indicates that the preliminary assumptions made to satisfy these goals could be improved.

For Goal (i), while the discrepancy between high- and low-fidelity models is often small relative to the high-fidelity model, this cannot be expected in general. For example, the high- and low-fidelity models in Section 5 are separated by a phase shift causing the discrepancy to actually be larger in size than the high-fidelity model itself. Hence one important improvement could involve a learned transformation of the low-fidelity model outputs that attempts to minimize the size of the discrepancy in the appropriate norm. Such a modification could also help to generalize the developed bi-fidelity method to handle more than one low-fidelity model. In such cases, it is not immediately clear what is meant by a discrepancy model, and instead we could attempt to target the difference between the high-fidelity model and some sort of aggregate learned transformation of all or some portion of the lower-fidelity models. Additionally, recalling Theorems 3.2 and 3.1, the generalization error bound for deep networks as a function of target quantity size is generally more favorable than its 1-layer counterpart. Hence an additional planned improvement is the extension of the developed method for deep Fourier features networks.

For Goal (ii) the assumption that the high- and low-fidelity models have some amount of explicitly overlapping frequency information is only true when the low-fidelity model appears linearly in the expression for the high-fidelity model. In cases where the mapping between the two models is non-linear, as in Section 5, leveraging the discovered low-fidelity frequency terms will require at minimum allowing linear transformations of low-fidelity frequency space. Despite not being able to leverage the learned low-fidelity frequency information directly, we still observed considerable benefit to our developed low-fidelity informed initialization evidenced by the order of magnitude reduction in mean squared error for our multi-fidelity surrogates over our single-fidelity surrogate trained on sparse data.

## REFERENCES

- [1] S.-K. AU AND J. L. BECK, *Estimation of small failure probabilities in high dimensions by subset simulation*, Probabilistic engineering mechanics, 16 (2001), pp. 263–277.
- [2] A. R. BARRON, *Approximation and estimation bounds for artificial neural networks*, Machine learning, 14 (1994), pp. 115–133.
- [3] I. DAUBECHIES, R. DEVORE, S. FOUCART, B. HANIN, AND G. PETROVA, *Nonlinear approximation and (deep) relu networks*, Constructive Approximation, 55 (2022), pp. 127–172.
- [4] O. DAVIS, M. MOTAMED, AND R. TEMPONE, *Residual multi-fidelity neural network computing*, arXiv preprint arXiv:2310.03572, (2023).
- [5] D. ELBRÄCHTER, D. PEREKRESTENKO, P. GROHS, AND H. BÖLCSKEI, *Deep neural network approximation theory*, IEEE Transactions on Information Theory, 67 (2021), pp. 2581–2623.
- [6] M. G. FERNÁNDEZ-GODINO, C. PARK, N.-H. KIM, AND R. T. HAFTKA, *Review of multi-fidelity models*, arXiv preprint arXiv:1609.07196, (2016).
- [7] 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.
- [8] 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.

- [9] M. B. GILES, *Multilevel Monte Carlo path simulation*, Operations research, 56 (2008), pp. 607–617.
- [10] I. GOODFELLOW, Y. BENGIO, AND A. COURVILLE, *Deep learning*, MIT press, 2016.
- [11] A. A. GORODETSKY, G. GERACI, M. S. ELDRED, AND J. D. JAKEMAN, *A generalized approximate control variate framework for multifidelity uncertainty quantification*, Journal of Computational Physics, 408 (2020), p. 109257.
- [12] M. GUO, *A brief note on understanding neural networks as Gaussian processes*, arXiv preprint arXiv:2107.11892, (2021).
- [13] 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.
- [14] A.-L. HAJI-ALI, F. NOBILE, AND R. TEMPONE, *Multi-index monte carlo: when sparsity meets sampling*, Numerische Mathematik, 132 (2016), pp. 767–806.
- [15] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [16] A. KAMMONEN, J. KISSLING, P. PLECHÁČ, M. SANDBERG, AND A. SZEPESSY, *Adaptive random Fourier features with Metropolis sampling*, arXiv preprint arXiv:2007.10683, (2020).
- [17] A. KAMMONEN, J. KISSLING, P. PLECHÁČ, M. SANDBERG, A. SZEPESSY, AND R. TEMPONE, *Smaller generalization error derived for a deep residual neural network compared to shallow networks*, arXiv preprint arXiv:2010.01887, (2020).
- [18] L. S. KATAFYGIOTIS AND K. M. ZUEV, *Geometric insight into the challenges of solving high-dimensional reliability problems*, Probabilistic Engineering Mechanics, 23 (2008), pp. 208–218.
- [19] 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.
- [20] D. P. KINGMA AND J. BA, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).
- [21] 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).
- [22] 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.
- [23] X. MENG, H. BABAEE, AND G. E. KARNIADAKIS, *Multi-fidelity bayesian neural networks: Algorithms and applications*, Journal of Computational Physics, 438 (2021), p. 110361.
- [24] 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.
- [25] M. MOTAMED, *A multi-fidelity neural network surrogate sampling method for uncertainty quantification*, International Journal for Uncertainty Quantification, 10 (2020).
- [26] M. MOTAMED, *Approximation power of deep neural networks: an explanatory mathematical survey*, arXiv preprint arXiv:2207.09511, (2022).
- [27] R. M. NEAL, *Bayesian learning for neural networks*, vol. 118, Springer Science & Business Media, 2012.
- [28] P. S. PALAR, T. TSUCHIYA, AND G. T. PARKS, *Multi-fidelity non-intrusive polynomial chaos based on regression*, Computer Methods in Applied Mechanics and Engineering, 305 (2016), pp. 579–606.
- [29] 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.
- [30] A. PASZKE, S. GROSS, F. MASSA, A. LERER, J. BRADBURY, G. CHANAN, T. KILLEEN, Z. LIN, N. GIMELSHEN, L. ANTIGA, ET AL., *Pytorch: An imperative style, high-performance deep learning library*, Advances in neural information processing systems, 32 (2019).
- [31] B. PEHERSTORFER, K. WILLCOX, AND M. GUNZBURGER, *Optimal model management for multifidelity Monte Carlo estimation*, SIAM Journal on Scientific Computing, 38 (2016), pp. A3163–A3194.
- [32] 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.
- [33] P. PERDIKARIS, D. VENTURI, J. O. ROYSET, AND G. E. KARNIADAKIS, *Multi-fidelity modelling via recursive co-kriging and Gaussian–Markov random fields*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471 (2015), p. 20150018.
- [34] A. PINKUS, *Approximation theory of the mlp model in neural networks*, Acta numerica, 8 (1999), pp. 143–195.

- [35] A. F. PSAROS, X. MENG, Z. ZOU, L. GUO, AND G. E. KARNIADAKIS, *Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons*, Journal of Computational Physics, 477 (2023), p. 111902.
- [36] I. SUTSKEVER, J. MARTENS, G. DAHL, AND G. HINTON, *On the importance of initialization and momentum in deep learning*, in International conference on machine learning, PMLR, 2013, pp. 1139–1147.
- [37] L. TIERNEY AND A. MIRA, *Some adaptive monte carlo methods for bayesian inference*, Statistics in medicine, 18 (1999), pp. 2507–2515.
- [38] X. ZENG, G. GERACI, A. GORODETSKY, M. ELDRED, J. JAKEMAN, AND R. GHANEM, *Multifidelity uncertainty quantification with models based on dissimilar parameters*, Computer Methods in Applied Mechanics and Engineering, 415 (2023), p. 116205.
- [39] Z. ZOU, X. MENG, A. F. PSAROS, AND G. E. KARNIADAKIS, *Neuraluq: A comprehensive library for uncertainty quantification in neural differential equations and operators*, arXiv preprint arXiv:2208.11866, (2022).
- [40] K. M. ZUEV AND L. S. KATAFYGIOTIS, *Modified metropolis-hastings algorithm with delayed rejection*, Probabilistic Engineering Mechanics, 26 (2011), pp. 405–412.

## Appendix A. Pseudocode.

---

### Algorithm 1 Adaptive Fourier features

---

**Input:**  $\{(\boldsymbol{\theta}^{(m)}, Q(\boldsymbol{\theta}^{(m)}) - z_{\ell-1}(\boldsymbol{\theta}^{(m)}))\}_{m=1}^N$  for some  $\ell = 1, \dots, L$   
**Output:** trained parameters  $\mathbf{b}_\ell, \mathbf{b}'_\ell, \omega_\ell, \omega'_\ell$   
 $A(\omega_\ell, \omega'_\ell) :=$  computes  $\mathbf{b}_\ell, \mathbf{b}'_\ell$  by solving convex inner optimization in (3.1)  
 $M :=$  sampling time  
 $\delta :=$  Metropolis step length  
 $K :=$  network width  
 $\gamma :=$  acceptance criteria exponent  
**Initialize Trainable Parameters:**  
 $\omega'_\ell \leftarrow \mathcal{N}(0, 1)$   
 $\omega_\ell \leftarrow \mathcal{N}(0, 1)$   
 $\mathbf{b}_\ell, \mathbf{b}'_\ell \leftarrow A(\omega_\ell, \omega'_\ell)$   
**Begin Metropolis-Hastings Algorithm:**  
**for**  $i = 1, \dots, M$  **do**  
     $\omega_{\ell,new} \leftarrow \omega_\ell + \delta \mathcal{N}(0, 1)$   
     $\mathbf{b}_{\ell,new}, \mathbf{b}'_{\ell,new} \leftarrow A(\omega_{\ell,new}, \omega'_\ell)$   
    **for**  $j = 1, \dots, K$  **do**  
        **if**  $(|\mathbf{b}_{\ell,new}^{(j)}| / |\mathbf{b}_\ell^{(j)}|)^\gamma > \mathcal{U}(0, 1)$  **then**  
             $\omega_\ell^{(j)} \leftarrow \omega_{\ell,new}^{(j)}$   
             $\mathbf{b}_\ell^{(j)}, \mathbf{b}_\ell^{(j)'} \leftarrow \mathbf{b}_{\ell,new}^{(j)}, \mathbf{b}_{\ell,new}^{(j)'}$   
        **end if**  
    **end for**  
**end for**  
**end for**  
 $\mathbf{b}_\ell, \mathbf{b}'_\ell \leftarrow A(\omega_\ell, \omega'_\ell)$   
**Return:**  $\mathbf{b}_\ell, \mathbf{b}'_\ell, \omega_\ell, \omega'_\ell$

---

## DEVELOPING A MACHINE-LEARNED INTERATOMIC POTENTIAL FOR NICKEL-PLATINUM CATALYSTS

ISABELLA FURRICK\*, MITCHELL WOOD†, AND ALYSSA HENSLEY‡

**Abstract.** The hydrogen oxidation reaction (HOR), where  $H_2$  and  $O_2$  are converted over a catalyst to  $H_2O$ , is a promising route for carbon-free energy production, but its widespread use is limited due to the high cost and rarity of the current state-of-the-art noble metal catalysts. Nickel (Ni) catalysts show promise for HOR, but require doping it with a secondary, noble metal such as platinum (Pt) to improve performance. Further complicating the design of Ni-Pt bimetallic catalysts is the fact that little is known about the structure and composition of such catalysts under HOR conditions, as surface reconstruction may occur, but cannot be probed experimentally. In this study, we aimed to develop a machine learned Spectral Neighbor Analysis Potential (SNAP) for Ni-Pt catalysts with oxygen ( $O^*$ ), a critical HOR intermediate, bound to the surface. The potential was trained on *ab initio* data generated for a diverse set of structures, compositions, and environmental conditions. Potential accuracy was evaluated by setting objective functions for lattice parameters, total energy, defect formation energy, and  $O^*$  adsorption energy.

**1. Introduction.** Hydrogen fuel cells are a promising alternative energy source to carbon-based fossil fuels. In such fuel cells, the hydrogen oxygen reaction (HOR) occurs where  $H_2$  is oxidized to  $H_2O$  by a catalyst, a solid material that increases the rate of chemical reactions. Platinum (Pt) is the most efficient catalyst for HOR[1]. However, the high cost and scarcity of Pt limits the widespread use of fuel cells. Thus, there is a critical need to find cheaper, more sustainable materials to replace the Pt catalysts in hydrogen fuel cells.

Currently, research shows that bimetallic catalysts, i.e., catalysts containing two transition metals, are the best alternatives to Pt[2]. In particular, nickel (Ni)-based bimetallic catalysts show promising reactivity towards HOR[3, 4]. Preliminary computational modeling of Ni catalysts doped with 14 different transition metals showed that Ni-Pt catalysts behave most similarly to pure Pt catalysts for HOR.

Bimetallic catalysts may undergo a phenomenon known as ‘surface reconstruction’,[5] shown in Figure 1.1, where atoms in the subsurface move to the surface and vice versa. For HOR, such surface reconstructions occur due to the concentration of a key intermediate, oxygen ( $O^*$ ), bound to the catalyst surface. Although virtually impossible to probe experimentally, tracking catalytic surface reconstruction is crucial because a catalyst’s structure determines reaction kinetics. Thus, to design appropriate catalysts, the thermodynamic driving forces behind surface reconstruction need to be understood. With computational modeling, surface reconstruction can be observed, and reconstruction trends identified allowing for more informed design decisions.

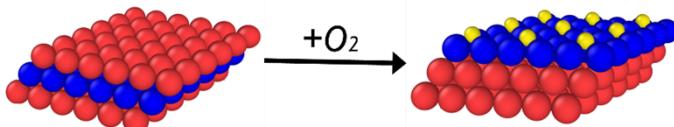


Fig. 1.1: Ni (red) and Pt (blue) bimetallic (111) facet undergoing complete surface reconstruction as  $O^*$  (yellow) adsorbs to the catalyst surface.

\*Stevens Institute of Technology, ifurrick@stevens.edu

†Sandia National Lab, mitwood@sandia.gov

‡Stevens Institute of Technology, ahensley@stevens.edu

Quantum mechanical calculations, such as Density Functional Theory (DFT) and Ab Initio Molecular Dynamics (AIMD), are the most accurate methods for energetically characterizing a system but are very computationally expensive. Molecular dynamics (MD) simulations can model systems at greater length and time scales than quantum mechanical simulations, but often with less accuracy. In this study, we took a multiscale approach to model larger systems, above the order of a few hundred atoms, at more realistic time scales and reaction conditions. In MD, the interatomic potential (IAP) determines the physical properties of the simulated system, specifically the forces and energies on the atoms. We used machine learning (ML) to train an IAP on a database of DFT and AIMD calculations, thus bridging quantum mechanical and molecular dynamics methods.

## 2. Interatomic Potential.

**2.1. Spectral Neighbor Analysis (SNAP).** Several types of IAP exist, each with their own mathematical approaches for expressing the energies, forces, and stress of atoms in a system. For many decades, the functional form of an IAP was derived based on the bonding characteristics of the material. These include simple models, also called empirical potentials, of dispersion interactions[6], hard spheres, or harmonic springs for covalent bonds[7]. Progress in this area has been tracked by the complexity and associated computational cost[8], and has brought about IAP that capture many-body bonds[9], dynamic charges[10], and magnetism[11]. In the last decade data-driven model forms, which includes ML methods, have demonstrated unprecedented accuracy and computational cost trade-offs[12]. These methods (ML-IAP) distinguish themselves from empirical IAP by constructing a generalized representation (also called features or descriptors) of atomic structure that can be mapped into any expression for energy/force. Distinguishing between ML-IAP comes down to the feature set, training set, and model form (i.e. neural network, Gaussian Process, polynomial, etc.). For this study, a Spectral Neighbor Analysis Potential (SNAP) potential, which expresses the energy of each atom in a system as a linear function of selected bispectrum components of neighbor atoms [13, 14], was chosen.

A SNAP ML-IAP can be efficiently constructed by utilizing the Sandia developed Fit-SNAP code[15], but generating a viable IAP for the complex surface interactions during HOR on Ni-Pt catalysts is critically dependent on the available training data.

**2.2. Constructing the Training Set.** ML-IAPs make force and energy predictions about a system by numerically interpolating between quantum-mechanical reference data. Most commonly these reference data are high-fidelity DFT calculations. A detailed summary of this method can be found in review articles such as Ref[16]. When making predictions about new atomic configurations, extrapolation is required, which often produces inaccurate or physically meaningless results[17]. To minimize extrapolation, we set out to adequately span the descriptor space by constructing a large, diverse training data set.

The initial DFT calculations were performed with the Vienna Ab Initio Simulation Package (VASP) on the Stevens Institute of Technology's high-performance computing (HPC) cluster, Dorothy. To account for electron-to-electron interactions within the surface and the adsorbate, a Revised Perdew-Burke-Ernzerhof (RPBE) exchange correlation functional was used. All calculations were performed with a minimum plane wave cutoff energy of 400 eV. The tolerances for the ionic and electronic relaxations were 0.02 eV/Å and 10-6 eV, respectively. The Methfessel Paxton [18] ( $N = 1$ ) smearing method was used with a smearing width of 0.1 eV. All VASP calculations were spin polarized to account for the magnetism of Ni.

Slab calculations were performed for three different Ni-Pt surface facets: (111), (110), and (100). The supercell sizes were p(2x2) for both (111) and (100) facets, while the more

open (110) facet was p(2x1). Gamma-centered k-points meshes were used for all facets, with the (111) and (100) facets using a mesh of (10x10x1) and the (110) facet using a mesh of (6x10x1). Bulk structures were simulated with a (20x20x20) Gamma-centered k-points mesh. Four different slab concentrations were included in the training data: pure Ni, pure Pt, Ni-rich (94% Ni, 6% Pt), and Pt-rich (94% Pt, 6% Ni). Promoter metal placement, specifically the proximity of the promoter metal to the adsorbate, affects its ability to interact with molecules on the surface. To account for this, a variety of promoter metal placements, both in the surface and subsurface of the facet, were scanned. Figure 2.1 depicts the promoter metal sites that were tested for each facet.

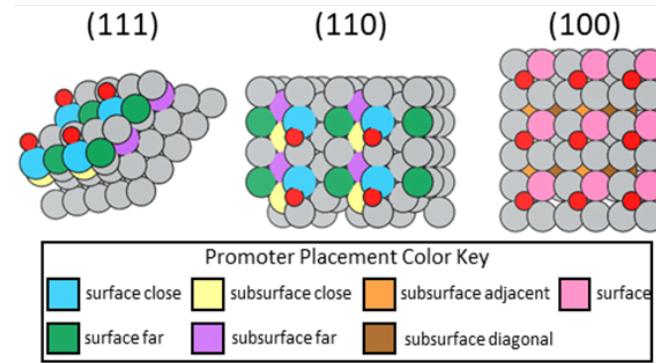


Fig. 2.1: Promoter metal placements for (111), (110), and (100) Ni-Pt slabs.

DFT was run for each energetically equivalent composition with and without O\*. Similar to the promoter metal, a variety of O\* placements were tested on each facet to account for the many possible adsorption sites of O\* during HOR. Bulk DFT data was obtained for Ni<sub>3</sub>Pt with lattice constants ranging from 3.50 to 5.75 Å, increasing in increments of 0.05 Å. Aside from slab and bulk data, the training set included a DFT calculation for a pure oxygen (O<sub>2</sub>) molecule run with a minimum cutoff energy of 400 eV and sampled at the gamma point.

AIMD simulations provide information about how atoms move with an applied temperature over short time scales which can help potentials more accurately predict atomic displacements and reconstructions. AIMD simulations were performed on select surfaces with VASP on the Sandia HPC clusters, Ghost and Attaway. The canonical ensemble was chosen to hold the number of atoms, volume of the structure, and temperature of the system constant throughout the simulation. Each simulation was run with 1.0 femtosecond timesteps at three different temperatures (300K, 1000K, and 12000K) with a minimum cutoff energy of 400 eV and Gamma-centered k-points meshes of (10x10x1).

With training data from DFT and AIMD, we were able to construct a training data set of approximately 90,000 training data points.

**2.3. Optimization Methodology.** To produce a viable IAP from our NiPtO training data, a number of free variables in a SNAP ML-IAP must be optimized. These generally fall into two categories, those that affect the calculation of the descriptors (hyper-parameters) and those that shift the value of different parts of the DFT training data (group weights). Optimizing these variables can improve ML-IAP accuracy. Two approaches were used to find an optimal set of parameters for the NiPtO potential: hand optimization and genetic

algorithm optimization. To build intuition about the system, hand tuning was done for SNAP's hyperparameters. Then, to find an optimal combination of group weights, we employed a genetic algorithm called GA-SNAP.

The hyperparameters of interest in this study were radial cutoff, radelem, and wj. The radial cutoff determines the number of atoms in a local neighbor list. Atoms exert energies, forces, and stresses on each other, and the radial cutoff determines how far an atom can be from another atom before its affect is negligible. The radial cutoff was scanned over the range 3.3 to 7.0 Å. A few radial cutoffs were tested based on intuition, including the Ni and Pt lattice parameters (3.524 Å and 3.924 Å, respectively). The radelem refers to the effective radius of each atom type and is unique for every element in the system. To optimize the radelem, Ni and Pt were chosen to be ‘parent materials’ as relative to O\*, they are the similar sizes. The Ni and Pt radelems were fixed at 0.5, while the O\* radelem was allowed to vary over the range 0.25 to 0.75 increasing in increments of 0.05. The wj determines the weight that neighboring atoms contribute. The Ni and Pt wjs were set at 1.0 while the O\* wj varied from 0.5 to 1.4 in increments of 0.1. The hyperparameters were scanned one at a time starting with the radial cutoff. After an optimal radial cutoff was determined, the O\* radelem was scanned. Finally, after an optimal O\* radelem was determined, the O\* wj was scanned.

To reduce computational cost, these hyperparameters were scanned on a small training data set of 1,200 ground-state training data points generated through a selective sampling of AIMD data. During AIMD simulations, structures were sampled once every femtosecond, thus producing an abundance of data for slightly different configurations of a single structure. To reduce computational cost during the hyperparameter scan, a small training data set was produced with a reduced number of AIMD training data points. In the small 1,200-structure training data set, only AIMD structures with outlying total energies were included. While we recognize that using a sample training data set during the scan may have resulted in a sub-optimal set of hyperparameters for the full training set, this approach significantly accelerated the project.

The overall accuracy of a ML-IAP is determined by a combination of the potential’s ability to predict material properties, system stability, and energies/forces. Thus, to evaluate model accuracy while scanning hyperparameters, we set six objective functions and tracked their errors. To discern the ability of the ML-IAP to predict the NiPtO system’s material properties, we set objective functions for the lattice parameters of Ni and Pt. We obtained the predicted lattice parameters by running short LAMMPS simulations on boxes of pure Ni and Pt. Then, we subtracted the predicted lattice constants from those obtained in experiment. Similarly, to test how well the potential was able to predict the stability of the system, we set objective functions for the formation and O\* adsorption energies of a (111) Ni-rich slab with O\* bound to the surface. We obtained the predicted energies from short LAMMPS simulations, then subtracted the predicted energies from those calculated with DFT. Lastly, with each potential fit, FitSNAP produces a metrics file with details about how the absolute energy and forces predicted by the potential compare to those in the training data set. The energy and force mean absolute errors (MAEs) from this metrics file were also tracked throughout the hyperparameter scan. Figure 2.2 below depicts how the accuracy of the model changed as each hyperparameter was scanned.

Determining an optimal hyperparameter was done based on the overall accuracy of the model, with each of the objective functions being weighted equally. As each of the hyperparameters were scanned, the six aforementioned objective function errors were recorded in a table like the one shown below in Figure 2.1. A second table was then produced, where each column assigned a score based on the relative magnitude of the objective function error. Taking the O\* Radelem hyperparameter scan as an example, eleven radelems were tested.

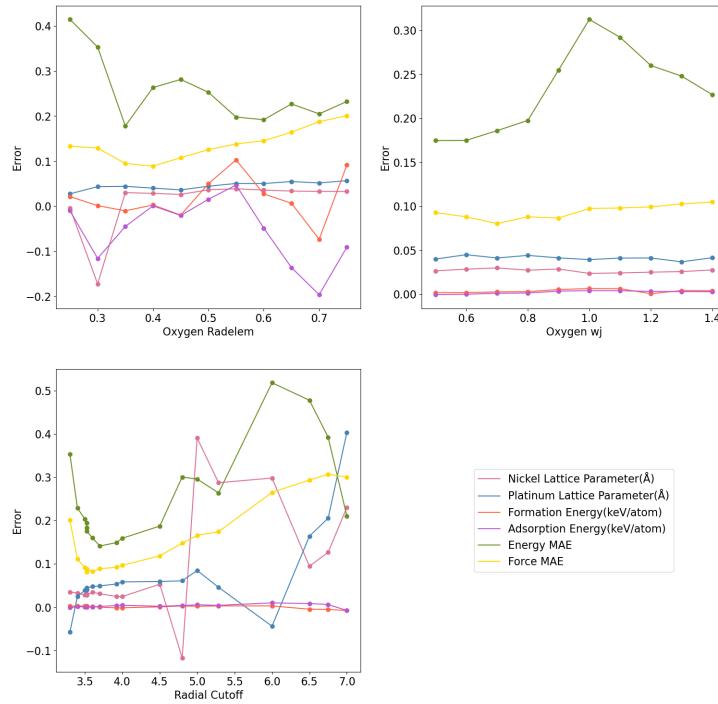


Fig. 2.2: Model accuracy in predicting system features as hyperparameters are varied. User determined trade-offs between these properties is necessary as there is no simultaneous global minima.

The radelem that produced the smallest Ni lattice parameter error was ranked '1' for that category, while the radelem that produced the largest Ni lattice parameter error was ranked '11' for that category. This process was repeated until the radelems were ranked for each of the six objective functions. Then, by adding across each row of the 'Ranking Table', shown in Figure 2.2, an 'Overall Score' was produced. The hyperparameters were once again ranked, that with the lowest 'Overall Score' being ranked '1' chosen as the optimal hyperparameter. Based on the above analysis, the optimal set of hyperparameters for this potential was determined to be a radial cutoff of 3.524 Å, an O\* radelem of 0.4, and an O\* wj of 0.7.

When developing the training set for this potential, the data was sorted into groups based on simulation type (traditional DFT, AIMD at 300K, 1000K, and 1200K) and material type (Ni, Pt, O<sub>2</sub>, Ni-rich, and Pt-rich). Each group was assigned a 'group weight' which tells FitSNAP how much to prioritize the group's energy and force errors[19]. In other words, the higher a group's weight, the more influence it has on a potential. To find an optimal set of group weights, a Sandia-developed genetic algorithm called GA-SNAP was implemented. GA-SNAP produces a set of prospective group weights, applies them to a potential with FitSNAP, then calculates the force and energy errors of the potential. It repeats this process, adjusting the prospective group weights until it sufficiently minimizes the force and energy errors of the potential.

**3. Results and Discussion.** With the generation of sufficient training data and the optimization of hyperparameters, the NiPtO ML-IAP improved over the course of its de-

Oxygen Radelem Scan: Raw Data						
Oxygen Radelem	Nickel Lattice Parameter Error	Platinum Lattice Parameter Error	Formation Energy Error	Adsorption Energy Error	Energy Unweighted Training MAE	Force Unweighted Training MAE
0.25	-0.0036169	0.0276144	21.4257	-9.90472	0.414386	0.13331
0.3	-0.172219	0.0434725	1.38785	-115.621	0.353126	0.129059
0.35	0.0301699	0.044245	-10.108	-44.923	0.17823	0.0948886
0.4	0.0286438	0.0402598	2.88661	1.02179	0.263299	0.0889589
0.45	0.0262455	0.0363305	-19.6019	-20.5399	0.281327	0.1077
0.5	0.0365915	0.0444707	50.0506	15.4499	0.252839	0.125546
0.55	0.0388026	0.0505763	102.877	47.0105	0.197814	0.138096
0.6	0.0358782	0.0503923	27.3777	-48.4857	0.191957	0.145411
0.65	0.0337184	0.0548726	6.67839	-136.017	0.226983	0.16426
0.7	0.0330442	0.0516918	-73.2718	-196.075	0.204924	0.187614
0.75	0.0327401	0.0568726	91.3927	-90.6372	0.23245	0.200896

Table 2.1: Table with raw data from O\* Radelem hyperparameter scan.

Oxygen Radelem Scan: Ranking								
Oxygen Radelem	Nickel Lattice Parameter Error Ranking	Platinum Lattice Parameter Error Ranking	Formation Energy Error Ranking	Adsorption Energy Error Ranking	Energy Unweighted Training MAE Ranking	Force Unweighted Training MAE Ranking	Overall Score	Overall Ranking
0.25	1	1	6	2	11	7	28	4
0.3	11	4	1	9	10	5	40	7
0.35	4	5	4	5	1	2	21	2
0.4	3	3	2	1	8	1	18	1
0.45	2	2	5	3	9	3	24	3
0.5	9	6	8	4	7	4	38	5
0.55	10	8	11	6	3	6	44	8
0.6	8	7	7	7	2	8	39	6
0.65	7	10	3	10	6	9	45	9
0.7	6	9	9	11	4	10	49	10
0.75	5	11	10	8	5	11	50	11

Table 2.2: Table with ranked data from O\* Radelem hyperparameter scan.

velopment. Similar to the hyperparameter scan, we bench marked the improvement of the ML-IAP over its development quantitatively by comparing model predictions to true values obtained experimentally and with DFT and AIMD calculation methods.

While several candidate ML-IAPs were produced and tested throughout the course of this study, this paper highlights four of them for simplicity. The ML-IAPs are numbered based on the order in which they were produced. Potential 1 includes only the initial DFT training data for the slab and bulk structures. Only one hyperparameter, the radial cutoff, was optimized, and no group weight optimization was performed. Potential 2 saw the inclusion of AIMD data for pure Ni and pure Pt at 300K. Potential 3 included AIMD data for pure Ni, pure Pt, Ni-rich, and Pt-rich structures with and without O\* bound to the surface at 300K and 1000K. The most recent potential, Potential 4, was trained on all 90,000 training data points discussed in the 'Constructing the Training Set' portion of this paper. Potential 4's hyperparameters were optimized by hand and group weights were optimized by GA-SNAP.

The data in Figure 3.1 below depicts the ability of each of the four potentials to quan-

titatively predict seven different system features. The properties of interest were the total energies of pure Ni and pure Pt bulk in eV/atom (shown in Panel A), the total energy of an O<sub>2</sub> molecule in eV/atom (Panel B), the lattice parameters of Ni and Pt in Å (Panel C), and the total energies of a Ni-rich (111) slab with and without O\* bound to the surface in eV/atom (Panel D). We chose to track the aforementioned properties because they provide insight on the model's ability to reproduce the physics and chemistry of systems of interest of different sizes and compositions.

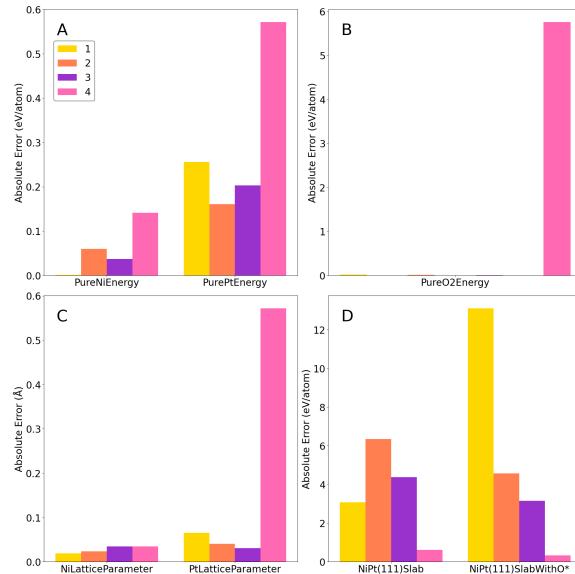


Fig. 3.1: Model accuracy over development where numerical index represents the generation/iteration that the top performing potential came from.

Of all the tested potentials, the most recent potential, Potential 4, showed the largest objective function errors for the energies of the pure elements, Ni, Pt, and O<sub>2</sub>. Similarly, Potential 4 was the least accurate in predicting the lattice parameter of pure Pt. Its platinum lattice parameter error was on the order of 10<sup>-2</sup>, while the platinum lattice parameter errors for the other three potentials were on the order of 10<sup>-3</sup>. These results show that despite additional training data and parameter tuning, over its development, the ML-IAP's ability to predict the properties of pure elements declined. However, Potential 4 was the most accurate in predicting the total energies of systems with more than one element. Potential 4's objective function error for the total energy of a NiPt (111) slab was between 5 and 10 times lower than the other three ML-IAPs. More impressively, Potential 4's objective function error for the total energy of a NiPt (111) slab with O\* bound to the surface was between 9 and 39 times lower than the other three ML-IAPs.

There is a clear trade-off in model accuracy for single element versus multi-element systems. While the ML-IAP became less accurate at predicting features of systems of single elements over the course of its development, it showed major improvements in its ability to capture the interactions between multiple element types. Because the goal of this study is to produce a potential that can be used to study O\* on a Ni-Pt catalyst, a system where three element types are present, Potential 4 is the one that best captures the physics and chemistry of interest.

Future work will look for new solutions to these observed accuracy trade-offs, including a modification of the model form, or utilizing alternate descriptor sets such as the Atomic Cluster Expansion[20]. These will alter the optimization procedure, but will be able to use the same training set generated for the SNAP studies to date.

**4. Conclusion.** In this study, we began developing a ML-IAP to model O\* adsorption on a Ni-Pt bimetallic catalyst. Over the course of this project, we increased the size of the training data set from less than 100 to approximately 90,000 structures of diverse sizes and compositions. While the ML-IAP saw improvement with the addition of each batch of training data, including AIMD simulation data was a turning point in model accuracy. Optimizing hyperparameters and group weights also played a significant role in ML-IAP development. While hand-tuning hyperparameters was a necessary step for building intuition about the system, using genetic algorithms, such as GA-SNAP, quickly produce optimal sets of variables and should be used whenever possible. The potential was enhanced significantly through the addition of training data and parameter tuning, however there is room for further refinement.

The next step in this project is to add structures to the training set that are beyond domain expertise. To do this, we will continue to implement USPEX [cite] to search out chemically complex structures. Given the chemical composition of a system, USPEX predicts stable (lowest energy) and metastable structures. We will then run DFT and/or AIMD on USPEX's predicted structures to obtain their total energies and forces. Adding structures beyond domain expertise will help to fill in the descriptor space and penalize the formation of unphysical structures from the ML-IAP. We also have a few strategies to help the potential better capture the behavior of O<sub>2</sub>. Switching from SNAP to ACE descriptors could drastically improve the model's ability to predict the behavior of a light, gaseous element like oxygen. We also plan to add training data for structures with higher concentrations of O\* and O<sub>2</sub> molecules bound to NiPt surfaces.

Computational modeling, specially ML-IAPs, help capture physical and chemical behaviors of systems that cannot be seen with the naked eye or probed experimentally. The model produced in this study is a stepping stone towards understanding oxygen adsorption on a Ni-Pt catalyst, but the work does not end there. We aim to be able to model HOR on Ni-based bimetallics of varying compositions. Understanding this fundamental reaction will help us make more informed design decisions for the production of novel catalysts for hydrogen fuel cells, leading us toward a carbon-free future.

## References.

- [1] O. Holton and J. Stevenson. "The Role of Platinum in Proton Exchange Membrane Fuel Cells". In: *Platinum Metals Review* 57 (2013), pp. 259–271. DOI: 10.1595/147106713X671222.
- [2] A. Serov and C. Kwak. "Review of non-platinum anode catalysts for DMFC and PEMFC application". In: *Applied Catalysis B: Environmental* 90 (2009), pp. 313–320. DOI: 10.1016/j.apcatb.2009.03.030.
- [3] et al. Roy A. "Nickel–copper supported on a carbon black hydrogen oxidation catalyst integrated into an anion-exchange membrane fuel cell". In: *Sustainable Energy and Fuels* 10 (2018), pp. 2268–2275. DOI: 10.1039/C8SE00261D.
- [4] et al. Tang M. "Nickel–silver alloy electrocatalysts for hydrogen evolution and oxidation in an alkaline electrolyte". In: *Physical Chemistry Chemical Physics* 36 (2014), pp. 19250–19257. DOI: 10.1039/C4CP01385A.
- [5] Zhoufeng Bain et al. "A Review on Bimetallic Nickel-Based Catalysts for CO<sub>2</sub> Reforming of Methane". In: *ChemPhysChem* 18 (22 Nov. 2017), pp. 3117–3134. DOI: 10.1002/cphc.201700529.

- [6] John Edward Jones. “On the determination of molecular fields.—II. From the equation of state of a gas”. In: *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character* 106.738 (1924), pp. 463–477.
- [7] Alex D MacKerell Jr et al. “All-atom empirical potential for molecular modeling and dynamics studies of proteins”. In: *The journal of physical chemistry B* 102.18 (1998), pp. 3586–3616.
- [8] Steven J Plimpton and Aidan P Thompson. “Computational aspects of many-body potentials”. In: *MRS bulletin* 37.5 (2012), pp. 513–521.
- [9] Adri CT Van Duin et al. “ReaxFF: a reactive force field for hydrocarbons”. In: *The Journal of Physical Chemistry A* 105.41 (2001), pp. 9396–9409.
- [10] Jianguo Yu, Susan B Sinnott, and Simon R Phillpot. “Charge optimized many-body potential for the Si/ SiO<sub>2</sub> system”. In: *Physical Review B* 75.8 (2007), p. 085311.
- [11] Julien Tranchida et al. “Massively parallel symplectic algorithm for coupled magnetic spin dynamics and molecular dynamics”. In: *Journal of Computational Physics* 372 (2018), pp. 406–425.
- [12] Yunxing Zuo et al. “Performance and cost assessment of machine learning interatomic potentials”. In: *The Journal of Physical Chemistry A* 124.4 (2020), pp. 731–745.
- [13] Aidan P Thompson et al. “Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials”. In: *Journal of Computational Physics* 285 (2015), pp. 316–330.
- [14] M. Wood and A. Thompson. “Extending the accuracy of the SNAP interatomic potential form”. In: *The Journal of Chemical Physics* 148 (2018). DOI: 10.1063/1.5017641.
- [15] A. Rohskopf et al. “FitSNAP: Atomistic machine learning with LAMMPS”. In: *Journal of Open Source Software* 8.84 (2023), p. 5118. DOI: 10.21105/joss.05118. URL: <https://doi.org/10.21105/joss.05118>.
- [16] Ann E Mattsson et al. “Designing meaningful density functional theory calculations in materials science—a primer”. In: *Modelling and Simulation in Materials Science and Engineering* 13.1 (2004), R1.
- [17] Y. Mishin. “Machine-learning interatomic potentials for materials science”. In: *Acta Materialia* 214 (2021). DOI: 10.1016/j.actamat.2021.116980.
- [18] M. Methfessel and A. T. Paxton. “High-precision sampling for Brillouin-zone integration in metals”. In: *Phys. Rev. B* 40 (6 Aug. 1989), pp. 3616–3621. DOI: 10.1103/PhysRevB.40.3616. URL: <https://link.aps.org/doi/10.1103/PhysRevB.40.3616>.
- [19] et. al Sikorski E. “Machine learned interatomic potential for dispersion strengthened plasma facing components”. In: *The Journal of Chemical Physics* 158 (2023). DOI: 10.1063/5.0135269.
- [20] James M Goff et al. “Permutation-adapted complete and independent basis for atomic cluster expansion descriptors”. In: *arXiv preprint arXiv:2208.01756* (2022).

## A PRELIMINARY STUDY FOR OBTAINING INVERSE SOBOLEV-TYPE INEQUALITIES FOR TANH NEURAL NETWORKS

EDWARD HUYNH\*, THEODORE MEISSNER†, PAVEL BOCHEV ‡, AND PAUL KUBERRY §

**Abstract.**

Neural networks (NN) have a potential to generate accurate and computationally efficient solution surrogates for PDEs. These surrogates are obtained by minimizing a loss function comprising residuals of the governing equations and the associated boundary conditions measured in suitable norms. Typically, the same discrete  $L^2$  norm is used for all residuals in the loss function. However, this contradicts the fact that optimal asymptotic accuracy of PDE residual minimization requires  $L^2$  norms weighted by factors that reflect the regularity properties of the underlying problem. This has led to convergence issues and problems with enforcing the boundary conditions in NN surrogates. In the context of least-squares finite elements [3] correct weighting factors for the  $L^2$  residual terms can be determined by using various inverse inequalities that hold for finite element functions. However, such inequalities are not available for neural nets, which hinders the development of robust and accurate NN surrogates. This work represents a step towards building theoretical foundations supporting the formulation of mathematically correct scaling factors for a class of NN functions employing the tanh activation. To that end, we perform both a parameter study and prove several inverse-type Sobolev inequalities for this NN class. We determine the relevant asymptotics relating Sobolev norm ratios for such neural networks and use these results to formulate weighting factors that ensure the same asymptotic behavior for weighted discrete  $L^2$  norms. In so doing we establish a basis for the development or properly weighted loss functions for NN PDE surrogates.

**1. Introduction.** To the best of our knowledge, the idea of using neural networks (NN) to numerically solve partial differential equations was first explored in [6]. Later, Lagaris et al. [9] developed an approach that would later become known as “physics-informed neural network” (PINN) [11]. As this term has become quite common we shall adopt it for use in this work. Recently, due to advances in computational technology, neural networks have resurfaced as a viable scientific computing technique. They have demonstrated a potential for generating accurate and computationally efficient solution surrogates for PDEs [11]. Such surrogates are obtained by first selecting a NN “architecture” defined by the number of layers (depth), the number of nodes within each layer (width), and the activation function applied to each layer. The surrogate PDE solution is then sought as the element of the function class defined by this architecture that minimizes the residuals of the PDEs and the associated boundary conditions, measured in some suitable norms. Typically, the same discrete  $L^2$  norm is used for all residuals in the loss function because this choice leads to simple and differentiable loss function definitions.

However, using the same  $L^2$  norms for all residuals is not necessarily the best choice for numerical methods based on residual minimization. Indeed, mathematically PINN is a least-squares collocation method and as such, its accuracy and convergence properties are subject to the same general mathematical principles that govern the formulation of well-posed least-squares methods for PDEs [3]. In particular, for elliptic PDEs one can use the Agmon-Douglis-Nirenberg (ADN) elliptic regularity theory [1] to determine the appropriate Sobolev norms for each PDE and boundary condition residual in the loss function. Specifically, the ADN theory classifies elliptic PDEs as either homogeneous or inhomogeneous elliptic. One can show that in the former case all PDE residuals can indeed be measured using the same  $L^2$  norm. However, for inhomogeneous elliptic problems a loss function that conforms with the ellipticity type of the underlying PDE would require combinations of  $L^2$  and possibly higher or lower order Sobolev space norms. Moreover, regardless of the ellipticity type, the

---

\*University of Arizona, huynh3@math.arizona.edu

†University of Arizona

‡Sandia National Labs, pbboche@sandia.gov

§Sandia National Labs, pakuber@sandia.gov

boundary condition residual must always be measured in a fractional order norm because it involves traces of Sobolev space functions.

It follows that a loss function that adheres to the correct choice of norms for the equation residuals and the boundary conditions will always necessarily include norms other than the  $L^2$  norm, such as fractional order norms and norms of higher or lower order Sobolev spaces. In the context of least-squares methods based on conventional, mesh-based discretization techniques such norms can always be converted to equivalent weighted  $L^2$  norms scaled by a factor of  $O(h^{-q})$ , where  $h$  is the mesh parameter and  $q$  is the order of the Sobolev space; see, e.g., [2, 3]. Other techniques, based on multilevel techniques and preconditioners are also available; see e.g., [12, 4]. Residual minimization in such properly scaled  $L^2$  norms leads to optimally convergent numerical solutions.

These scaling factors are intimately connected to the structure of the underlying discrete space and especially the fact that it is based on a mesh. As a result, the principles governing the design of weighted  $L^2$  norms for conventional mesh-based discretizations cannot be extended in an obvious manner to PINNs, which involve residual minimization over a class of globally defined mesh-free functions.

At the same time, to improve robustness and accuracy of PINNs it is desirable to develop a more rigorous basis for proper scaling of the individual contributions to the loss function that accounts for the mathematical properties of the underlying PDE. This paper represents a first attempt at this goal and focuses on analysis of the asymptotic behavior of Sobolev space norms restricted to a class of globally defined differentiable functions corresponding to a neural network architecture commonly used in PINNs.

There has been previous work in the literature about approximating functions in different smoothness spaces with either deep RELU or tanh networks such as in [5], [7], and [8]. However, these papers deal primarily with error bounds in Sobolev norms, whereas our main focus is on the development of norm-equivalence relations between  $L^2$  norms and higher order Sobolev norms, restricted to neural network functions. In particular, we are interested in bounding higher-order Sobolev norms on the space of neural networks by properly scaled lower-order Sobolev norms. Similar to the existence of inverse inequalities that are established for finite element functions, we are interested in developing similar inequalities for feed-forward neural networks with tanh activation functions. To the authors' knowledge, there has been no work in establishing inequality estimates of the sort discussed in this manuscript.

Given a specific instance of a NN architecture defined by fixed width and depth parameters, and an activation function, we hypothesize that one can bound the  $H^1$  norm of the functions having this architecture by their properly scaled  $L^2$  norms. This conjecture is rooted in the fact that, although the associated NN space is not finite-dimensional, it is completely characterized by a finite number of parameters. Thus, while in finite element spaces the proper scaling is a function of the mesh size, for NN spaces this scaling is likely to be a function of their parameterization, i.e., the weights and biases. Our main goal is to discover and analyze the structure of these weighting functions for a select class of NN functions.

However, before we commence with the formal analysis and development of inverse type inequalities for NN we shall perform a computational parameter study of ratios of Sobolev space norms restricted to spaces of NN functions with varying numbers of parameters. The goal of this study is to provide numerical validation for the subsequent mathematical theory and to confirm that neural network functions, satisfy inverse-type inequalities with scaling factors that are function of depth and width.

**2. Mathematical Preliminaries.** We start by defining the architecture of feed-forward neural networks. Let  $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$  and let  $\{w_i\}_{i=1}^d$  denote the set of network widths, where  $d \in \mathbb{N}$  is the network depth. Let  $w_1 = n$ . For  $1 \leq i \leq d$ , let  $A_i \in \mathbb{R}^{w_{i+1} \times w_i}$ ,  $b_i \in \mathbb{R}^{w_{i+1}}$  and define  $f_i : \mathbb{R}^{w_i} \rightarrow \mathbb{R}^{w_{i+1}}$  as

$$f_i(\mathbf{z}) = \sigma(A_i \mathbf{z} + b_i), \quad \forall \mathbf{z} \in \mathbb{R}^{w_i},$$

where  $\sigma(\mathbf{z}) = (\sigma(z_1), \sigma(z_2), \dots, \sigma(z_{w_i}))$  and  $\sigma : \mathbb{R} \rightarrow \mathbb{R}$  is a nonlinear activation function.

Finally, let  $A_{d+1} \in \mathbb{R}^{1 \times w_d}$ ,  $b_{d+1} \in \mathbb{R}$  and define  $g : \mathbb{R}^{w_d} \rightarrow \mathbb{R}$  as

$$g(\mathbf{z}) = A_{d+1} \mathbf{z} + b_{d+1}.$$

The widths, the depth and the activation function  $\sigma$  define a network architecture  $(d, \{w_i\}, \sigma)$ , while the matrices  $A_i$  and the vectors  $b_i$  are the parameters of the neural network that specify an instance of a (scalar-valued) neural network function  $f_w^d : \mathbb{R}^n \rightarrow \mathbb{R}$  defined as

$$f_w^d(\mathbf{x}) := (g \circ f_d \circ f_{d-1} \circ \dots \circ f_2 \circ f_1)(\mathbf{x}). \quad (2.1)$$

The set of all neural network functions (2.1) is an infinite dimensional function space characterized by a finite number of parameters.

Popular choices for the activation function include ReLu, Swish, and Sigmoid. However, since our focus is on inverse-type inequalities that can be used to determine proper scaling of the loss function terms in the context of PINNs, we will assume an activation function that is smooth so that  $f_w^d$  is also smooth. For our study we will select  $\sigma = \tanh$  and assume  $w_i = w$  for all  $1 \leq i \leq d$ . For notational clarity we denote the associated network architecture simply by  $(d, w)$ . We will then analyze the relationship between Sobolev space norms restricted to the class of neural functions (2.1) having the architecture  $(d, w)$ . Our goal will be to determine the norm-equivalence constants between these norms as a function of the architecture parameters  $d$  and  $w$ .

We recall the Sobolev space  $W_{k,p}(\Omega)$  of functions having  $k$  (weak) derivatives which are elements of  $L^p(\Omega)$ . We are primarily interested in the case  $p = 2$ , for which  $W_{k,p}(\Omega)$  are Hilbert spaces equipped with norms  $\|\cdot\|_{k,\Omega}$ . When  $k = 1$  we have the Sobolev space  $H^1$  with norm  $\|\cdot\|_{1,\Omega}$  defined as

$$\|f\|_{1,\Omega} := \|f\|_{0,\Omega} + \|\nabla f\|_{0,\Omega}.$$

The term  $\|\nabla f\|_{0,\Omega}$  is also called the  $H^1$  semi-norm of  $f$ .

**2.1. Mathematical basis for optimally accurate residual minimization methods.** Development of optimally accurate residual minimization methods for conventional mesh-based discretizations relies on the Agmon-Douglis-Nirenberg (ADN) theory [1] to identify the correct combinations of Sobolev norms in which to measure the residuals of the equations and the boundary conditions. While in the context of NN functions we cannot use the mesh-based scaling factors from these methods, the correct norm combinations remain in full force for residual minimization methods based on neural network functions such as PINNs. In particular, these norm combinations should be incorporated in the design of the PINN loss function. To state the relevant result from the ADN theory assume that  $\Omega$  is a bounded open region in  $\mathbb{R}^n$  with a Lipschitz continuous boundary  $\Gamma = \partial\Omega$  and consider the boundary value problem

$$\begin{aligned} \mathcal{L}(u) &= f, & x \in \Omega \\ B(u) &= g, & x \in \partial\Omega, \end{aligned} \quad (2.2)$$

where  $\mathcal{L} = \{\mathcal{L}_{ij}\}$ ,  $i, j = 1, \dots, N$  is a linear differential operator and  $B = \{B_{lj}\}$ ,  $l = 1, \dots, m$ ,  $j = 1, \dots, N$  is a boundary operator. In this paper we restrict attention to operators  $\mathcal{L}$  that are elliptic in the sense of ADN; see [1]. For simplicity we assume that (2.2) has zero nullity. Then, assuming sufficient solution regularity, the ADN theory implies the existence of a constant  $C$  such that

$$\sum_{j=1}^N \|u_j\|_{q+t_j, \Omega} \leq C \left( \sum_{i=1}^N \left\| \sum_{j=1}^N L_{ij} u_j \right\|_{q-s_i, \Omega} + \sum_{l=1}^m \left\| \sum_{j=1}^N B_{lj} u_j \right\|_{q-r_l-1/2, \Gamma} \right). \quad (2.3)$$

where  $s_i \leq 0$ ,  $t_j \geq 0$ , and  $r_l \leq 0$  are the ADN equation, variable and boundary condition indices and  $q \geq 0$  is the solution regularity index. The problem (2.2) is homogeneous elliptic if (2.3) holds with  $t_j = 1$ ,  $s_i = 0$  and  $r_l = 0$ . Otherwise, (2.2) is inhomogeneous elliptic.

The significance of (2.3) for residual minimization methods stems from the fact that its right hand side prescribes the norms that would result in a norm-equivalent loss functional. For example, setting  $q = 0$  it is easy to see that for a homogeneous elliptic system with Dirichlet boundary conditions a norm-equivalent loss functional is given by

$$J(u) = \left( \sum_{i=1}^N \left\| \sum_{j=1}^N L_{ij} u_j - f_i \right\|_{0, \Omega}^2 + \sum_{l=1}^m \left\| \sum_{j=1}^N u_j - g_l \right\|_{1/2, \Gamma}^2 \right). \quad (2.4)$$

All equation residuals in this case can be measured using the  $L^2$  norm but the boundary condition residual requires a fractional Sobolev space norm. In addition to fractional norms for the boundary terms a norm-equivalent loss function for inhomogeneous elliptic problem also requires some equation residuals to be measured in the  $H^1$  norm or seminorm.

One key issue with such loss functions, that remains in full force for PINNs as well, is that fractional order norms are not easily computable. In mesh-based residual minimization methods this problem can be resolved by replacing  $\|\cdot\|_{1/2, \Gamma}$  with the weighted  $L^2$  norm  $h^{-1/2} \|\cdot\|_{0, \Gamma}$ . One can show that minimization of the resulting weighted  $L^2$  loss function yields optimally accurate discrete solutions. Similarly, for inhomogeneous elliptic problems one can replace  $\|\cdot\|_{1, \Omega}$  by the weighted  $L^2$  norm  $h^{-1} \|\cdot\|_{0, \Omega}$ .

Such substitutions rely on results collectively known as inverse inequalities, which are a consequence of the fact that all norms are equivalent in finite dimensional normed vector spaces. For mesh-based discretizations one can show that the equivalence constant is a function of the mesh size. Although the set of all neural network functions with a fixed architecture  $(d, w)$  is not a finite dimensional space, it is completely described by a finite number of parameters. Thus, we conjecture that similar inverse inequalities should also hold for neural network functions. Also, by analogy with mesh-based methods, where the mesh size  $h$  is proportional to the dimension of the discrete vector space, we conjecture that for neural network functions the equivalence constant will be a function of the network architecture  $(d, w)$  because it governs the total number of parameters defining each neural network function.

In this work we focus on the more general question of proving inequalities of the form

$$\|u_w^d\|_X \leq C \|u_w^d\|_Y, \quad (2.5)$$

where  $u_w^d$  is a neural network with a given architecture  $(d, w)$  and  $X$  and  $Y$  are appropriate function spaces. If  $X = L^2(\Omega)$  and  $Y = H^1(\Omega)$ , then under suitable conditions (2.5) becomes the statement for *Poincaré's inequality*. On the other hand, letting  $X := H^1(\Omega)$  and  $Y := L^2(\Omega)$  yields an *inverse inequality*. In the next section we perform a parameter study

aiming to illustrate (2.5) numerically. Although we perform this study with tanh activation functions, the subsequent analysis in this paper applies to the class of trigonometric activation functions such as  $\sin(\cdot)$ ,  $\cos(\cdot)$ .

### 3. Parameter Study.

**3.1. Methodology.** In our study, we consider  $w = 2, 3, \dots, 20$  and  $d \in \{1, 2, 3, 4, 5\}$ , i.e. we examine networks defined on  $\Omega = [0, 1]^2 := [0, 1] \times [0, 1]$  with up to 5 layers and up to 20 neurons. For each architecture  $(d, w)$ , we sample all the values of the parameters uniformly from  $Unif(-1, 1)$ . There has been previous work in sampling instead from a normal distribution, but we have opted to choose only values that lie in  $[-1, 1]$ . Since it is important to explore what happens to the values of the Sobolev norms, it is crucial to observe the effects of having both positive and negative weights. Moreover, the interval  $[-1, 1]$  is a good starting point for roughly understanding the behavior of these norms, as increasing the range of values will approximately scale the norms up to a multiplicative constant in the matrix norms. For each sampled neural network function we use numerical integration to approximate its  $L^2$  norm,  $H^1$ -norm and  $H^1$  seminorm, denoted by  $\|\cdot\|_0$ ,  $\|\cdot\|_1$ , and  $|\cdot|$ , respectively. In particular, we use a composite trapezoidal rule with 100 nodes in each coordinate direction, for a total of 10000 points in  $\Omega$ .

While one can use a more accurate numerical integration rule, a second order rule is more than sufficient for our study which focuses on the statistics of norms and their ratios rather than their accurate values for specific function arguments. On a practical level, choosing other quadratures may also lead to longer computational times, so we opted for a balance of feasibility with accuracy. This procedure is performed for each architecture  $(d, w)$  100 times. Figures 3.1 summarize the distribution of values for the  $L^2$ -norm, the seminorm, the ratio of the seminorm to the  $L^2$  norm (which we will call the *Sobolev ratio*), and the ratio of the Sobolev norm to the  $L^2$  norm. We construct surface plots of these for both the mean and median values for each quantity in Figure 3.2. There were trials that yielded very large values in the semi-norm, and therefore yielding large values for the Sobolev ratio. We consider these to be outliers and therefore opt to remove these from the analysis. While the median is generally more robust to outliers, since we removed the outliers from our graph and included only the points within the 1% – 99% percentiles, we find the mean and median values do not differ much from each other. Thus we chose to analyze the mean ratio values by considering the plots of the curves as a function of width for each layer. These plots are summarized in Figure 3.5. In each of these plots, we performed curve fitting with a linear function, a quadratic function, and a cubic function and listed the resulting equations along with their  $R^2$  values. We did a similar analysis by plotting the curves as a function of layer for each number of neurons. These are plotted in Figures 3.6. In addition we plot the behavior as a function of the number of parameters in each neural network 3.3. To compare the behavior of the number of layers to the number of neurons, we rescale the y-axis to the maximum in Figure 3.4. Similarly, we also rescale the dimension of the y-axis to compare behavior across number of neurons in Figure 3.7.

**3.2. Analysis.** When we fix the number of hidden layers and vary the number of neurons from  $w = 2$  to  $w = 20$ , we find that not only do the  $L^2$  norms increase, but the semi-norms also increase. However, the semi-norms increase much faster compared to the  $L^2$  norms, which is reflected in the increasing Sobolev ratios and in the ratio of  $H_1$  to  $L^2$  (Figure 3.1). Interestingly, this pattern is observed uniformly across the number of layers except that the value of the semi-norms increases by approximately an order of 10. Intuitively, the larger semi-norms are expected since increasing the number of neurons per layer results in neural network functions that can support more oscillatory behavior. On

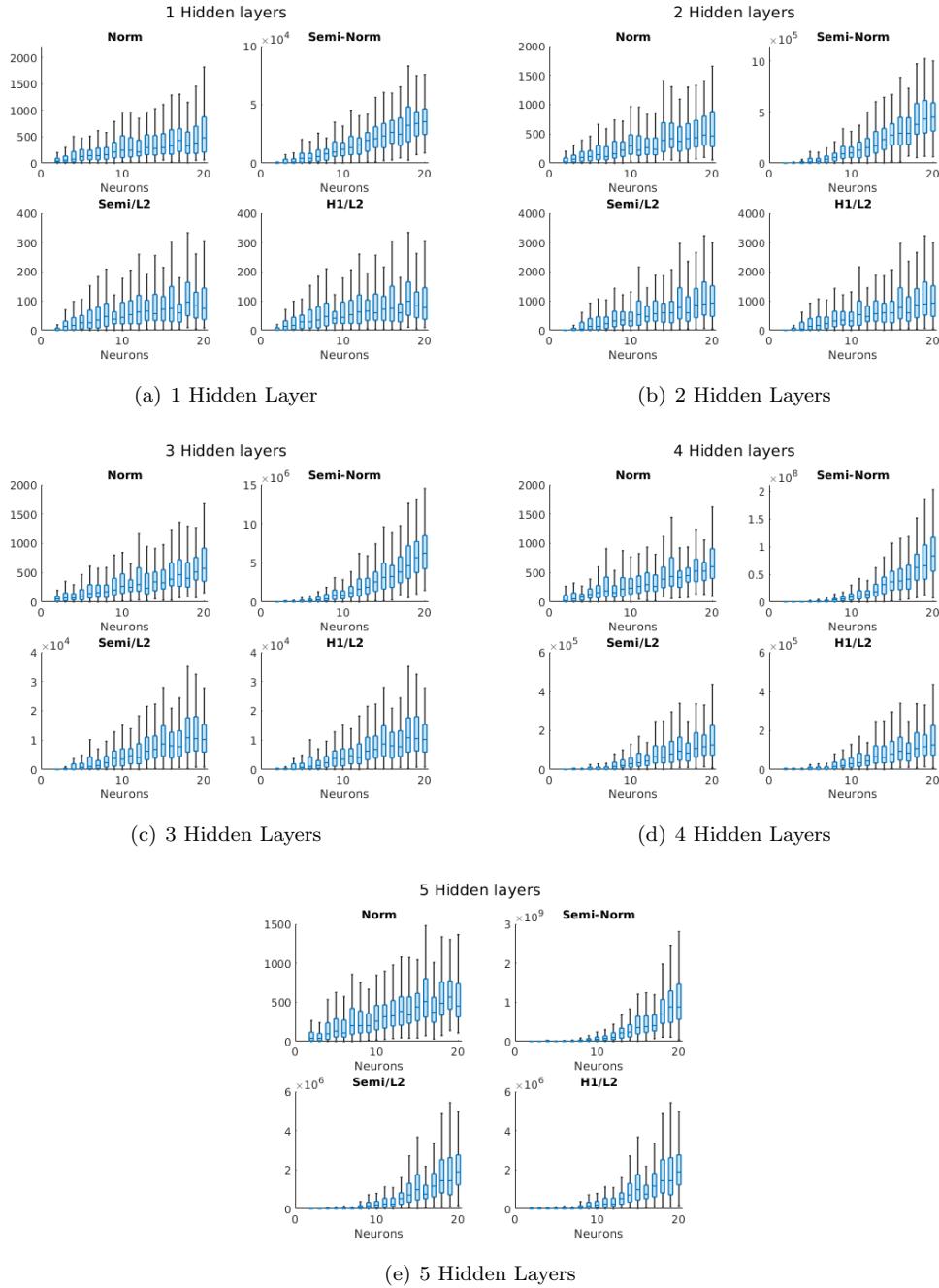


FIG. 3.1. *Boxplots of Norms and Relevant Quantities with 1 – 5 Hidden Layers*

the other hand, fixing the number of neurons  $w$  while increasing the number of layers means adding an additional layer with  $w$  neurons, which in light of the previous explanation should also yield an increase in the semi-norm.

To visualize the magnitude of these increases, we plot surface plots of the Sobolev ratio

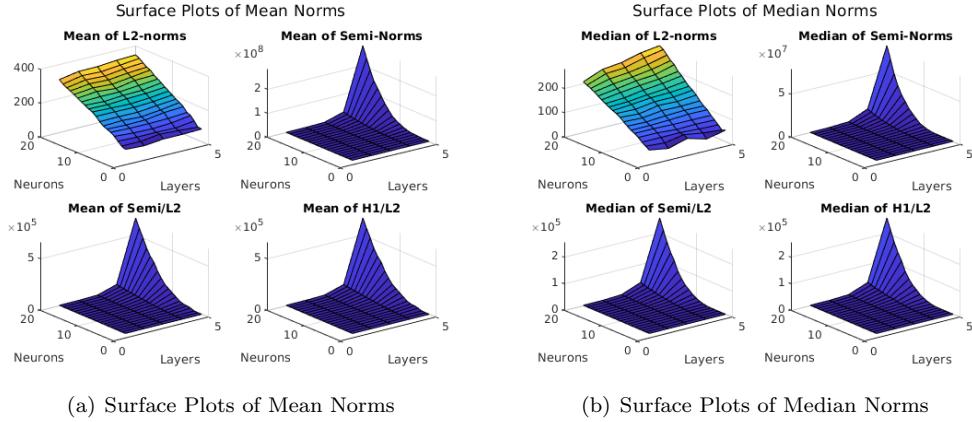


FIG. 3.2. *Surface Plots of Relevant Quantities by Number of Layers ( $d$ ) and Neurons ( $w$ )*

as a function of the number of layers and the number of neurons per layer in Figure (3.2). We note that the  $L^2$  norms of the neural networks increase in both the number of layers and the number of neurons. Interestingly, the value of the semi-norm increases almost exponentially once  $d = 5$ . Relative to the increase in the  $L^2$  norm, the change in the value of the semi-norm drastically changes from  $d = 4$  to  $d = 5$ .

Next, we plot the Sobolev ratio as a function of the total number of parameters (Figure (3.3)). Across the number of layers, we notice increases in the value of the ratio as the number of neurons are increased (thereby increasing the number of parameters). At first, the ratio curve looks concave and progresses slowly into a convex function. In addition, we plot the same quantity except as a function of the number of neurons and scaled so that all the plots are on the same scale in Figure (3.4). Relative to the larger number of layers, a lower number of layers ( $d = 1, 2, 3$ ) appear flat (although they are not constant). However, once  $d = 4$ , the curve is noticeable and visually looks more like an exponential function by  $d = 5$ . It is understandable that this increase occurs, since if we fix  $w = 20$ , adding an additional layer means approximately an increase of 400 parameters. This allows for extra variation in the number of oscillations, thereby exaggerating an increase in the semi-norm.

We perform curve fitting for Sobolev ratio as a function of the number of neurons for each layer in Figure 3.5. The plots in this figure compare linear, quadratic and cubic regression of the data and state the model fit in terms of  $R^2$ . We note that for  $d = 1, 2, 3$  the linear regression already yields  $R^2 > 0.96$ , which suggests that for relatively shallow neural network functions the Sobolev ratio is roughly linear. This observation is reaffirmed by noting that in the quadratic and cubic approximations of the data the coefficients of the quadratic and the cubic terms are relatively small.

However, as the network functions become deeper, i.e., as the parameter  $d$  becomes larger, the contributions from the higher order terms become more prominent. In particular, when  $d = 5$  the quadratic and cubic regressions are dominated by the highest order term. This further corroborates the observation that the Sobolev ratio grows exponentially with the depth of the neural network architecture.

Finally, we plot the values of the Sobolev ratios as a function of the depth for each width ( $w = 2, \dots, 20$ ) in Figures (3.6) and (3.7). There is no discernable pattern when  $w = 2, 3$ , but the exponential nature of the ratio becomes more apparent as the number of neurons increase. It is interesting that this exponential behavior is observed uniformly across each plot. To see the effect of growth, we observe in Figure (3.7) that the plots appear flat for

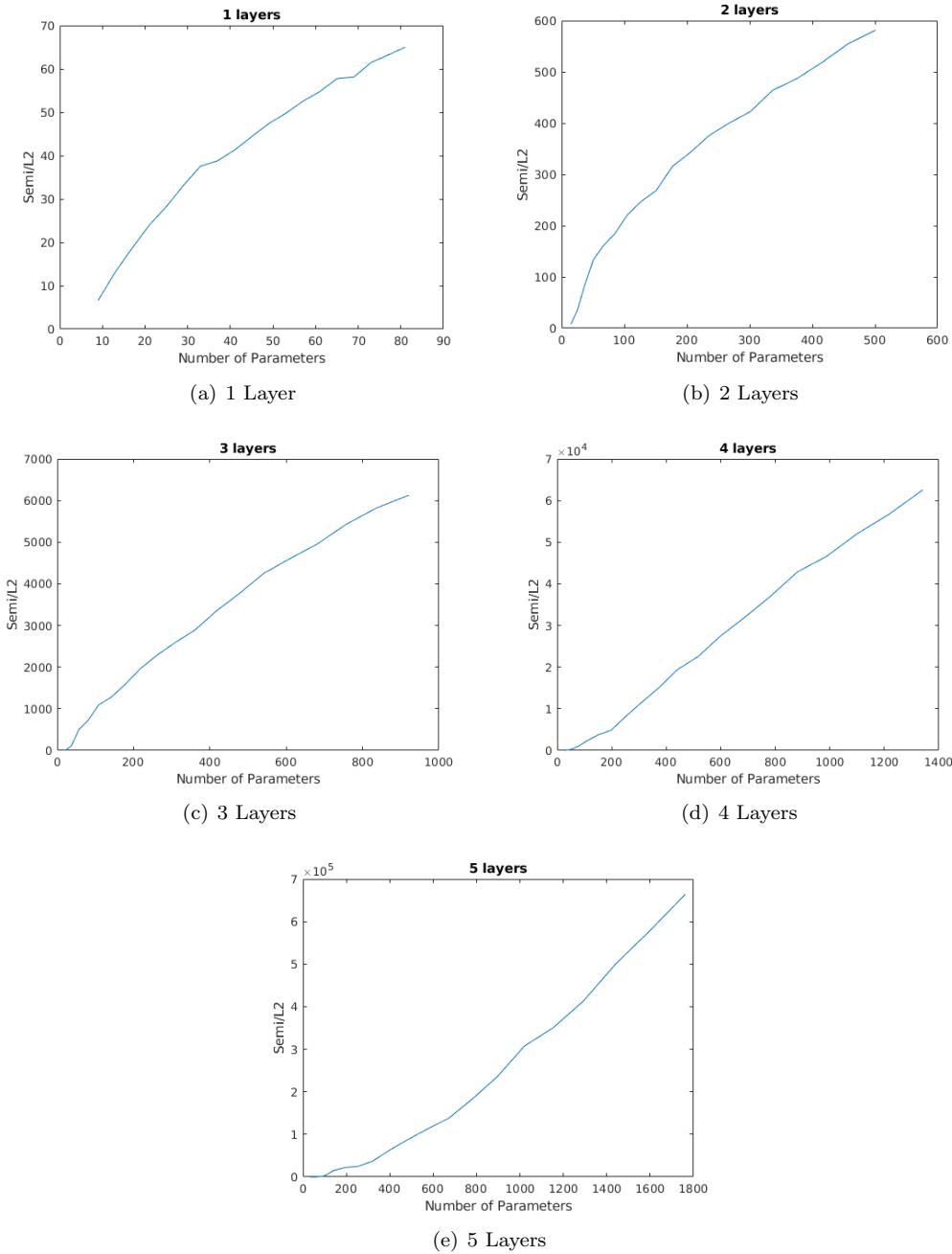


FIG. 3.3. Plots of Sobolev Ratio  $\frac{\|\nabla f\|_{L^2}}{\|f\|_{L^2}}$  by Number of Layers (d) and Total Number of Parameters

smaller number of neurons. As the number of neurons increases (around  $w = 10$ ), the plots evolve into a function that exhibits exponential-like growth.

**4. Inverse-type Bounds.** In this section, we obtain several results for different architectures of networks based on number of layers, number of neurons, and spatial dimension.

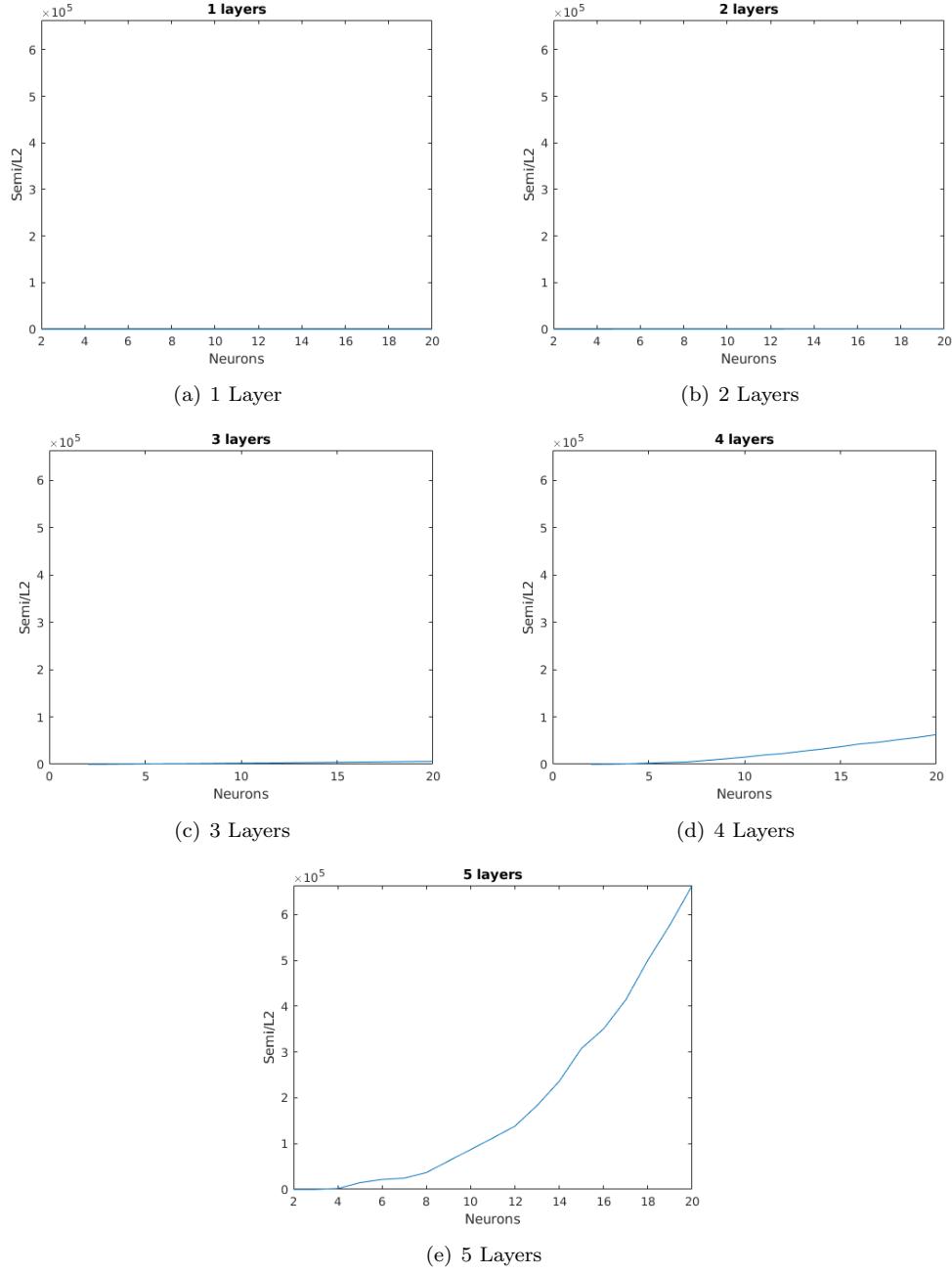


FIG. 3.4. Rescaled Plots of Sobolev Ratio by Layer

These will include expressions for a bounding constant as well as asymptotics based on the scaling coefficients. The latter reveal the behavior of the quantities of interest as their argument  $a$  tends to infinity. For example, we say that  $f(a)$  is asymptotically equivalent to  $g(a)$  if  $\lim_{a \rightarrow \infty} (f(a)/g(a)) = C$ . The earlier results will help to build intuition for when we prove the asymptotics for the most general case. For ease of notation, we will write  $x$

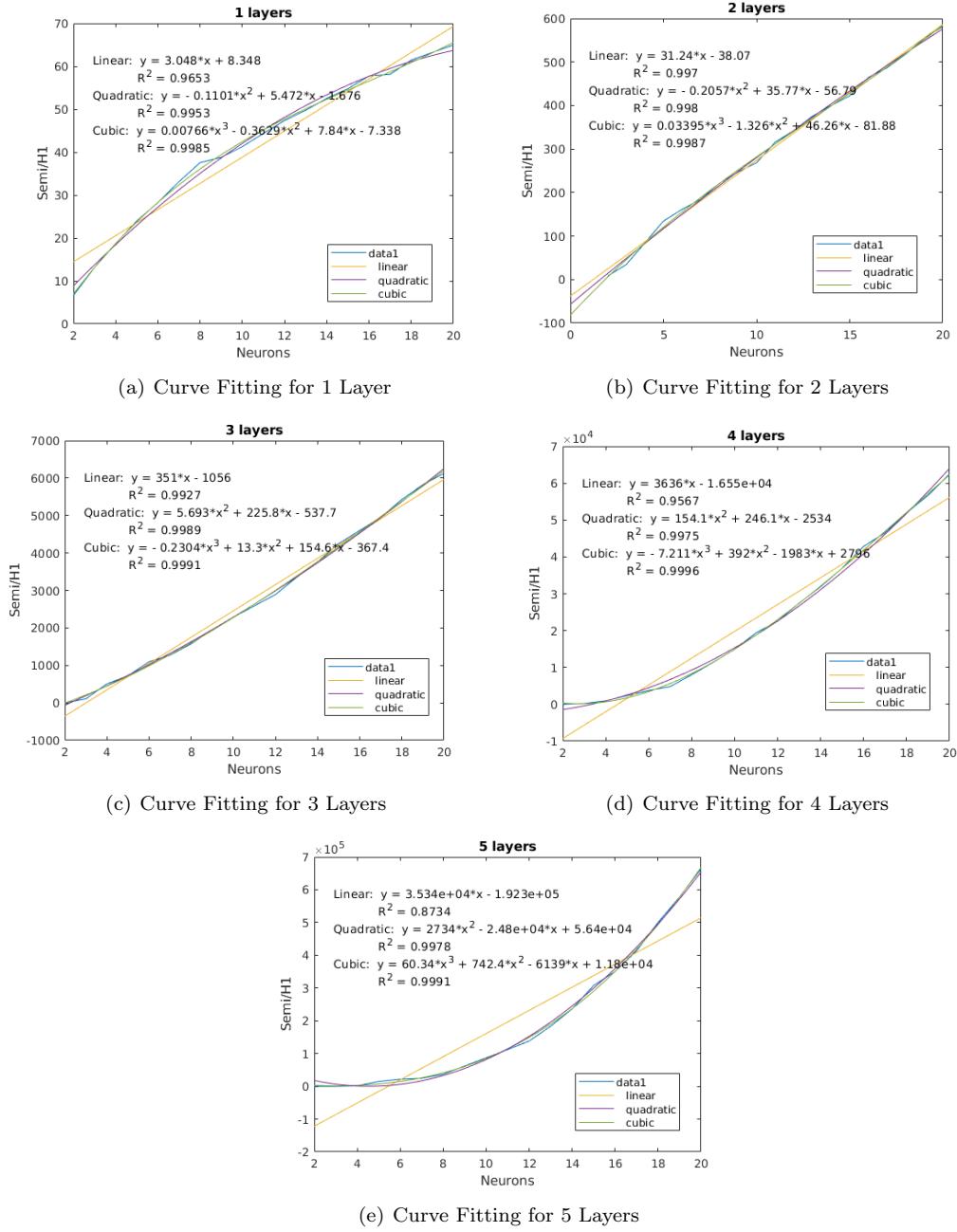


FIG. 3.5. *Curve Fitting for Sobolev Ratio  $\frac{\|\nabla f\|_{L^2}}{\|f\|_{L^2}}$  by Number of Layers (d)*

as simply  $x$ , with the understanding that  $x$  is an element of  $\mathbb{R}^n$ . Furthermore, when  $A$  is a vector (or matrix), unless otherwise stated,  $\|A\|$  is taken to be the standard Euclidean norm (or Frobenius norm) of the vector (matrix).

To begin our study, we consider the case when  $f(x) \equiv f_1(x)$  and  $x \in \mathbb{R}$ . This case will

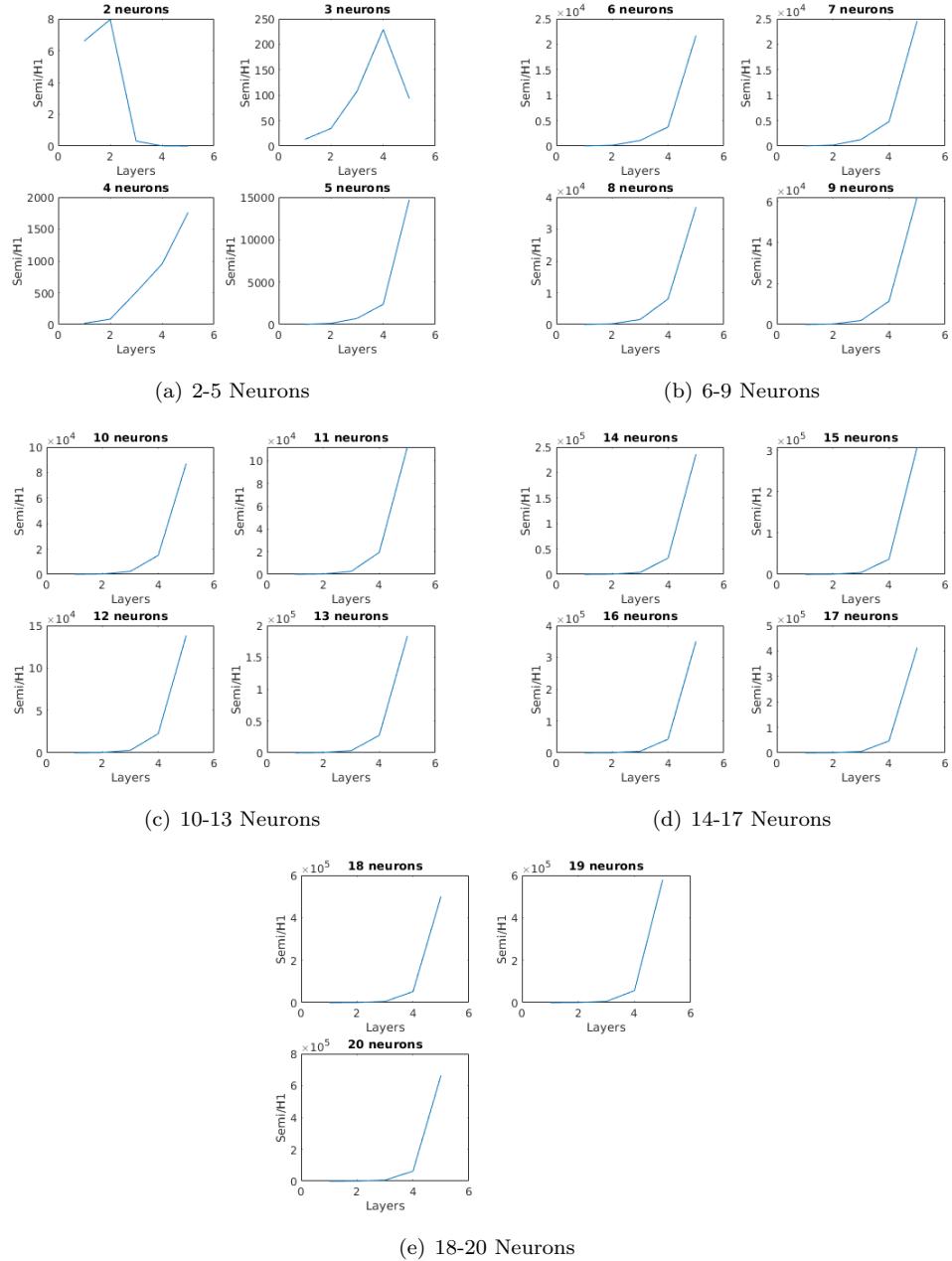


FIG. 3.6. Plots of Sobolev Ratio  $\frac{\|\nabla f\|_{L^2}}{\|f\|_{L^2}}$  as a function of the depth for subsets of  $(w)$

be instructive when we compare the asymptotics. In this case we have:

**PROPOSITION 1.** *Let  $\Omega = [x_0, x_1]$  and  $f(x) = \tanh(ax+b)$ . Then there exists a constant  $C(a, \Omega) \in \mathbb{R}$  such that*

$$\|f'\|_0 \leq C(a, \Omega) \|f\|_0.$$

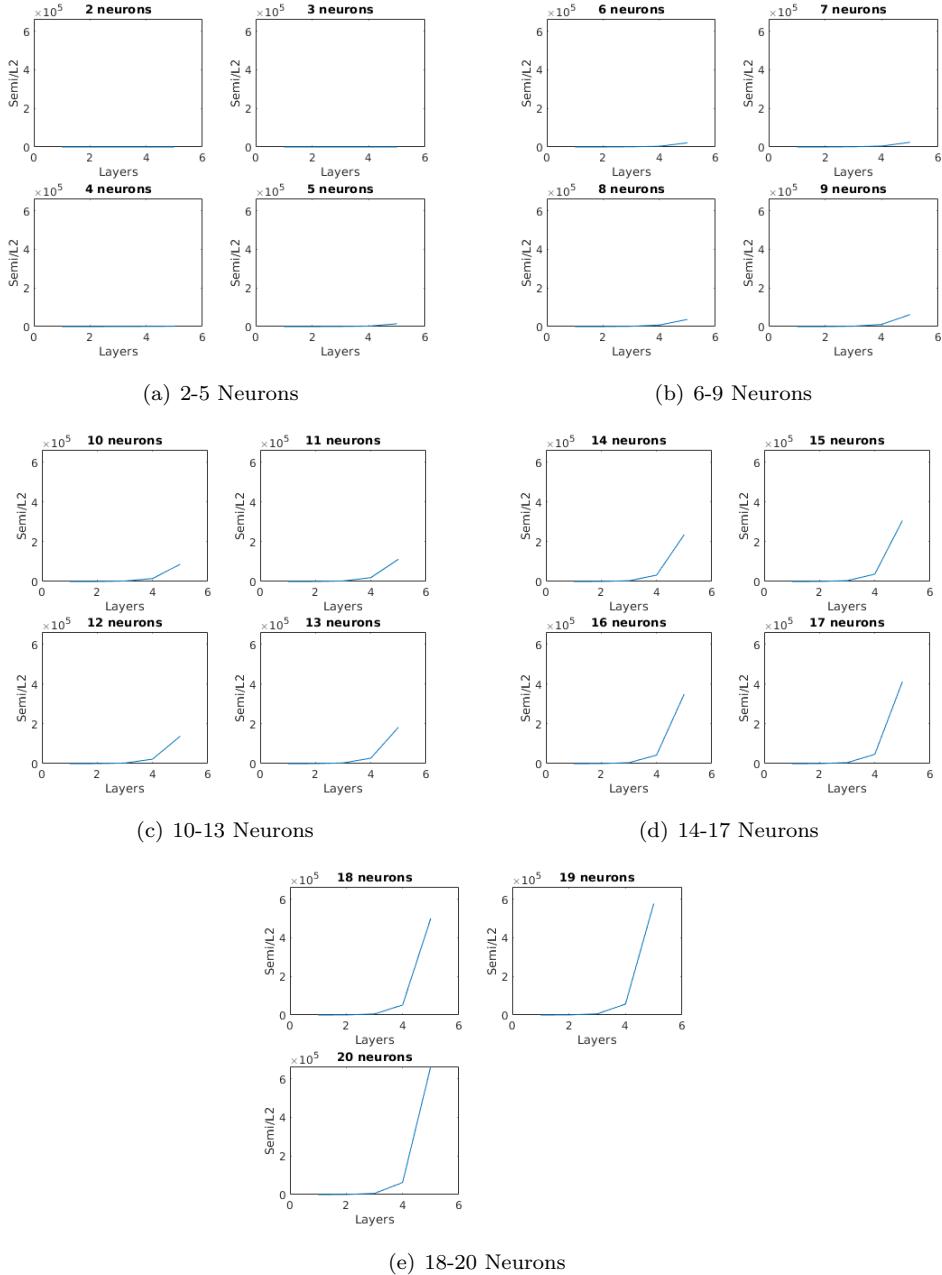


FIG. 3.7. Rescaled Plots of Sobolev Ratio  $\frac{\|\nabla f\|_{L^2}}{\|f\|_{L^2}}$  by Number of Neurons Per Layer ( $w$ )

*Proof.* We observe that

$$f(x) = \tanh(ax + b), \quad f'(x) = \operatorname{sech}^2(ax + b),$$

$$\begin{aligned}\|f\|_0^2 &= \int_{x_0}^{x_1} (\tanh(ax+b))^2 dx \\ &= \frac{a(x_1 - x_0) + \tanh(ax_0 + b) - \tanh(ax_1 + b)}{a}, \\ \|f'\|_0^2 &= \int_{x_0}^{x_1} (a \operatorname{sech}^2(ax+b))^2 dx \\ &= a(\tanh(ax_1 + b) - \tanh(ax_0 + b)) + \frac{a}{3}(\tanh^3(ax_0 + b) - \tanh^3(ax_1 + b)).\end{aligned}$$

It follows that

$$\|f'\|_0^2 = C^2 \|f\|_0^2$$

where,

$$C^2 = \frac{a^2 (\tanh(ax_1 + b) - \tanh(ax_0 + b) + \frac{1}{3}(\tanh^3(ax_0 + b) - \tanh^3(ax_1 + b)))}{a(x_1 - x_0) + \tanh(ax_0 + b) - \tanh(ax_1 + b)}.$$

Taking the square root of this expression yields the desired conclusion.  $\square$

REMARK 1. *From Proposition 1, we have asymptotically in a that*

$$C \sim \sqrt{\frac{|a|}{x_1 - x_0}}.$$

*Intuitively, we should expect that the Sobolev constant depends on the weights of the network from the fact that the weights control how large the semi-norm can be, and therefore the upper bound on the ratio. This is an allusion to our parameter study in the previous section. It also implies that when considering weighing certain terms of the loss function, we should expect to weigh the  $H_1$  norms proportional to the magnitude of the weights.*

Returning to tanh neural networks, we will now extend the result in 1 dimension to a general number of dimensions  $n$ . Before doing this, we state a lemma which will be used in our future results:

LEMMA 1. *Let  $g(x) = 0.1x^2 \mathbb{1}_{|x| \leq 3}(x) + 0.9\mathbb{1}_{|x| > 3}(x)$ . Then*

$$\tanh^2(x) \geq g(x), \quad \forall x \in \mathbb{R}.$$

PROPOSITION 2. *Let  $\Omega \subset \mathbb{R}^n$  be a compact set with positive measure and  $f(x) = \tanh(a^T x + b)$  for  $x, a \in \mathbb{R}^n$  and  $b \in \mathbb{R}$ . Then there exists a constant  $C := C(n, a, b, \Omega) > 0$  such that*

$$\|\nabla f\|_0 \leq C \|f\|_0.$$

*Proof.* Let  $f(x) = \tanh(a^T x + b)$ . Then  $\nabla f = (a_i \operatorname{sech}^2(a^T x + b))_{i=1}^n$ . Note that

$$\|\nabla f\|_0^2 = \int_{\Omega} \left( \sum_{i=1}^n a_i^2 \right) \operatorname{sech}^4(a^T x + b) dx \leq m(\Omega) \sum_{i=1}^n a_i^2.$$

Next, note that

$$(a^T x + b)^2 = \langle (a, b), (x, 1) \rangle^2 \leq (|a|^2 + b^2)(1 + \|x\|^2).$$

Next, using Lemma 1 we have  $g(x) \leq \tanh^2(x)$  for all  $x$ . Then

$$\int_{\Omega} \tanh^2(a^T x + b) dx \geq \int_{\Omega} 0.1(a^T x + b)^2 \mathbb{1}_{|a^T x + b| \leq 3}(x) dx + \int_{\Omega} 0.9 \mathbb{1}_{|a^T x + b| > 3}(x) dx.$$

We therefore have

$$\|\nabla f\|_0^2 \leq m(\Omega) \|a\|^2 \leq \frac{m(\Omega) \|a\|^2}{\int_{\Omega} 0.1(a^T x + b)^2 \mathbb{1}_{|a^T x + b| \leq 3}(x) dx + \int_{\Omega} 0.9 \mathbb{1}_{|a^T x + b| > 3}(x) dx} \|f\|_0^2.$$

Letting  $C = \left( \frac{m(\Omega) \|a\|^2}{\int_{\Omega} 0.1(a^T x + b)^2 \mathbb{1}_{|a^T x + b| \leq 3}(x) dx + \int_{\Omega} 0.9 \mathbb{1}_{|a^T x + b| > 3}(x) dx} \right)^{1/2}$  finishes the proof.  $\square$

In fact, we are able to extract the asymptotics as a function of  $\|a\|$  from the constant.

**REMARK 2.** *The following corollary is a preliminary result and provides a coarse upper bound for the Sobolev ratio. This is only an initial result. We believe it can be improved and will focus on this in future work.*

**COROLLARY 1.** *We have  $\frac{\|\nabla f\|}{\|f\|} \sim \sqrt{\|a\|^n}$ .*

*Proof.* Note that  $(a^T x + b)^2 = O(\|a\|^2)$  and  $\int_{\Omega} \mathbb{1}_{|a^T x + b| \leq 3}(x) dx = O(1/\|a\|^n)$ . To prove the first claim, we note that by triangle inequality

$$\frac{|a^T x + b|}{\|a\|} \leq \frac{|a^T x|}{\|a\|} + \frac{|b|}{\|a\|},$$

so it suffices to show that  $|a^T x|/\|a\|$  is bounded in  $a$ . By Cauchy-Schwarz we have

$$\frac{|a^T x|}{\|a\|} \leq \|x\|,$$

which shows that  $|a^T x| = O(\|a\|)$ . The first statement therefore follows.

Next, using the first statement we have

$$\int_{\Omega} (a^T x)^2 \mathbb{1}_{|a^T x + b| \leq 3} dx \leq \|a\|^2 \int_{\Omega} \|x\|^2 \mathbb{1}_{|a^T x + b| \leq 3} dx \leq \|a\|^2 \left( \max_{\Omega} \|x\|^2 \right) m(\Omega \cap \{|a^T x + b| \leq 3\}).$$

We note that if  $x \in \Omega \cap \{|a^T x + b| \leq 3\}$ , then  $x \in \Omega$  and satisfies:

$$|a^T x + b| \leq 3 \Leftrightarrow \left| \frac{a^T x}{\|a\| \|x\|} \|x\| + \frac{b}{\|a\|} \right| \leq \frac{3}{\|a\|} \Leftrightarrow |O(1)\|x\| + o(1)| \leq \frac{3}{\|a\|}.$$

This implies that  $m(\Omega \cap \{|a^T x + b| \leq 3\})$  is asymptotically equivalent to the measure of a ball of radius  $1/\|a\|$ . In particular, since  $\Omega$  is a fixed finite domain then

$$m(\Omega \cap \{|a^T x + b| \leq 3\}) = O\left(m(B_{\frac{1}{\|a\|}}(0))\right) = O\left(\frac{1}{\|a\|^n} m(B_1(0))\right) = \frac{1}{\|a\|^n} O(1).$$

Thus, we have for  $\|a\|$  large:

$$\int_{\Omega} 0.1(a^T x + b)^2 \mathbb{1}_{|a^T x + b| \leq 3}(x) dx = O(\|a\|^{2-n}).$$

From our previous argument, we also have

$$\lim_{\|a\| \rightarrow \infty} m(\Omega \cap \{|a^T x + b| \leq 3\}) = 0.$$

Then we have

$$\lim_{\|a\| \rightarrow \infty} m(\Omega \cap \{|a^T x + b| > 3\}) = m(\Omega) - \lim_{\|a\| \rightarrow \infty} m(\Omega \cap \{|a^T x + b| \geq 3\}) = m(\Omega).$$

It follows that  $\int_{\Omega} 0.9 \mathbb{1}_{|a^T x + b| > 3}(a^T x + b) dx \rightarrow \int_{\Omega} 0.9 dx = 0.9m(\Omega)$  as  $\|a\| \rightarrow \infty$ . This implies that for large  $\|a\|$ , we have

$$\left( \frac{m(\Omega)\|a\|^2}{\int_{\Omega} 0.1(a^T x + b)^2 \mathbb{1}_{|a^T x + b| \leq 3}(x) dx + \int_{\Omega} 0.9 \mathbb{1}_{|a^T x + b| > 3}(x) dx} \right) \sim \frac{m(\Omega)\|a\|^2}{\|a\|^{2-n}O(1) + 0.9m(\Omega) + o(\|a\|)} \sim \|a\|^n.$$

Subsequently taking the square root yields the result.  $\square$

**REMARK 3.** Note that the asymptotics with respect to  $a$  are the same in Proposition 1 and in Corollary 1 when  $n = 1$ .

Next, let us consider what happens when we vary the number of neurons in a single hidden layer. This yields the following proposition:

**PROPOSITION 3.** Let  $\Omega \subset \mathbb{R}^n$  be a compact domain with positive measure and define

$$f(x) = \sum_{i=1}^w \tanh(a_i^T x + b_i)$$

where  $a_i \in \mathbb{R}^n$  and  $b_i \in \mathbb{R}$  for each  $i$ .

Then there exists  $C := C(n, a_i, b_i, \Omega) > 0$  such that

$$\|\nabla f\|_0 \leq C\|f\|_0.$$

*Proof.* Note that  $\frac{\partial f}{\partial x_j} = \sum_{i=1}^w a_{ij} \operatorname{sech}^2(a_i^T x + b_i)$ , where  $a_{ij}$  is the  $j$ th component of  $a_i$ . Then using the inequality  $2|ab| \leq a^2 + b^2$  we have

$$\|\nabla f\|_0^2 = \sum_{j=1}^n \int_{\Omega} \left( \sum_{i=1}^w a_{ij} \operatorname{sech}^2(a_i^T x + b_i) \right)^2 dx \leq \sum_{j=1}^n \int_{\Omega} 2 \sum_{i=1}^w a_{ij}^2 \operatorname{sech}^4(a_i^T x + b_i) dx.$$

Next, using the fact that  $\operatorname{sech}^4(x) \leq 1$  for all  $x$ , we have

$$\|\nabla f\|_0^2 \leq 2m(\Omega) \sum_{j=1}^n \sum_{i=1}^w a_{ij}^2 = 2m(\Omega)\|A\|^2,$$

where  $A = [a_1, \dots, a_w]$  denotes the matrix with columns equal to  $a_i$ .

Using Lemma 1, we have

$$\begin{aligned} \left( \sum_{i=1}^w \tanh(a_i^T x + b_i) \right)^2 &= \sum_{i=1}^w \tanh^2(a_i^T x + b_i) + 2 \sum_{i \neq j} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) \\ &\geq \sum_{i=1}^w \left( 0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3}(x) \right) + 2 \sum_{i \neq j} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) \end{aligned}$$

Using this, we have

$$\|\nabla f\|_0^2 \leq \frac{2m(\Omega)\|A\|^2}{\int_{\Omega} \sum_{i=1}^w \left( 0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3}(x) \right) + 2 \sum_{i \neq j} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) dx} \|f\|^2.$$

We then have

$$C = \left( \int_{\Omega} \sum_{i=1}^w (0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3}(x)) + 2 \sum_{i \neq j} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) dx \right)^{1/2}$$

is the desired constant.

□

Similarly, we may obtain an asymptotic:

COROLLARY 2. We have  $\frac{\|\nabla f\|_0}{\|f\|_0} \sim \sqrt{\|A\|^n}$ .

*Proof.* Using the proof of Corollary 1, we have for each  $i$

$$\int_{\Omega} 0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3} dx = \|a_i\|_2^{2-n} O(1).$$

In particular, we have

$$\int_{\Omega} \sum_{i=1}^w 0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3} dx = \left( \max_{1 \leq i \leq w} \|a_i\|_2 \right)^{2-n} O(1).$$

If we let  $A = (a_i)_{i=1}^n \in \mathbb{R}^{w \times n}$  denote the  $w \times n$  matrix with columns  $a_i$ , then  $\|A\|_{2,\infty} = \max_{1 \leq i \leq w} \|a_i\|_2$ . Since  $\mathbb{R}^{nw} \cong \mathbb{R}^{w \times n}$ , then by the equivalence of norms there exist constants  $K_1, K_2 > 0$  such that  $K_1\|A\| \leq \|A\|_{2,\infty} \leq K_2\|A\|$ , where  $\|A\| := \|A\|_{2,2}$  denotes the Frobenius norm. It follows that

$$\int_{\Omega} \sum_{i=1}^w 0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3} dx \sim \|A\|^{2-n}.$$

Next, we observe that for every  $i \neq j$ ,  $1 \leq i, j \leq w$  we have

$$|\tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j)| \leq 1, \quad \forall a_i, a_j$$

which implies

$$\int_{\Omega} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) dx = O(1).$$

Thus, we obtain

$$\begin{aligned} & \frac{2m(\Omega)\|A\|^2}{\int_{\Omega} \sum_{i=1}^w (0.1(a_i^T x + b_i)^2 \mathbb{1}_{|a_i^T x + b_i| \leq 3} + 0.9 \mathbb{1}_{|a_i^T x + b_i| > 3}) + 2 \sum_{i \neq j} \tanh(a_i^T x + b_i) \tanh(a_j^T x + b_j) dx} \\ & \sim \frac{\|A\|^2}{\|A\|^{2-n} + O(1)} \sim \|A\|^n. \end{aligned}$$

The proof is finished after taking a square root. □

We also have a corresponding result for arbitrarily deep networks with a single neuron in each hidden layer.

PROPOSITION 4. Let  $\Omega \subset \mathbb{R}^n$  be a compact domain with positive measure,  $f_i(x) = \tanh(a_i x + b_i)$  where  $a_i \in \mathbb{R}^n$ ,  $b_i \in \mathbb{R}$ , and  $a_i, b_i \in \mathbb{R}$  for  $i \geq 2$ . Let  $g(x) = a_{d+1}x + b_{d+1}$ . Then let  $\phi = (a_1, a_2, \dots, a_d, a_{d+1}, b_1, \dots, b_d, b_{d+1})$  and define

$$f(x) = g \circ f_d \circ f_{d-1} \circ \dots \circ f_1(x).$$

Then there exists  $C := C(n, \phi, \Omega) > 0$  such that

$$\|\nabla f\|_0 \leq C\|f\|_0.$$

*Proof.* We have

$$\frac{\partial f}{\partial x_j} = a_{1j}a_2 \dots a_{d+1} \operatorname{sech}^2(a_d * f_{d-1} \circ \dots f_1(x) + b_d) \operatorname{sech}^2(a_{d-1}f_{d-1} \circ \dots f_1(x) + b_{d-1}) \dots \operatorname{sech}^2(a_1^T x + b_1).$$

Since we have  $\operatorname{sech}^2(x) \leq 1$ , then we have

$$\|\nabla f\|_0^2 \leq \|a_1\|^2 a_2^2 \dots a_d^2 a_{d+1}^2 m(\Omega).$$

On the other hand, we apply Lemma 1 recursively to obtain

$$\begin{aligned} f(x)^2 &= a_{d+1}^2 (f_d \circ f_{d-1} \circ \dots f_1(x)) + 2a_{d+1}b_{d+1}(f_d \circ f_{d-1} \circ \dots f_1(x)) + b_{d+1}^2 \\ &\geq a_{d+1}^2 g(f_d \circ f_{d-1} \circ \dots f_1(x)) + O(|a_{d+1}|) \geq a_{d+1}^2 a_d^2 (g(f_{d-1} \circ f_{d-2} \circ \dots f_1(x))) + O(|a_{d+1}a_d|) \geq \dots \\ &\dots \geq a_{d+1}^2 a_d^2 \dots a_2^2 (a_1^T x + b_1)^2 \mathbb{1}_{|a_d \tanh(\dots) + b_d| \leq 3} \dots \mathbb{1}_{|a_1^T x + b_1| \leq 3} + O(|a_{d+1}a_d \dots a_1|). \end{aligned}$$

Therefore we have

$$\|\nabla f\|_0^2 \leq \frac{\|a_1\|^2 a_2^2 \dots a_{d+1}^2 m(\Omega)}{\int_{\Omega} a_{d+1}^2 a_d^2 \dots a_2^2 (a_1^T x + b_1)^2 \mathbb{1}_{|a_d \tanh(\dots) + b_d| \leq 3} \dots \mathbb{1}_{|a_1^T x + b_1| \leq 3} + O(|a_{d+1}a_d \dots a_1|) dx} \|f\|_0^2.$$

Letting  $C = \left( \frac{\|a_1\|^2 a_2^2 \dots a_{d+1}^2 m(\Omega)}{\int_{\Omega} a_{d+1}^2 a_d^2 \dots a_2^2 (a_1^T x + b_1)^2 \mathbb{1}_{|a_d \tanh(\dots) + b_d| \leq 3} \dots \mathbb{1}_{|a_1^T x + b_1| \leq 3} + O(|a_{d+1}a_d \dots a_1|) dx} \right)^{1/2}$  yields the desired constant.  $\square$

Similarly, we derive the corresponding asymptotic:

COROLLARY 3. *We have  $\frac{\|\nabla f\|_0}{\|f\|_0} \sim \sqrt{\|a_1\|^n |a_2|^n \dots |a_d|^n}$ .*

*Proof.* So for  $i > 1$  consider

$$m(\Omega \cap \{|a_i \tanh(z) + b_i| \leq 3\}) = \int_{\Omega} \mathbb{1}_{\{|a_i \tanh(z) + b_i| \leq 3\}} dx.$$

Then the set  $\{|a_i \tanh(z) + b_i| \leq 3\}$  is equivalent to

$$|a_i \tanh(z) + b_i| \leq 3 \Leftrightarrow \left| \tanh(z) + \frac{b_i}{a_i} \right| \leq \frac{3}{|a_i|}.$$

This implies that  $m(\Omega \cap \{|a_i \tanh(z) + b_i| \leq 3\}) = O\left(\tanh^{-1}\left(\frac{1}{|a_i|}\right)^n\right)$ . Since  $\tanh^{-1}(1/|a_i|) = O(1/|a_i|)$  then it follows that

$$m(\Omega \cap \{|a_i \tanh(z) + b_i| \leq 3\}) = O\left(\frac{1}{|a_i|^n}\right).$$

Hence, we have

$$\begin{aligned} \int_{\Omega} a_{d+1}^2 a_d^2 \dots a_2^2 (a_1^T x + b_1)^2 \mathbb{1}_{|a_d \tanh(\dots) + b_d| \leq 3} \dots \mathbb{1}_{|a_1^T x + b_1| \leq 3} + O(|a_{d+1}a_d \dots a_1|) &= O\left(\frac{a_{d+1}^2}{\|a_1\|^{n-2} |a_2|^{n-2} \dots |a_d|^{n-2}}\right) \\ &= a_{d+1}^2 \|a_1\|^{2-n} |a_2|^{2-n} \dots |a_d|^{2-n} O(1). \end{aligned}$$

The result follows in a similar manner as the previous corollaries.  $\square$

REMARK 4. *We notice that as one varies the number of layers but fix to a single hidden layer, the asymptotic follows the Frobenius norm of the matrix defined by the scaling coefficient in each tanh function. In contrast, deep networks with a single neuron yield asymptotics that are products of the norms of each scaling coefficient in each tanh layer. Moreover, the presence of the exponential  $n$  shows the dependence of the network asymptotics on the spatial dimension. We conjecture that for arbitrary networks of architecture  $(d, w)$ ,*

we must have a combination of all the above. Indeed, this is the case, as we show in the next theorem.

**THEOREM 1.** *Let  $\Omega \subset \mathbb{R}^n$  be a compact domain with positive measure,  $f_i(x) = \tanh(A_i x + b_i)$  where  $A_i \in \mathbb{R}^{w \times n}$ ,  $b_i \in \mathbb{R}^w$ , and  $A_i, b_i \in \mathbb{R}^w$  for  $2 \leq i \leq d$ . (Here,  $\tanh(\cdot)$  is assumed to apply pointwise to each component.) Let  $g = a_{d+1}^T x + b_{d+1}$ ,  $a_{d+1} \in \mathbb{R}^{w \times 1}$  and  $b_{d+1} \in \mathbb{R}$ . Then let  $\phi = (A_1, A_2, \dots, a_{d+1}, b_1, \dots, b_{d+1})$  and define*

$$f(x) = g \circ f_d \circ \dots \circ f_1(x).$$

*Then there exists  $C := C(n, \phi, \Omega) > 0$  such that*

$$\|\nabla f\|_0 \leq C\|f\|_0.$$

*Proof.* We have by the chain rule

$$\frac{\partial f}{\partial x_i} = a_{d+1}^T \operatorname{sech}^2(A_d \tanh(\dots) + b_d) A_d \operatorname{sech}^2(\dots) \dots A_1 e_i$$

where  $e_i$  is the  $i$ th canonical basis vector in  $\mathbb{R}^n$ . It follows that

$$\left( \frac{\partial f}{\partial x_i} \right)^2 = (a_{d+1} \operatorname{sech}^2(\dots))^2 \|A_d \operatorname{sech}^2(\dots)\|^2 \dots \|A_1^T e_i\|^2 \leq \|a_{d+1}\|^2 \|A_d\|^2 \dots \|A_1\|^2,$$

where each matrix norm is the Frobenius norm taken with respect to the space of each matrix  $A_i$ . We therefore have

$$\|\nabla f\|_0^2 = \int_{\Omega} \sum_{i=1}^n \left( \frac{\partial f}{\partial x_i} \right)^2 dx \leq n \|a_{d+1}\|^2 \|A_d\|^2 \dots \|A_1\|^2 m(\Omega).$$

We prove the following lemma:

**LEMMA 2.** *For any neural network with architecture  $(d, w)$  with  $d \geq 1$  and  $w \geq 1$  we have*

$$f(x)^2 = O(\|a_{d+1}\|^2 \|A_d\|^{2-n} \dots \|A_1\|^{2-n}).$$

*Proof.* By modifying the proof of Proposition 3 and Corollary 2 except by multiplying by a vector  $c \in \mathbb{R}^{1 \times w}$ , then the case of  $d = 1$  is proven.

Next, we define our induction hypothesis: for any neural network  $\tilde{f}$  of architecture  $(k, w)$ , we have

$$\tilde{f}(x)^2 = O(\|a_{k+1}\|^2 \|A_k\|^{2-n} \dots \|A_1\|^{2-n}).$$

for  $k = 1, 2, \dots, N$ . Now, consider a neural network  $f(x)$  of architecture  $(N+1, w)$ . Then

$$f(x)^2 = (a_{N+2}^T f_{N+1}(\dots))^2 + O(\|a_{N+2}\|) = \sum_{j=1}^w a_{N+2,j}^2 (f_{N+1,j}(\dots))^2 + O(\|a_{N+2}\|).$$

Note that each  $a_{N+2,j} (f_{N+1,j})_j$  defines a neural network of architecture  $(N, w)$  for each  $j$ , so we apply the induction hypothesis to obtain

$$\begin{aligned} f(x)^2 &= \sum_{j=1}^w a_{N+2,j}^2 O(\|a_{(N+1),j}\|^{2-n} \dots \|A_1\|^{2-n}) \\ &= \|a_{N+2}\|_{\infty}^2 \sum_{j=1}^w \frac{a_{N+2,j}^2}{\|a_{N+2}\|_{\infty}^2} O(\|A_{N+1}\|^{2-n} \dots \|A_1\|^{2-n}). \end{aligned}$$

Using the equivalence of norms, we have

$$f(x)^2 = O(||a_{N+2}||^2 ||A_{N+1}||^{2-n} \dots ||A_1||^{2-n}).$$

By induction, we have proved the lemma.  $\square$

Thus, using the lemma we have

$$||\nabla f||_0^2 \leq \frac{n ||a_{d+1}||^2 ||A_d||^2 \dots ||A_1||^2 m(\Omega)}{\int_{\Omega} O(||a_{d+1}||^2 ||A_d||^{2-n} \dots ||A_1||^{2-n}) dx} ||f||_0^2.$$

The proof of the theorem directly follows.  $\square$

From this, we derive the asymptotic

**COROLLARY 4.** *We have  $\frac{||\nabla f||_0}{||f||_0} \sim \sqrt{||A_d||^n ||A_{d-1}||^n \dots ||A_1||^n}$ .*

We summarize the results so far for the bounding constant with respect to the dimension, number of layers, and number of neurons in Table 4. While we have computable bounds, we did not have time to verify these bounds numerically. However, in [10] we computed the Sobolev ratios during training for each iteration and found that the curve followed behaved similarly to the given asymptotics in Table 4.

(No. of Layers, No. of Neurons)	Dimension	Asymptotics of $\frac{  \nabla f  }{  f  }$
(1, 1)	1	$\sqrt{\frac{ a }{x_1 - x_0}}$
(1, 1)	$n$	$\sqrt{  a  ^n}$
(1, $w$ )	$n$	$\sqrt{  A  ^n}$
( $d$ , 1)	$n$	$\sqrt{  a_1  ^n   a_2  ^n \dots   a_d  ^n}$
( $d$ , $w$ )	$n$	$\sqrt{  A_d  ^n   A_{d-1}  ^n \dots   A_1  ^n}$

\* These results are preliminary, conservative estimates of Sobolev ratios. Tighter bounds will be the focus of future research.

TABLE 4.1

*Summary of Sobolev constants in domain  $\Omega ([x_0, x_1]$  in 1D) obtained thus far based on dimension, number of hidden layers, and number of nodes in each layer ( $n, d, w$ ). The estimated asymptotic as a function of the coefficients  $a$  is also given.*

**5. Conclusion.** We find that the asymptotics as defined in the previous corollaries and Theorem 1 match up correctly with the case when  $n = d = w = 1$ . This suggests that by using the theory of Agmon-Douglis-Nirenberg that the correct way to weigh the loss terms is to multiply each mean-squared error term with the corresponding asymptotic as suggested by Corollary 4. It also suggests that for an arbitrary neural network architecture  $(d, w)$  that

$$\|u_w^d\|_{L^{1/2}(\Omega)} \leq (\sqrt{||A_d||^n ||A_{d-1}||^n \dots ||A_1||^n})^3 \|u_w^d\|_{L^2(\Omega)}.$$

In other words, the proper weight to adjust for the trace norm is to raise the asymptotic by an exponent of 3.

In this manuscript, we presented both novel empirical and theoretical results that estimate the ratio of the semi-norm of a neural net to its  $L^2$  norm. Other aspects to consider

would be to analyze the norms of the curl and divergence operators acting on a neural network in a similar way as the analysis carried out for neural network functions with the Sobolev ratios. In this case, the ratios under consideration would involve finding inverse inequalities that bound functions in  $\text{Curl}(\Omega)$  by functions  $G(\Omega)$  and  $\text{Div}(\Omega)$  by functions in  $\text{Curl}(\Omega)$ , where  $G$ ,  $\text{Curl}$ , and  $\text{Div}$  represent the gradient, curl and divergence spaces and  $\Omega \subset \mathbb{R}^n$ .

## REFERENCES

- [1] S. AGMON, A. DOUGLIS, AND L. NIRENBERG, Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions II, Comm. Pure Appl. Math., 17 (1964), pp. 35–92.
- [2] A. K. AZIZ, R. B. KELLOGG, AND A. B. STEPHENS, Least squares methods for elliptic systems, Mathematics of Computation, 44 (1985), pp. pp. 53–70.
- [3] P. B. BOCHEV AND M. D. GUNZBURGER, Least-squares finite element methods, vol. 166, Springer Science & Business Media, 2009.
- [4] J. H. BRAMBLE, R. D. LAZAROV, AND J. E. PASCIAK, A least-squares approach based on a discrete minus one inner product for first order systems, Mathematics of Computation, 66 (1997), pp. pp. 935–955.
- [5] T. DE RYCK, S. LANTHALER, AND S. MISHRA, On the approximation of functions by tanh neural networks, Neural Networks, 143 (2021), pp. 732–750.
- [6] M. W. M. G. DISSANAYAKE AND N. PHAN-THIEN, Neural-network-based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.
- [7] I. GÜHRING, G. KUTYNIOK, AND P. PETERSEN, Error bounds for approximations with deep relu neural networks in  $w$  s,  $p$  norms, Analysis and Applications, 18 (2020), pp. 803–859.
- [8] I. GÜHRING AND M. RASLAN, Approximation rates for neural networks with encodable weights in smoothness spaces, Neural Networks, 134 (2021), pp. 107–130.
- [9] I. LAGARIS, A. LIKAS, AND D. FOTIADIS, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9 (1998), pp. 987–1000.
- [10] T. MEISSNER, E. HUYNH, P. KUBERRY, AND P. BOCHEV, Pinns for stokes: An elliptic regularity approach, CSRI 2023 Proceedings, (2023).
- [11] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019), pp. 686–707.
- [12] G. STARKE, Multilevel boundary functionals for least-squares mixed finite element methods, SIAM Journal on Numerical Analysis, 36 (1999), pp. 1065–1077.

## UNCERTAINTY QUANTIFICATION OF MODEL FORM ERROR THROUGH SURROGATE MODELS

EMILIANO ISLAS QUINONES\* AND KATHRYN A. MAUPIN†

### **Abstract.**

Uncertainty quantification relies on a set of methods for gathering and analyzing model information and attempts to classify it. With the complexity of high-fidelity models, surrogate models can be used as an approximation of the high-fidelity models to simplify their behavior, with their own function of the data. Validation, verification, and sensitivity analysis methods can be used to quantify function inaccuracy through the use of vector norms or other methods to compare behavior between models. Measuring the change in surrogate model prediction between runs and models tends to highlight areas of the original model where its most uncertain. Using this error as a measure to correct its origin is one of the principal goals of model form error research.

### **1. Introduction.**

From the view of information theory, data and the information therein is often abstract and relative. Scientists interpret information with their own perspectives and explanations. Similarly, each mathematical function interprets given data to produce a prediction. The existence of these differences in perspective is a driving force behind Uncertainty Quantification (UQ), as its goal is to measure the uncertainty in data, model, and predictions. Methods for quantifying uncertainty often rely on surrogate models to interpret data and create subsequent function approximations of it for further analysis. Surrogate models can be used to interpolate existing data sets, e.g. to create more data for later use, to explore more regions in-between data that was not analyzed before, or to potentially extrapolate, to see behaviour outside the known data ranges, although trusting this extrapolated data depends on the complexity of the data the surrogate model was trained on [17]. Yet another use is to understand where and how another model, such as a high-fidelity physics-based model, deviates from experimental data [7, 8]. This can be analyzed through the use of conventional metrics that are capable of measuring distances and even the difference in function trend and distribution [9, 11].

With the driving goal of uncertainty quantification, this paper demonstrates the utility of surrogate models in capturing model form error. To this end, Section 2 provides background theory of model form error. Details regarding the types of surrogates considered in this work are given in Section 3. Section 4 contains the mathematical formulation of two validation metrics. A demonstration of the interplay of these ideas is given in Section 5 before we conclude in Section 6.

### **2. Model Form Error.**

The primary goal of computational model calibration is to find model parameters that accurately capture experimental (or high-fidelity model) data. Mathematically, this is written as

$$\theta^* = \operatorname{argmin} \|d(x) - m(\theta, x)\|, \quad (2.1)$$

where  $d$  is the experimental data and  $m$  is the model, which depends on parameters  $\theta$  and state or configuration variables  $x$ . This minimization problem can then be solved with a variety of optimization routines [12]. In the case of Bayesian calibration, Equation 2.1 is incorporated into the likelihood function. This topic is beyond the scope of the current work, but further details can be found in [4].

---

\*Sandia National Laboratories, eislasq@sandia.gov

†Sandia National Laboratories, kmaupin@sandia.gov

Realistically, a computational model will not exactly reproduce experimental data, as physical data is corrupted by experimental noise. Equation 2.1 can therefore be rewritten as,

$$d(x) = m(\theta, x) + \epsilon, \quad (2.2)$$

with  $\epsilon$  representing the experimental or measurement noise. Despite this consideration, in some cases, the model predictions yielded by the optimal model parameters do not adequately capture the experimental data. Discrepancy between a model's prediction of an experiment and the actual data, can be attributed to model form error, which may also be called model discrepancy, model inadequacy, or structural error. An additional model error term  $\delta$  can then be added to Equation 2.2,

$$d(x) = m(x, \theta) + \delta(x) + \epsilon. \quad (2.3)$$

Methods to quantify the possible causes of discrepancy often rely on the use of surrogate models, from which new parameters can be tested onto the computational model's surrogate, and through subsequent analyses look for the potential source of the discrepancy between the high-fidelity model and the experimental data. It is often found that the model is missing physics or has incorrect structure or formulation of the underlying equations. Through the use of a surrogate model, we may be able to pinpoint which modeling choices, such as the spatial location, parameter value, or model structure, are causing the model and the experimental data to differ. Different assumptions lead to different results, all to fix the potential errors. The more perspectives we have on the data and the model, the more we can understand what their respective output distributions could depend on.

### 3. Surrogate Models.

The shape of data can be described by a function  $f(x)$ , which turns given inputs into a set of outputs  $x \rightarrow f(x) \rightarrow y$ . However, that function is not always known. The primary goal of function approximators is to find how independent inputs  $x$  lead to their dependent variables  $y$ . Surrogate models themselves attempt to substitute the data generated by an unknown model with a new physics-agnostic function approximation. The intent of the new function is flexible and depends on what is needed, be it creation of more data or a deeper understanding of it.

Surrogate models help analysts understand the distribution and potential location of new data through interpolation and extrapolation (see Figure 3.1). Interpolation adds new

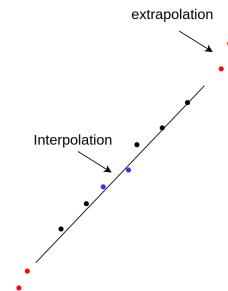


Fig. 3.1: Difference between function interpolation points and extrapolation points

data within the bounds of the trained parameters of the approximated function, while

extrapolation includes it outside those bounds, making it a harder method to track and analyze.

Although more complex machine learning techniques which take into account knowledge of the input variables or underlying physics exist [13, 5], we consider here simpler machine learning models that require knowledge only of the response data. After training, even these simple models are capable of creating predictions with new tweaked parameters, this exploration should be tackled with care however, as the lack of physics in the function approximation could lead to incorrect estimation of the parameters.

The use of multiple surrogate models could also aid in understanding the behavior of the data, as each surrogate model interprets inputs differently. Analyses such as sensitivity analysis [16], model selection [2], and model validation [10] illuminate such differences. This difference in interpretation could give an extra layer of verification amongst the surrogate models, as similar sensitivity in parameters would mean they are training akin to each other.

### 3.1. Dakota.

Dakota is a toolkit for optimization and uncertainty quantification that can perform parametric analyses which can enable design exploration, model calibration, risk analysis, and quantification of margins and uncertainty with computational models [1]. Its methods include optimization with gradient- and nongradient-based methods; uncertainty quantification with sampling, reliability, stochastic expansion, and epistemic methods; and sensitivity/variance analysis with design of experiments and parameter study methods. Dakota also offers several surrogate modeling options, two of which are featured in this work, as detailed in the following sections.

#### 3.1.1. Gaussian Process/Kriging.

Gaussian Process (GP) models are interpolatory models that produce surface fits of response values or quantities of interest, given a set of data points, using ideas from geostatistics and spatial statistics [1, 20]. A GP takes the form

$$\hat{f}(x) \approx g(x)^T \beta + r(x)^T R^{-1}(f - G\beta), \quad (3.1)$$

where  $x$  is the current point in  $n$ -dimensional parameter space;  $g(x)$  is the vector of trend basis functions evaluated at  $x$ ;  $\beta$  is a vector containing the generalized least squares estimates of the trend basis function coefficients;  $r(x)$  is the correlation vector of terms between  $x$  and the data points;  $R$  is the correlation matrix for all of the data points;  $f$  is the vector of response values; and  $G$  is the matrix containing the trend basis functions evaluated at all data points. Although there are numerous mathematically sound options for the correlation function [20], the Dakota implementation uses a Gaussian correlation function in each entry of the correlation vector and matrix. The hyperparameters are calibrated using a maximum likelihood estimation [1]. GP models are quite accurate interpolants, as they are capable of having no uncertainty at the data points it was built from, meaning it can reproduce the outputs made by the original inputs [1]. This allows for a more accurate analysis of the surrogate model at data points within the parameter ranges it was built on. The main drawbacks of this advantage is that GP models struggle making predictions outside the trained parameter range.

#### 3.1.2. Stochastic Layered Perceptron Network.

The Artificial Neural Network (ANN) implementation in Dakota uses a Stochastic Layered Perceptron (SLP) ANN based on a direct training approach. This approach has a lower training cost compared to more complex Dakota surrogate models, making it useful for optimization schemes in which ANNs need to be constructed repeatedly [1]. The form

of the SLP ANN is:

$$\hat{f}(x) \approx \tanh(\tanh((xA_0 + \theta_0)A_1 + \theta_1)) \quad (3.2)$$

where  $x$  is the current point in the n-dimensional parameter space, and  $A_0, \theta_0, A_1, \theta_1$  are the matrices and vectors that correspond to the neuron weights and offset values in the model. These terms are computed during the training phase of the model and are analogous to the polynomial coefficients in a quadratic surface fit. In the Dakota implementation, the weights and offsets are estimated using a singular value decomposition [1]. Unlike the GP, the SLP ANN is not guaranteed to always match the response values of the data points.

### 3.2. Tensorflow/Keras.

In general, ANNs are useful a variety of problems, including classification and single- and multi-output regression [3]. Keras is a high-level API for Tensorflow that provides an implementation of artificial neural networks that can be extensively modified to suit the needs of the project at hand. Similarly to the Dakota ANN, the Keras implementation is able to closely match the response values of the data it was built from, with some level of uncertainty. This is mainly due to the non-linearity of the networks and the data so extensive network tuning has to be done if that accuracy is needed. The non-linear nature of ANN allows it to approximate functions of any dimension, making for a great flexible tool. In the present work, the modifications that were used are detailed in the following sections.

#### 3.2.1. Activation Functions.

Activation functions are vital for artificial neural networks, as they prevent nodes from creating linear functions once their outputs are combined. One of the most popular functions is sigmoid, which returns values between 0 and 1, indicating the probability of a binary decision. However, this poses a limitation for regression problems. The Rectified Linear Unit (ReLU) function is better for regression, as it can keep the identity of the input. ReLU returns the max between an input value  $x$  and 0,  $\max(0, x)$ . However, ReLU can fail when inputs are too small, driving to the problem called “dying relu” where a node is never activated again due to the zero-to-identity mapping [6]. To address this, similar functions, such as leaky ReLU, ELU, SeLU, and GeLU aim to allow some negative values to pass through and activate a node. The GeLU function has the form:

$$GeLU(x) = xP(X \leq x) = x\Phi(x) = x\frac{1}{2}[1 + erf(\frac{x}{\sqrt{2}})]. \quad (3.3)$$

GeLU has been of particular interest, as it attempts to keep ReLU’s zero-to-identity mapping, while scaling input  $x$  to how much greater it is than other inputs, which is equivalent to multiplying  $x$  by the standard Gaussian cumulative distribution function of the input  $\Phi(x)$ . In turn,  $\Phi(x)$  can be calculated with the Gaussian error function, yielding the third equality. Due to computational costs,  $\Phi(x)$  can also be approximated with  $\tanh$  as  $0.5 * x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)])$ . This approximation increases computation speed at a slight accuracy decrease [6].

#### 3.2.2. Loss measures.

There exist metrics that are capable of obtaining an error score between each run of the ANN. These inform the network about its accuracy and later scale parameters appropriately during backpropagation, as discussed in the next section. Loss measures may be as simple as Root Mean Square Error, Mean Absolute Error, or  $\ell_p$  norms [18]. Other metrics scale in complexity when observing the distribution of the data needed, as opposed to point-to-point distance. The Kullback-Liebler Divergence is one of these metrics as it focuses on how much

the distribution of the ANN model prediction output changed from the target data [18]. There has also been some recent work on the use of the Jensen-Shannon Divergence as a loss function [19].

### 3.2.3. Optimizers.

Backpropagation is what allows ANNs to “learn,” and gradient descent is capable of discerning whether or not a network is learning and by how much. Loss or objective function  $J(\theta)$  is minimized through the update of the model parameters  $\theta$  in the opposite direction of the gradient of the objective function. The standard formulation being:  $\theta = \theta - \eta \times \nabla_{\theta} J(\theta)$ , where  $\eta$  represents the size of the steps per iteration (learning rate) [15].

## 4. Validation Metrics.

Computational models, including surrogate models, are a vital analytical tool once they are determined to be accurate for predictions of interest. Formal model validation analyses use metrics and a pre-determined tolerance to evaluate the proximity of the model to the target data. Among the most commonly used metrics are variations of  $\ell_p$  vector norms, which directly compare model predictions with experimental data without acknowledgement of their uncertainty. Other, more complex metrics based in statistics or information theory take into consideration the distribution (uncertainty) of the model predictions and/or the actual data. These metrics can also be used as a loss function, as mentioned before, to inform how much a machine learning model has improved over time. This work considers and compares the utility of two validation metrics.

### 4.1. Root Mean Square Error.

A commonly used validation metric is the root mean square error (RMSE), which is a variant of the Euclidean distance. Given predictions  $\hat{y}_i$  and observational data  $y_i$ ,

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}, \quad (4.1)$$

where  $n$  is the number of observations. This is a fast and simple metric that can return a score for point-to-point distance error, which if comparing vectors, can be averaged to get a scalar score. Figure 4.1 depicts the calculated distance between an expected interpolation

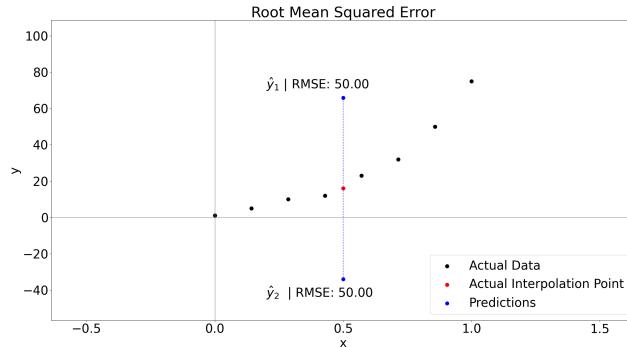


Fig. 4.1: RMSE error distance between Expected/Actual interpolation point and failed attempted prediction points  $\hat{y}_1$  and  $\hat{y}_2$  made by a surrogate trained on the actual data.

point, and two incorrect attempts of a model's prediction of it. A good root mean squared error score would have a score closer to zero.

#### 4.2. Mahalanobis Distance.

Deterministic metrics, such as RMSE can inform the point-to-point distance between the model and the data it is attempting to approximate. However, those metrics fail to compare the covariance difference between the predicted point and the expected data, as shown in Figure 4.1. The Mahalanobis Distance (MD) can be used to create an error ellipse that describes the covariance of the observational data. Each new point is compared against ellipse, yielding a score that describes how well the point followed the covariance of the data [9].

The standard MD formula for an experimental dataset  $D$ , with a covariance matrix  $\Sigma_D$ , and model data  $P$  is,

$$MD = \sqrt{(P - D)^T \Sigma_D^{-1} (P - D)}. \quad (4.2)$$

This equation can be modified to produce a vector of MD values containing entries for each individual prediction point, giving a better perspective to single out what data points are affecting distribution of the correct covariance the most. The vector for each new prediction point  $\hat{y}_i$  in the  $i^{th}$  entry in the MD vector  $\overrightarrow{MD}$  is given by,

$$\overrightarrow{MD}_i = \sqrt{\left( \begin{pmatrix} x_i - \bar{x} \\ \hat{y}_i - \bar{y} \end{pmatrix}^T \Sigma_{xy}^{-1} \begin{pmatrix} x_i - \bar{x} \\ \hat{y}_i - \bar{y} \end{pmatrix} \right)}, \quad (4.3)$$

where  $y$  is the observational data with parameters  $x$ ,  $\Sigma_{xy}^{-1}$  being their inverse covariance matrix, and lastly  $\bar{y}$  and  $\bar{x}$  as their respective means.

Figure 4.2 demonstrates a situation where the RMSE yields the same score for two new prediction points  $\hat{y}_1$  and  $\hat{y}_2$ . Although their distance to the actual data point is the same,  $\hat{y}_1$  matches the covariance of the function closer than  $\hat{y}_2$ , as shown by the ellipse. This can

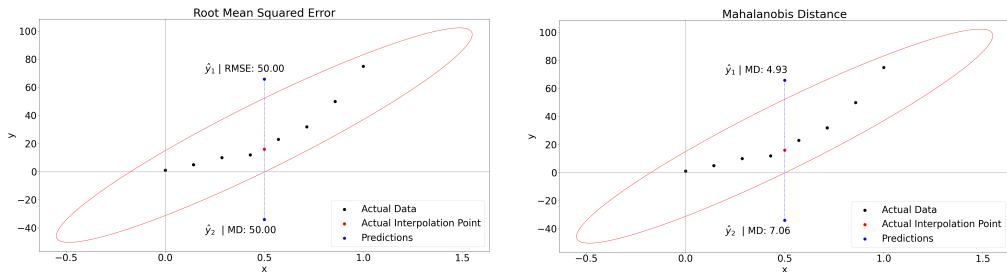


Fig. 4.2: RMSE and Mahalanobis difference in error distance between prediction points  $\hat{y}_1$  and  $\hat{y}_2$  to the covariance ellipse of the actual data

be addressed with the mean of the previous formula as a score,

$$\overrightarrow{MD} = \frac{\sum_i^n \sqrt{\left( \begin{pmatrix} x_i - \bar{x} \\ \hat{y}_i - \bar{y} \end{pmatrix}^T \Sigma_{xy}^{-1} \begin{pmatrix} x_i - \bar{x} \\ \hat{y}_i - \bar{y} \end{pmatrix} \right)}}{n} \quad (4.4)$$

It may also be useful to further extend this formulation to include a direct comparison of

the covariance of the prediction to the covariance of the data, as

$$MD = \frac{\sum_i^n \sqrt{(\hat{x}_i - \bar{x})^T \Sigma_{xy}^{-1} (\hat{x}_i - \bar{x})}}{n} + \sqrt{\sum_{ij} |\Sigma_{x\hat{y}_{ij}} - \Sigma_{xy_{ij}}|}. \quad (4.5)$$

Here, the second term is the distance between the covariance matrix of the predictions  $\Sigma_{x\hat{y}}$  and the covariance matrix of the experimental data  $\Sigma_{xy}$ .

The modified MD in Equation 4.5 contains information about the point-to-distribution distance, as well as the distance between two distributions, all in a single score. By simultaneously highlighting specific areas of the distribution where the model is inaccurate and directly comparing the covariance matrices, we may be able to obtain additional information about how a model is failing. The closer the score is to 0, the closer the distributions are to equivalence. Generally a good score lies between 0-10, indicating a distribution impartiality.

### 5. Example.

As an example, we consider data from a laminar hypersonic double-cone experiment (LENS-I). Experimental data is available, produced at Calspan University at Buffalo Research Center's Large Energy National Shock Tunnel. High-fidelity simulation data is also available from Sandia's Parallel Aerodynamic Reentry Code (SPARC). Extensive details regarding the experiment and high-fidelity simulations can be found in [14]. In this experimental scenario, the experimental conditions were varied and a single test was performed for each set of conditions. We have at our disposal three of these data sets. Given the limited data, our goal is to compare the behavior and performance of the surrogate models detailed in Section 3 in interpolation scenarios, as well as in extrapolation.

The available data is parameterized by the freestream conditions: density  $\rho$ , velocity  $U$ , and temperature  $T$ , and well as the spatial variable  $x$ . The two dependent variables, our quantities of interest, are pressure and heat flux. Figure 5.1 shows the distributions of each

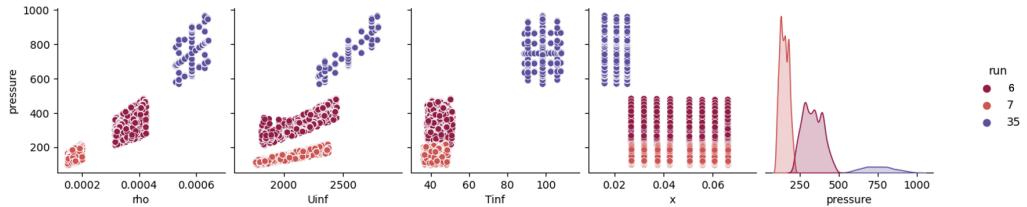


Fig. 5.1: Physics-based parameters against dependent variable pressure with the respective runs produced by the high-fidelity model SPARC

physics-based parameter against pressure, the dependent variable of interest. This sample data is produced by SPARC in an attempt to find the parameter combination that outputs the pressure at a specific sensor location  $x$  detected by LENS-I.

Taking a closer look at parameter  $x$ , it can be noted that 8 sensor locations from Run 6 are analyzed. For each sensor, there are 300 different parameter combinations leading to the variation of pressures per sensor. The following figure shows the closest combination to LENS-I output by SPARC.

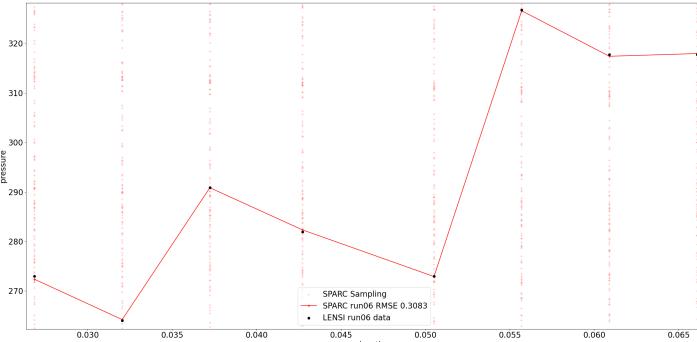


Fig. 5.2: SPARC parameter sample combinations per sensor "x" with closest vector to LENS-I data

In the analysis that follows, only single output models will be used for the function approximation. The surrogates discussed in Section 3 are constructed from the parameters and outputs from the shown Run 6 data of the high-fidelity model SPARC using the validation metrics detailed in Section 4 to determine the accuracy of the surrogates. Leading to capable surrogates of replicating the vector shown in Figure 5.2.

### 5.1. Interpolation Tests.

An advantage of having a function approximation of a high-fidelity model is the ability to input new parameters within the bounds of what it was trained on while getting a response to the new data rather quickly. The surrogate models are capable of acting as a new run of the SPARC model with different parameters without the need to do heavy calculations. We perform two types of interpolation tests of our surrogate models, to validate whether or not the model is an accurate representation of SPARC. The first is a cross-validation test, in which some data is selected for training, and the rest is used for testing the newly calibrated models. We chose to split the data such that 70% of the data points were used for training, and the remaining 30% were used for testing. Results from the cross-validation test are shown in Figure 5.3.

The same data was used to train (calibrate) a Dakota Gaussian Process (DGP), a Dakota stochastic layered perceptron (DSLP), and a Keras model, earlier described in Section 3. For all models, their function approximation was highly accurate, not only in distance to each individual data point but also in terms of shape of the data set as a whole.

For this example, it can be noted that DGP had the best fit as both its MD and MSE scores are the lowest of the models, its only limitation being the longest calculation runtime, which are later discussed and compared. The DSLP achieved a close result to the DGP with a faster training time. However, it required a larger number of nodes to achieve this accuracy. The Keras model also fit the data well, with a similar execution time as the DSLP. The lack of precision in this model could be due to the need of hyper-parameter tuning. A summary of the calculated validation metrics and training runtime is given in Table 5.1. This first test case showcases model performance, and other methods of interpolation can be used to test the limits of the new models.

For the second test case, data was chosen such that there is a gap between two sections of training data, making interpolation assumptions harder to achieve. One group of data contained pressures less than 150, and the other contained pressures above 500. Not sur-

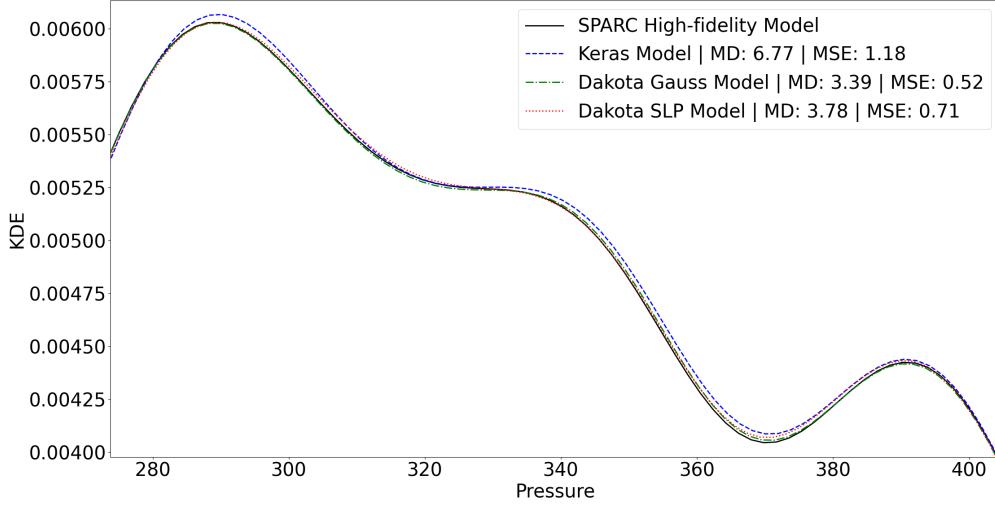


Fig. 5.3: Cross-Validation test between multiple surrogate models built on SPARC data with respective performance metrics

prisingly, in Figure 5.4 the largest deviation from the simulation model occurs for pressures between 150 to 500, which is where the missing data was split, and the models attempted to interpolate. The DGP does a great job fitting training data but struggles where data is lacking, and it attempts to make more evenly distributed predictions, as observed by the bell-like shapes of the function.

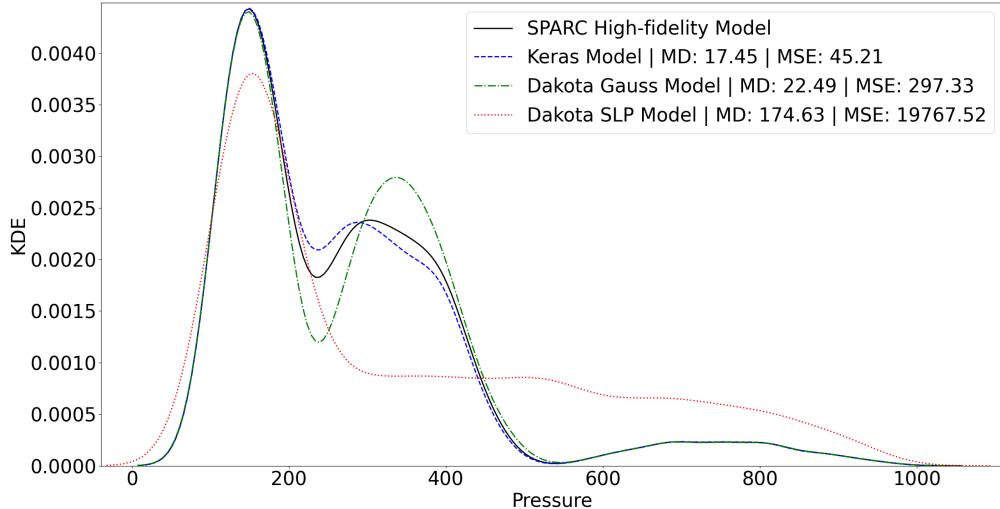


Fig. 5.4: Function approximations of models trained on data with a great gap in data between two different pressure ranges 0-300 and 600-1000

The DSLP also struggles with predictions between the training data sets. This could be due to overfitting, as this model creates a great amount of nodes for each sample of the data. The Keras artificial neural network, however, has more accurate function approximates. The validation metric scores and training time are summarized in Table 5.1. Note that although the DGP does reasonably well, the Keras model follows the distribution of the function to a greater extent.

### 5.2. Extrapolation Test.

Model extrapolation is notoriously risky, and requires careful considerations in the training and validation stages. To emulate extrapolation, data at one of the bounds of the output range is withheld during model training. As demonstrated in Figure 5.5 although the models have almost no trouble fitting the function in the range where data is available (pressures below 500), they are less accurate when making predictions, especially if the model tends to overfit.

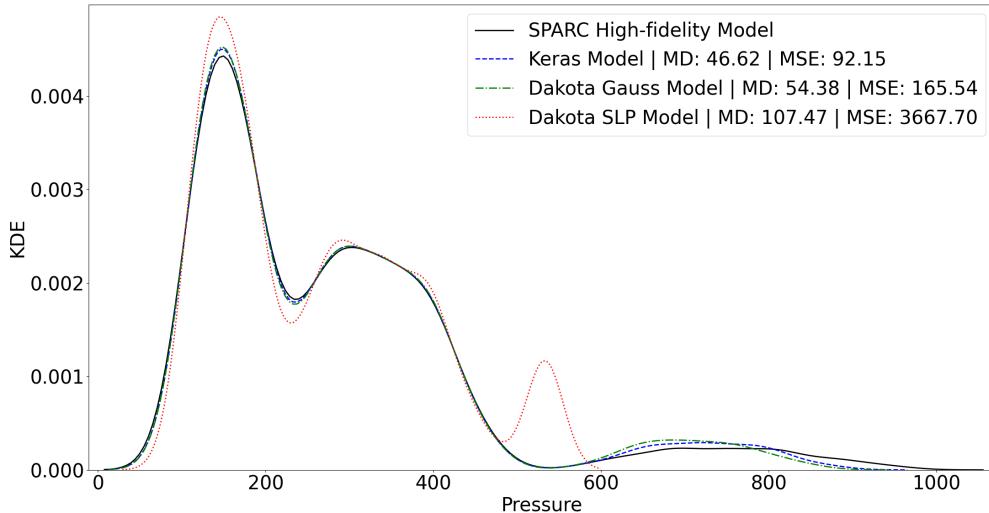


Fig. 5.5: Function approximations of models trained on data with a great gap in data between pressure ranges 600-1000, to emulate extrapolation

For the extrapolation test, the density of the data that the model was not trained on is much less as shown by the low density in Figure 5.5 along the Pressure ranges of (550 to 1000). Thus, more sampling data is needed to fit the overall function better, a possible solution would be to create more sampling data from either the DGP model or the Keras model and use it to inform the DSLP model. This would allow the DSLP model to approximate the functions of the other models.

### 5.3. Runtime and Results.

Model performance can be measured according to validation metrics, as previously discussed. Additionally, the runtime between multiple runs can describe how the time complexity scales, depending on the variation of the inputs. An average error score determines, how well a model is able to capture the high-fidelity data of the SPARC model. In the case of Mahalanobis Distance, information can also be gleaned as to how close the surrogate model is to approaching the trend of the data distribution.

Table 5.1: Surrogate model run data depicting difference between runtime and accuracy of models with each different test

Model w/1680 samples	Runtime	MSE	MD+
Dakota GP	02 <sub>m</sub> : 37 <sub>s</sub> : 47 <sub>ms</sub>	0.52	3.39
Dakota SLP	01 <sub>m</sub> : 15 <sub>s</sub> : 49 <sub>ms</sub>	0.7	3.79
Keras DNN	01 <sub>m</sub> : 35 <sub>s</sub> : 20 <sub>ms</sub>	1.18	6.77
Model w/2733 samples	Runtime	MSE	MD+
Dakota GP	14 <sub>m</sub> : 58 <sub>s</sub> : 04 <sub>ms</sub>	297.33	22.49
Dakota SLP	00 <sub>m</sub> : 51 <sub>s</sub> : 06 <sub>ms</sub>	19767.52	174.63
Keras DNN	02 <sub>m</sub> : 43 <sub>s</sub> : 39 <sub>ms</sub>	45.21	17.45
Model w/4800 samples	Runtime	MSE	MD+
Dakota GP	32 <sub>m</sub> : 56 <sub>s</sub> : 23 <sub>ms</sub>	165.54	54.38
Dakota SLP	10 <sub>m</sub> : 44 <sub>s</sub> : 13 <sub>ms</sub>	3667.70	107.47
Keras DNN	04 <sub>m</sub> : 7 <sub>s</sub> : 32 <sub>ms</sub>	92.15	46.62

The difference between the runtime of the surrogate models increases as the number of data samples increases, as shown in Table 5.1. The Dakota models tend to fit the data accurately with interpolation runs if the data is evenly distributed; however, when the data is too sparsely distributed, the models tend to overfit. The Keras DNN is the most consistent, as it is able to replicate the function somewhat closely regardless of the data distribution. This complexity of the DNN model underperforms while doing simpler tasks like more standard interpolation as shown in Figure 5.3. It is expected for the runtime of the neural networks to scale consistently with the number of samples. As previously discussed, GPs suffer from the curse of dimensionality if large amounts of data are available. This is further supported by the summary in Table 5.1.

As the surrogate models are capable of behaving closely to the original high-fidelity model, the input parameter space of each surrogate can be tweaked or scaled further to analyze the independent parameter effect on the overall output. Reflected in the following figures, changing individual parameters can have positive effects that can lead to a closer approximation to the data SPARC is attempting to match.

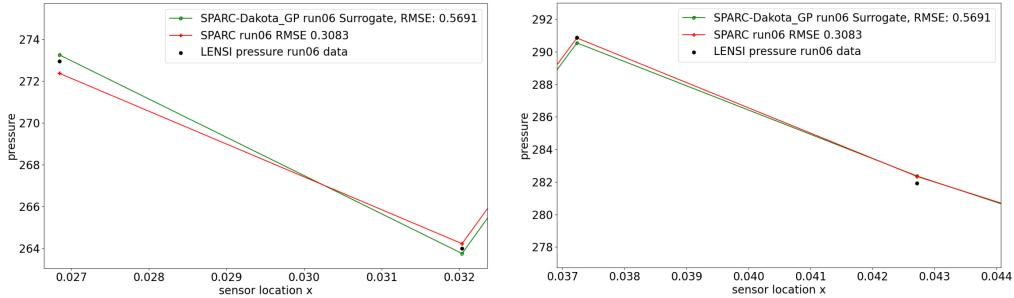


Fig. 5.6: Parameter scaling of an individual surrogate depicting more accurate local results to LENSI data

Through parameter calibration methods, it is possible to obtain the best parameter combinations that lead to the closest output to the actual data. The multiple combinations can be further analyzed to see how local calibrated parameters change from the original high-fidelity model parameters and look for a potential trend in the global parameter space. Coupling the calibrated parameters into the high-fidelity model is a different experiment overall, but if possible this could help validate if the calibrated parameter output of the surrogate follows the unknown expected result of the simulation model, reducing the number of samples the high-fidelity model would have to run.

## 6. Conclusion.

The presented work addressed the computational exercise of creating a function out of a real experiment or a corresponding high-fidelity simulation. Surrogate model implementations interpolate the available data using various mathematical and statistical methods. One principal goal is to gather as much information as possible from this new interpretation, from both extrapolation and interpolation, with the hopes of producing additional data without the use of the original model.

Validation metrics provide a quantitative method of evaluating a model's discrepancy from an expected result. Appropriate metrics are also able to capture key characteristics of a model. With more information about a particular function's trend change, a deeper understanding of how a function attributes to the distribution of the data can be gathered. Furthermore, with the analysis of the behavior of multiple surrogate model runs, it can be observed where discrepancy most commonly lies amongst the runs. From this, a particular variable or parameter can be tied as the cause of the inferred discrepancy. All with the goal of informing the model of what could be creating the discrepancy, or creating new data that describes the effects of inputs on the output.

## REFERENCES

- [1] B. ADAMS, W. BOHNHOFF, K. DALBEY, M. EBEIDA, J. EDDY, M. ELDRED, R. HOOPER, P. HOUGH, K. HU, J. JAKEMAN, M. KHALIL, K. MAUPIN, J. MONSCHKE, L. SWILER, AND J. WINOKUR, *Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis*, Tech. Rep. SAND2020-12495, Sandia National Lab.(SNL-NM), Albuquerque, NM, 2021.
- [2] D. ANDERSON AND K. BURNHAM, *Model selection and multi-model inference*, Second. NY: Springer-Verlag, 63 (2004), p. 10.
- [3] F. CHOLLET ET AL., *Keras*. <https://keras.io>, 2015.
- [4] K. FARRELL, J. T. ODEN, AND D. FAGHIHI, *A Bayesian framework for adaptive selection, calibration, and validation of coarse-grained models of atomistic systems*, Journal of Computational Physics, 295 (2015), pp. 189–208.
- [5] A. L. FRANKEL, R. E. JONES, AND L. P. SWILER, *Tensor basis gaussian process models of hyperelastic materials*, Journal of Machine Learning for Modeling and Computing, 1 (2020).
- [6] D. HENDRYCKS AND K. GIMPEL, *Gaussian error linear units (gelus)*, 2023.
- [7] M. C. KENNEDY AND A. O'HAGAN, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63 (2001), pp. 425–464.
- [8] K. A. MAUPIN, T. PORTONE, J. RAY, AND A. TRAN, *Informing missing physics with model form error and model selection*, Tech. Rep. SAND2022-11009C, Sandia National Lab.(SNL-NM), Albuquerque, NM, 2022.
- [9] 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), p. 031002.
- [10] W. L. OBERKAMPF AND C. J. ROY, *Verification and validation in scientific computing*, Cambridge University Press, 2010.
- [11] N. W. PORTER, K. A. MAUPIN, L. P. SWILER, AND V. A. MOUSSEAU, *Validation metrics for fixed effects and mixed-effects calibration*, Journal of Verification, Validation and Uncertainty Quantification, 6 (2021), p. 011005.

- [12] A. QUATERONI, R. SACCO, AND F. SALERI, *Numerical mathematics*, vol. 37, Springer Science & Business Media, 2010.
- [13] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707.
- [14] J. RAY, S. KIEWEG, D. DINZL, B. CARNES, V. G. WEIRS, B. FRENO, M. HOWARD, T. SMITH, I. NOMPÉLIS, AND G. V. CANDLER, *Estimation of inflow uncertainties in laminar hypersonic double-cone experiments*, AIAA journal, 58 (2020), pp. 4461–4474.
- [15] S. RUDER, *An overview of gradient descent optimization algorithms*, CoRR, abs/1609.04747 (2017).
- [16] A. SALTELLI, M. RATTO, T. ANDRES, F. CAMPOLONGO, J. CARIBONI, D. GATELLI, M. SAISANA, AND S. TARANTOLA, *Global sensitivity analysis: the primer*, John Wiley & Sons, 2008.
- [17] R. C. SMITH, *Uncertainty quantification: theory, implementation, and applications*, vol. 12, SIAM, 2013.
- [18] J. TERVEN, D. M. CORDOVA-ESPARZA, A. RAMIREZ-PEDRAZA, AND E. A. CHAVEZ-URBIOLA, *Loss functions and metrics in deep learning. a review*, 2023.
- [19] P. THIAGARAJAN AND S. GHOSH, *A Jensen-shannon divergence based loss function for bayesian neural networks*, (2022).
- [20] C. K. WILLIAMS AND C. E. RASMUSSEN, *Gaussian processes for machine learning*, vol. 2, MIT press Cambridge, MA, 2006.

## A NEURAL ODE BASED FLUX SURROGATE ALGORITHM FOR COUPLED TRANSMISSION PROBLEMS

RISHI PAWAR\*, PAVEL BOCHEV†, AND JUSTIN OWEN‡

**Abstract.** In this paper we develop and demonstrate a new surrogate-based partitioned scheme for a model advection-diffusion transmission problem. The work is motivated by the Implicit Value Recovery (IVR) partitioned method [5] in which the interface flux is approximated by the dual Schur complement of a discrete monolithic formulation of the coupled problem. Unless the discretized equations employ lumped mass matrices, forming and solving the Schur complement equation for the flux adds nontrivial computational cost at every time step. To reduce the computational cost we replace the Schur complement equation by an efficient flux surrogate. The latter is based on a Neural ODE (NODE) which learns the dynamic behavior of the interface flux from a suitable training set during an offline training phase. The NODE is then used during the online stage to provide accurate flux approximations to each subdomain problem. Numerical examples illustrate the potential of the approach.

**1. Introduction.** Explicit partition methods for coupled problems enable independent solution of subdomain equations by codes that can be optimized for each specific physics model. Such schemes improve computational efficiency and enable reuse of existing simulation codes.

Loosely coupled partitioned schemes exchange interface states and/or fluxes to define suitable boundary conditions for the subdomain equations, thereby enabling their independent solution. Such schemes are minimally intrusive and computationally efficient. However, loosely coupled schemes are equivalent to a single step of an iterative solution procedure for the underlying coupled system and, as a result, they can experience instabilities and loss of accuracy. An alternative approach is to develop a partitioned scheme starting from a monolithic formulation of the coupled problem in which the continuity of the states is enforced by a Lagrange multiplier. The interface flux is then estimated by solving a dual Schur complement system. This approach leads to provably stable and accurate solutions [5] and is second order accurate provided the monolithic problem employs consistent mass matrices. However, in this case forming the Schur complement involves inversion of the subdomain mass matrices and leads to a small but dense linear system. As a result, such an approach can incur significant computational costs, especially in three dimensions.

The main goal of this work is to demonstrate that a data-driven surrogate flux approach can potentially combine the computational efficiency of loosely coupled schemes with the accuracy of methods based on monolithic formulations. Specifically, we replace the Schur complement step of the IVR algorithm by a flux surrogate based on Neural ODEs (NODE). We choose NODEs because they provide an effective mechanism to learn the dynamics of the interface flux from time series data. In so doing we shift the main computational burden to an offline training phase where the NODE flux surrogate is inferred from a suitable training set. During the online phase, the trained NODE surrogate is used to compute interface fluxes at a fraction of the cost of solving the Schur complement.

We have organized this paper as follows. In Section 2 we review the Implicit Value Recovery Algorithm, which provides the basis for the NODE surrogate-based partitioned scheme. Section 3 summarizes the basics of NODEs and formulates the NODE flux surrogate. This section also describes the modified IVR scheme in which the Schur complement solve is replaced by the NODE flux surrogate. Section 4 describes the offline training stage for the NODE flux surrogate. In Section 5 we present numerical results with NODE

---

\*University of Arizona, rmpawar@math.arizona.edu

†Sandia National Laboratories, pbboche@sandia.gov

‡Sandia National Laboratories, jowen@sandia.gov

surrogate-based scheme for a model advection-diffusion transmission problem. Finally, in Section 6 we offer some conclusions and comment on the efficacy of the NODE surrogate-based partitioned scheme.

**2. Implicit Value Recovery.** In this paper we restrict attention to a rectangular region  $\Omega \in \mathbb{R}^2$  partitioned into two non-intersecting subdomains  $\Omega_1$  and  $\Omega_2$  by an interface  $\gamma$ ; see Fig. 2.1. We orient the interface by a unit normal vector  $\mathbf{n}_\gamma$  and set  $\Gamma_i = \partial\Omega_i \setminus \gamma$ .

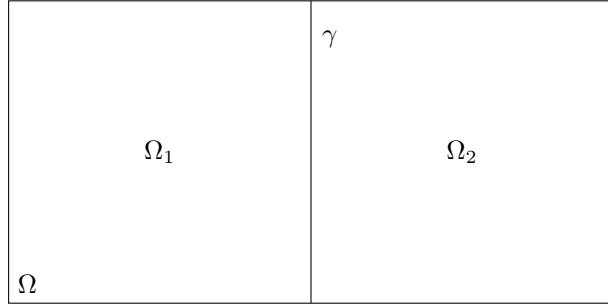


Fig. 2.1: The domain  $\Omega$  is divided into subdomains  $\Omega_1$  and  $\Omega_2$ .

We assume that each subdomain is endowed by an independently defined finite-element partition  $\Omega_i^h$  with mesh parameter  $h_i$ , vertices  $x_{i,r}$ , and elements  $K_{i,s}$ . The subdomain partitions induce finite element partitions  $\gamma_1^h$  and  $\gamma_2^h$  of the interface that are spatially coincident but not required to have matching nodes.

The model advection-diffusion transmission problem is defined by a pair of governing equations on each subdomain

$$\begin{cases} \dot{\varphi}_i - \nabla \cdot F_i(\varphi_i) = f_i & \text{in } \Omega_i \times [0, T] \\ \varphi_i = g_i & \text{in } \Gamma_i \times [0, T] \end{cases} \quad (2.1)$$

augmented with initial conditions,

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

and interface conditions,

$$\begin{aligned} 0 &= \varphi_1(\mathbf{x}, t) - \varphi_2(\mathbf{x}, t), \\ 0 &= F_1(\mathbf{x}, t) \cdot \mathbf{n}_\gamma - F_2(\mathbf{x}, t) \cdot \mathbf{n}_\gamma \end{aligned} \quad (2.3)$$

on  $\gamma \times [0, T]$ . The total flux  $F_i$  is assumed to have the form

$$F_i(\varphi_i) = \epsilon_i \nabla(\varphi_i) - \mathbf{u} \varphi_i \quad (2.4)$$

where  $\epsilon_i > 0$  is a diffusion coefficient in  $\Omega_i$  and  $\mathbf{u}$  is a given velocity field that can be zero.

Partitioned solution of (2.1), (2.2), and (2.3) is based on the observation that the transmission problem can be written in an equivalent form as

$$\begin{cases} \dot{\varphi}_i - \nabla \cdot F_i(\varphi_i) = f_i & \text{in } \Omega_i \times [0, T] \\ \varphi_i = g_i & \text{in } \Gamma_i \times [0, T] \\ F_i(\varphi_i) \cdot \mathbf{n}_i = (-1)^i \lambda & \text{on } \gamma \times [0, T] \end{cases} \quad i = 1, 2, \quad (2.5)$$

where  $\lambda$  denotes the interface flux. Formally, each equation in (2.5) is a mixed boundary value problem on its respective domain with Neumann data provided by the interface flux. These equations remain coupled because  $\lambda$  is one of the unknowns in this system. However, if one can obtain an approximation of the interface flux these equations can be solved independently, thereby decoupling the problem.

The IVR scheme [5] computes an approximation of the interface flux by forming and solving the Schur complement of a discrete monolithic system obtained by enforcing the alternative coupling condition

$$\varphi_1(\mathbf{x}, t) - \varphi_2(\mathbf{x}, t) = 0 \quad (2.6)$$

by a Lagrange multiplier. Assuming that the initial data is continuous along the interface, i.e.  $\varphi_0(\mathbf{x}^-) = \varphi_0(\mathbf{x}^+) \forall \mathbf{x} \in \gamma$  condition (2.6) is equivalent to the original one.

The resulting semi-discrete monolithic system is equivalent to the following system of differential algebraic equations (DAEs); see [5]:

$$\begin{bmatrix} M_1 & 0 & G_1^T \\ 0 & M_2 & -G_2^T \\ G_1 & -G_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1(\varphi_1) \\ \mathbf{f}_2(\varphi_2) \\ 0 \end{bmatrix}, \quad (2.7)$$

for finite element coefficient vectors  $\varphi_1, \varphi_2$ , and  $\lambda$ . Here,  $M_1, M_2$  are mass matrices, and  $G_1, G_2$  are matrices of inner products between the finite element bases functions and the test functions along boundary  $\gamma$ ; full definitions for these matrices can be found in [5].

The IVR scheme then proceeds to eliminate the interface flux from (2.7) by solving the system:

$$S\lambda = G_1 M_1^{-1} \mathbf{f}_1(\varphi_1) + G_2 M_2^{-1} \mathbf{f}_2(\varphi_2), \quad (2.8)$$

where

$$S = G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T \quad (2.9)$$

is the Schur complement of (2.7). It is straightforward to see that application of explicit time integration to the resulting coupled system of ODEs decouples this system and enables the independent solution of each subdomain ODE problem for  $\varphi_1$  and  $\varphi_2$ . Algorithm 1 summarizes the IVR scheme.

---

**Algorithm 1** Implicit Value Recovery (IVR) Algorithm Summary

---

**Require:** Evaluation of  $G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T$ ,  $\varphi_1^1$ , and  $\varphi_2^1$

**while**  $n < N_{max}$  **do**

1. **Compute**  $\mathbf{f}_i(\varphi_i^n)$  for  $i = 1, 2$ ,
2. **Solve:**

$$(G_1 M_1^{-1} G_1^T + G_2 M_2^{-1} G_2^T) \lambda^n = G_1 M_1^{-1} \mathbf{f}_1(\varphi_1^n) + G_2 M_2^{-1} \mathbf{f}_2(\varphi_2^n) \quad (2.10)$$

for  $\lambda^n$ .

3. **Solve**

$$M_i D^{n+1} \varphi_i = \mathbf{f}_i(\varphi_i^n) + (-1)^i G_i^T \lambda^n \quad i = 1, 2, \quad (2.11)$$

for  $\varphi_i^{n+1}$ , where  $D^{n+1}$  denotes an explicit approximation of the time derivative at  $t^{n+1}$ .

**end while**

---

Although the IVR scheme can be implemented using lumped mass matrices, in order to achieve optimal accuracy one has to employ consistent mass matrices. In this case, forming the Schur complement in Step 2 involves inversion of the sparse matrices  $M_i$ , which results in a dense Schur complement matrix. In principle, one could precompute and store the Schur complement in factored form, however, the storage burden may be unacceptable in three dimensions and even in some two-dimensional configurations employing very fine meshes.

In this paper we investigate a data-driven alternative in which the Schur complement system for the interface flux is replaced by a data-driven surrogate for the dynamics of this variable. To that end we shall use Neural ODEs which can be viewed as a generic nonlinear system of ODEs with a right hand side function parameterized by a Deep Neural Network (DNN); see, e.g., [4, 1]. The weights and the biases of this DNN are inferred offline from a suitable time series data of the solutions of the transmission problem. Once the NODE surrogate is trained we propose to use it in Step 2 of the IVR algorithm as cost and storage effective replacement for the Schur complement.

**3. Neural ODEs Flux Surrogate.** Neural ODEs are a relatively recent development introduced in 2019 [1], that has attracted a significant interest in multiple application areas; see, e.g., [2, 3, 4]. NODEs can be viewed as a limit of a Residual Neural Network (ResNet) architecture as the number of its layers tends to infinity. Specifically, the ResNet

$$h_{t+1} = h_t + \mathcal{NN}(h_t, \theta_t), \quad (3.1)$$

where  $t \in \{0, \dots, T\}$  is the ResNet layer number,  $h_t \in \mathbb{R}^D$  is the hidden state at layer  $t$ , and  $\mathcal{NN}$  is a functional representation of the layer, can be interpreted as a forward Euler discretization of an underlying continuous system of ODEs with  $\Delta t = 1$ . Taking a limit in the number of layers then yields

$$\frac{dh(t)}{dt} = \mathcal{NN}(h(t), t, \theta), \quad (3.2)$$

with the input of  $h(0)$  and final output  $h(T)$  [1]. Conversely, it is easy to see that discretization of (3.2) by the forward Euler scheme recovers (3.1).

The right hand side function  $\mathcal{NN}(h(t), t, \theta)$  in (3.2) is parameterized by weights and biases denoted by  $\theta$ . These parameters are inferred by minimizing the loss between the output of (3.2) and suitable time series data using backpropagation. Note that in practice, the solution of (3.2) has to be computed by a numerical time integration scheme. In the next section we specialize (3.2) to obtain a flux surrogate for our partitioned scheme.

**3.1. NODE Surrogate-Based Partitioned Scheme.** To develop the NODE flux surrogate we consider inputs defined by concatenation of the states and the fluxes from each side of the interface, i.e.,

$$\boldsymbol{x}^n = \begin{bmatrix} \boldsymbol{\varphi}_1^n \\ \boldsymbol{\varphi}_2^n \\ \boldsymbol{\lambda}_1^{n-1} \\ \boldsymbol{\lambda}_2^{n-1} \end{bmatrix} \quad (3.3)$$

with trained parameters  $\boldsymbol{\theta}$ . For convenience, we define

$$\boldsymbol{\lambda}^n = \begin{bmatrix} \boldsymbol{\lambda}_1^n \\ \boldsymbol{\lambda}_2^n \end{bmatrix}. \quad (3.4)$$

We choose (3.3) as our state since its components represent all relevant variables for the coupled monolithic DAE (2.7) computed during a single step of the IVR algorithm. We also note that using  $\boldsymbol{\lambda}$  alone would not represent the state of a closed dynamical system.

We provide details about the training of the NODE surrogate in Section 4. Assuming that such a surrogate is available we use it to compute an approximation of the interface flux at  $t^n$  by forming the state  $\mathbf{x}^n$  and solving numerically the NODE surrogate for one time step. The flux component  $\boldsymbol{\lambda}^n$  is then extracted from the integrated state and used to decouple the equations for  $\varphi_i$ . Algorithm 2 summarizes the resulting NODE surrogate-based partitioned scheme.

---

**Algorithm 2** NODE Surrogate-based partitioned scheme

---

**Require:**  $\mathcal{NN}(\cdot, \boldsymbol{\theta})$  and  $\mathbf{x}^0$

**while**  $n < N_{max}$  **do**

1. **Compute**  $f_i(\varphi_i^n)$  for  $i = 1, 2$ ,
2. **Assemble**  $\mathbf{x}^n$  from  $\boldsymbol{\lambda}^{n-1}$ ,  $\varphi_1^n$ , and  $\varphi_2^n$ .
3. **Solve:**

$$\dot{\mathbf{x}} = \mathcal{NN}(\mathbf{x}^n, \boldsymbol{\theta}) \quad (3.5)$$

for  $\bar{\mathbf{x}}^{n+1}$ .

4. **Extract**  $\boldsymbol{\lambda}_1^n$  and  $\boldsymbol{\lambda}_2^n$  from  $\mathbf{x}^{n+1}$ .
5. **Solve**

$$M_i D^{n+1} \varphi_i = \mathbf{f}_i(\varphi_i^n) + (-1)^i G_i^T \boldsymbol{\lambda}_i^n, i = 1, 2 \quad (3.6)$$

for  $\varphi_i^{n+1}$ .

**end while**

---

**4. Training the Model.** We now describe the offline training stage for the NODE flux surrogate.

**4.1. Training Data.** To generate training data for the NODE flux surrogate we use the IVR scheme to solve the model transmission problem for an initial condition given by a Gaussian distribution centered at a point  $(x, y) \in \Omega$  and a rotating velocity field; see Figure 4.1 for a sample solution trajectory. To compute the solution we use a uniform mesh with a total of 2145 degrees of freedom in each subdomain and 65 degrees of freedom for the interface flux.

The training set comprises time series data for solution trajectories corresponding to 9 Gaussian initial conditions centered along the line  $y = 0.5$ . The  $x$ -coordinates of the initial conditions are given by  $x = \{0.05, \dots, 0.45\}$  with increments of 0.05. The transmission problem is simulated for one full revolution using  $n=1866$  time steps. Figure 4.1 shows a few snapshots of a representative solution trajectory.

The raw training data consists of solution states  $\varphi_i^n \in \mathbb{R}^{2145}$  and interface fluxes  $\boldsymbol{\lambda}_i^{n-1} \in \mathbb{R}^{65}$  and  $n = \{1, \dots, 1866\}$ . To define the training data for the NODE surrogate  $\varphi_i^n$  and  $\boldsymbol{\lambda}_i^{n-1}$  are stacked on top of each other as follows:

$$\mathbf{x}^n = \begin{bmatrix} \varphi_1^n \\ \varphi_2^n \\ \boldsymbol{\lambda}_1^{n-1} \\ \boldsymbol{\lambda}_2^{n-1} \end{bmatrix} \in \mathbb{R}^{4420}. \quad (4.1)$$

For faster training, an augmentation variable  $\mathbf{a} = \mathbf{0} \in \mathbb{R}^{20}$  is further concatenated with  $\mathbf{x}^n$

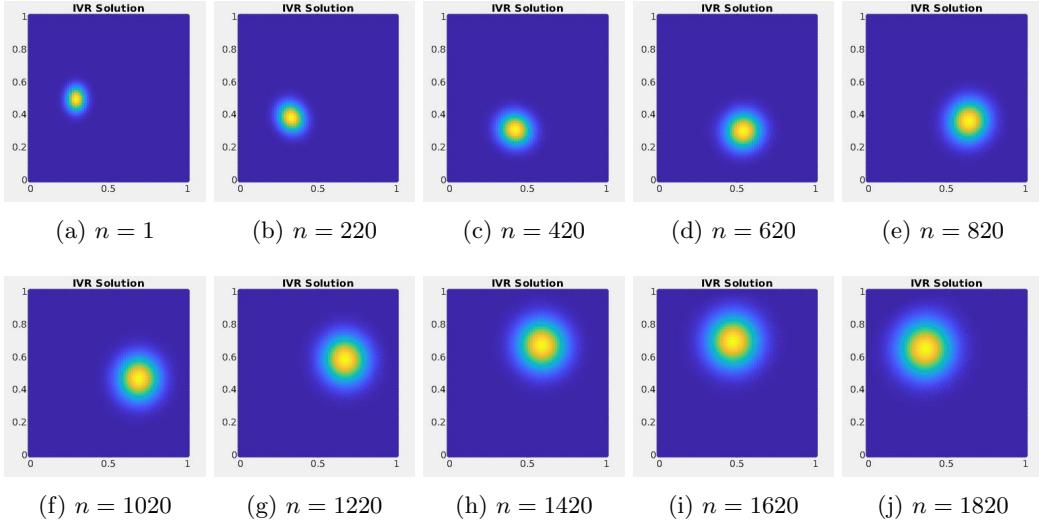


Fig. 4.1: A representative solution trajectory corresponding to a Gaussian initial condition centered at  $(0.3, 0.5)$ .

to obtain an augmented NODE state given by

$$\boldsymbol{x}^n = \begin{bmatrix} \boldsymbol{x}^n \\ \boldsymbol{a} \end{bmatrix}. \quad (4.2)$$

Augmentation of the NODE state increases its dimensionality and accelerates the training process; see [2].

The solution states in the training data are generally on the order of  $O(10^{-4})$  while the flux variables are generally on the order of  $O(10^{-8})$ . Additionally, there are 4290 solution variables to the 130 flux variables. As a result, a standard loss function based on this data will tend to favor the states over the fluxes during the training process. To equilibrate the contributions of these variables we scale the flux data  $\boldsymbol{\lambda}_i^{n-1}$  by a factor that makes it comparable in magnitude and cardinality to the solution data  $\boldsymbol{\varphi}_i^n$ .

The following summarizes the training process for the NODE flux surrogate.

1. Define a mini-batch function as follows:

$$\mathcal{M} : \boldsymbol{x}_i^n \rightarrow \{\boldsymbol{X}_i^{n+1}, \dots, \boldsymbol{X}_i^{n+S}\}, \quad (4.3)$$

where  $\boldsymbol{x}_i^n$  is an augmented state vector,  $S$  is the number of time steps to model over, and  $i$  refers to the  $i$ th sample from the mini-batch size  $B$ . The mini-batch function associates an augmented state vector  $\boldsymbol{x}_i^n$  with its succeeding  $S$  state vectors.

2. **Predictor step.** Solve numerically<sup>1</sup> the NODE

$$\dot{\boldsymbol{X}}_i = \mathcal{NN}(\boldsymbol{x}_i^n, \theta). \quad (4.4)$$

to compute approximate solution trajectories  $\{\mathcal{T}^{n+s}(\boldsymbol{x}_i^n, \theta)\}_{s=1}^S$  emanating from  $\boldsymbol{x}_i^n$ .

---

<sup>1</sup>In this work we used Matlab's RK45 routine.

- 3. Loss Function Computation.** Use the approximate trajectories to form the loss function for the  $i$ th component of the mini-batch:

$$L_i := \frac{1}{S} \sum_{s=1}^S (\mathbf{X}_i^{n+s} - \mathcal{T}^{n+s}(\mathbf{x}_i^n, \theta))^2. \quad (4.5)$$

To determine the loss over the entire mini-batch, the average of losses  $L_i$  over the mini-batch size  $B$  is taken.

- 4. Parameter Update.** Use back propagation to update the parameters  $\theta$ .

For training data containing  $I$  different trajectories, trajectories are distinguished by their initial conditions. To train over this more general selection, a list of randomly generated indices from 1 to  $I$  is created based off the size of the desired mini-batch. Then a secondary list, of the same length, of indices from 1 to  $1866 - S$  is randomly generated; this selects the starting point for a training trajectory. A pairing between the trajectory index and the starting point is made and this combination specifies a sample trajectory to use in the mini-batch.

**5. Results.** In this section we provide numerical results illustrating the NODE surrogate-based partitioned scheme. The experiments were performed using single and multiple trajectories as the training data.

**5.1. Network Configuration.** In the present context the “best” configuration for the NODE function defining the right hand side of the NODE flux surrogate was found to employ  $\tanh x$  as an activation function and 3 hidden layers with a condense and expand framework of 1000-500-1000 nodes. This network configuration experienced the fastest loss reduction.

The Adam optimizer was used with a starting learning rate parameter of 0.002. The network would be trained until the loss became stationary. The next step was to reduce the learning rate parameter by magnitude 10 and to resume training. The network would be trained until the lowest possible loss, as defined in (4.5), was achieved.

To assess the accuracy of the NODE surrogate as an approximate solver of the model problem we used an additional metric defined as the relative  $L^2$  error in the state prediction, i.e.,

$$\text{Error} = \frac{\|\mathbf{Y}_{\text{NODE}}^n - \mathbf{Y}^n\|_2}{\|\mathbf{Y}^n\|_2}, \quad (5.1)$$

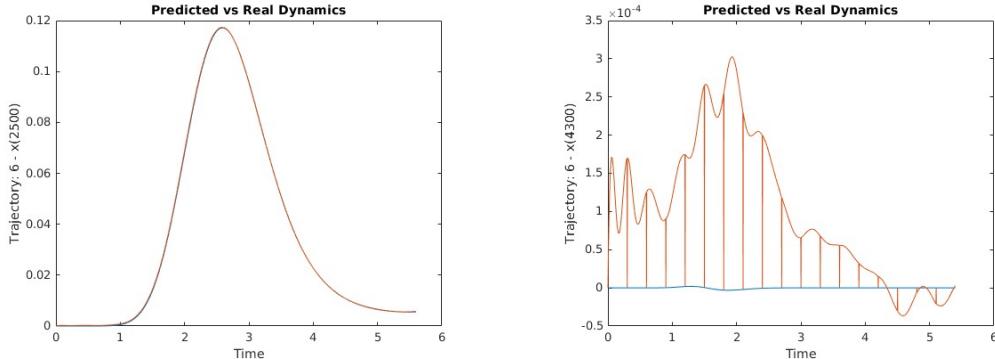
where

$$\mathbf{Y}^n = \begin{bmatrix} \boldsymbol{\varphi}_1^n \\ \boldsymbol{\varphi}_2^n \end{bmatrix} \quad (5.2)$$

and  $\mathbf{Y}_{\text{NODE}}^n$  is obtained from the approximate NODE solution. This error estimates the accuracy of the NODE to model the solution vector  $\mathbf{Y}^n$ , from the IVR solution, given the previous solution vector  $\mathbf{Y}^{n-1}$  and flux vector  $\boldsymbol{\lambda}^{n-2}$ .

**5.2. Preliminary observations.** We first illustrate the importance of proper scaling of the flux data in order to ensure that its contribution to the loss function is balanced with that of the subdomain states. Figure 5.1 shows representative results obtained with unscaled flux data. From the plots in this figure it is clear that while the trained NODE would learn the dynamics of the solution data, see Figure 5.1a, it would have difficulty learning the flux data, see Figure 5.1b.

Furthermore, if the flux data was not appropriately scaled the relative error of the NODE flux prediction would be on the order of  $O(10^4)$ , see Figure 5.2. To see the consequence of such high relative error, see Figure 5.3.



(a) The time evolution of component 2500 of the state vector (corresponding to component 2500 of the solution vector) is accurately approximated by the trained NODE (orange curve).

(b) For unscaled flux components in the state vectors, the time evolution of component 4300 of the state vector (component 10 of the corresponding flux vector) yields a visibly poor approximation by the trained NODE.

Fig. 5.1: The NODE tends to learn the dynamics of the solution components rather than the flux components of the state vector when the latter are not scaled.

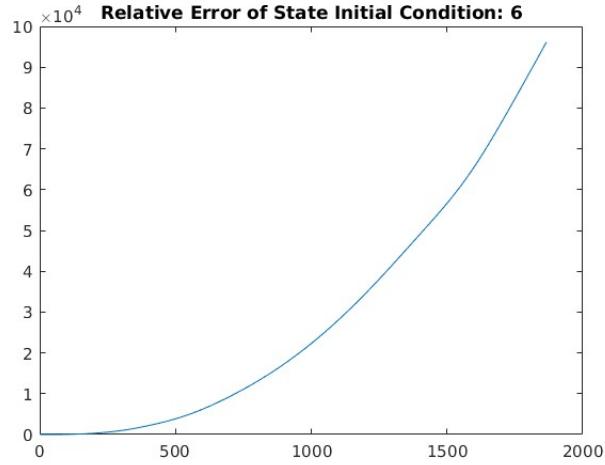


Fig. 5.2: Relative NODE solution error (5.1) as a function of time when the NODE is trained using unscaled flux data. The horizontal axis represents the time steps from 1 to 1866.

**5.3. Reproductive Tests of the NODE Surrogate-Based Partitioned Scheme.** In this section we focus on a reproductive test of Algorithm 2 implemented with an NODE

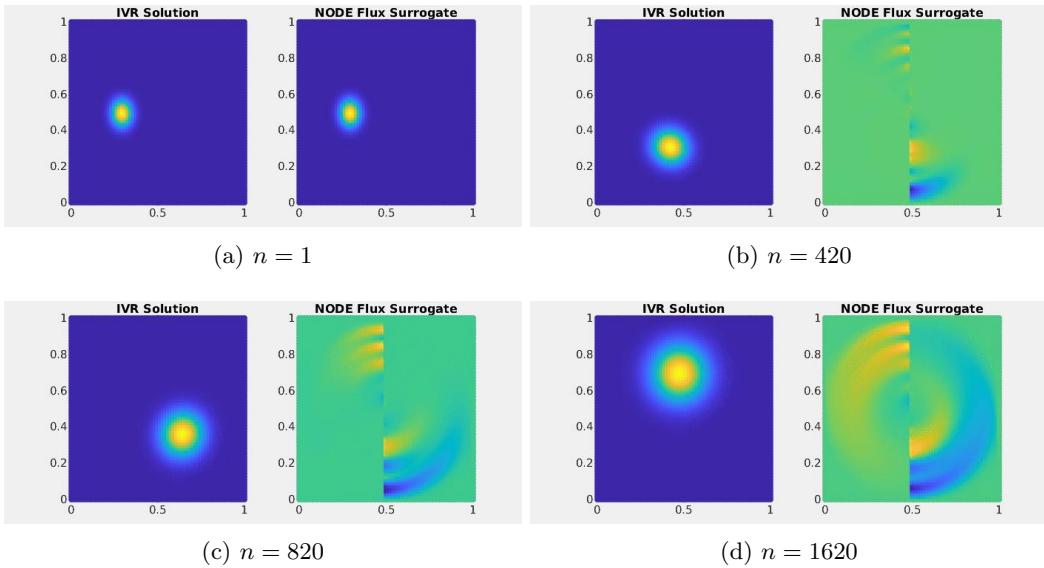


Fig. 5.3: NODE surrogate-based partitioned solution using an NODE surrogate trained with unscaled flux data.

flux surrogate trained with properly scaled flux data. The right hand side of the NODE is configured using the neural net architecture described in Section 5.1 and the flux is scaled by a factor of  $10^4$ . The training set for this test comprises a time series with the solution trajectory for a single Gaussian initial condition. The trained NODE flux surrogate is then used in Algorithm 2 to solve the transmission problem for the same initial condition as in the training data.

Results for this single trajectory reproductive test are shown in Figure 5.4. The plots in this figure compare snapshots of the NODE surrogate-based partitioned solution with the IVR solution computed using the same initial condition. Compared to the plots in Figure 5.3 we see that proper scaling of the flux significantly improves the surrogate-based solution to a point where it is not visibly distinguishable from the benchmark IVR solution.

These results confirm that a NODE flux surrogate can learn the dynamics of the interface flux for a single solution trajectory. Plots of the relative NODE solution error in Figure 5.5 also suggest that the flux surrogate has reasonable accuracy with relative errors on the order of  $O(10^{-3})$ .

**5.4. Generalization to Arbitrary Initial Conditions Using Multiple Trajectory Training.** The reproductive test in Section 5.3 reveals that the NODE flux surrogate can learn and accurately reproduce the dynamics of the flux for a single solution trajectory. In this section we focus on extending the NODE flux surrogate to arbitrary initial condition data. To that end we consider the training data from Section 4.1 comprising nine solution trajectories corresponding to nine different Gaussian initial conditions. We use the same scaling factor for the flux as in the reproductive test, i.e., this variable is scaled by  $10^4$ .

We then performed predictive tests of the NODE flux surrogate-based scheme by using Gaussian initial conditions outside of the training set. Our simulation results indicate that for initial Gaussian conditions centered near the initial conditions in the training set, the

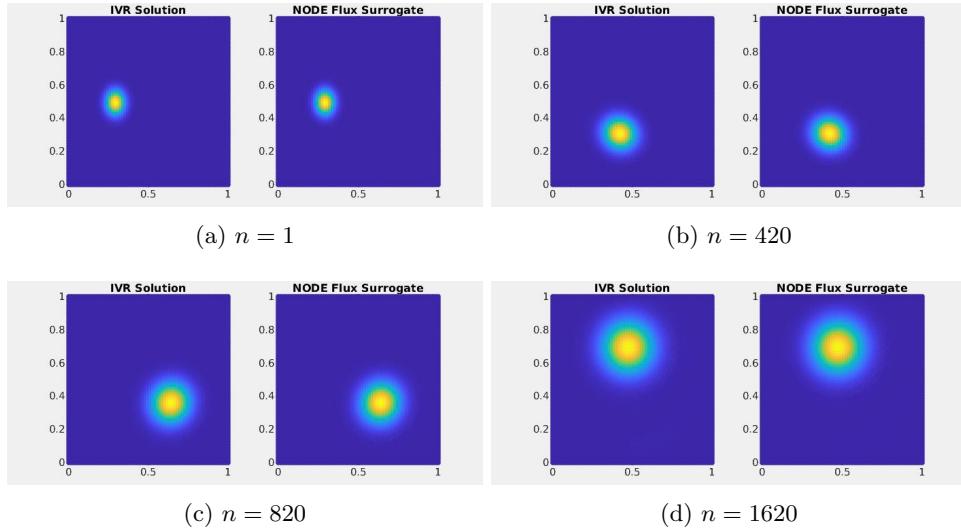


Fig. 5.4: Reproductive test of the NODE surrogate-based partitioned scheme for a single solution trajectory. The NODE surrogate was trained using scaled flux data.

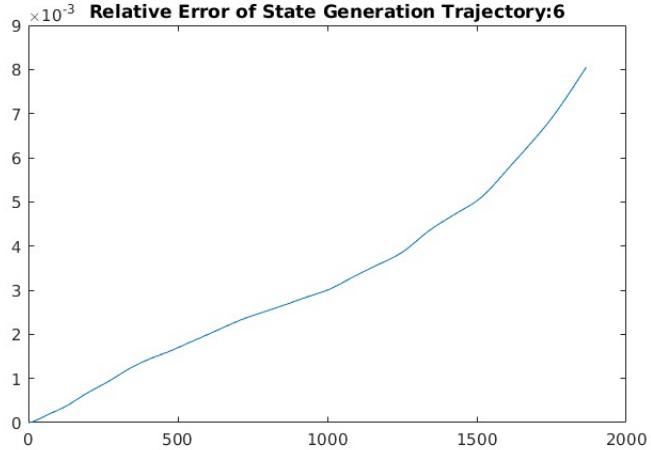
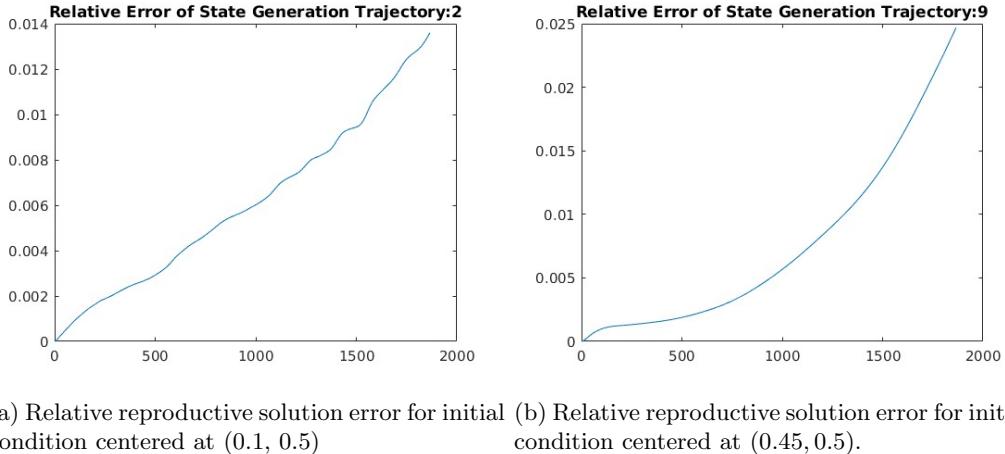


Fig. 5.5: The x-axis represents the time steps  $n = 1$  to 1866. The NODE's flux approximation integrated with the IVR yields low relative error, measured with respect eq. (5.1).

surrogate-based scheme yields solutions that are qualitatively similar to the benchmark IVR solutions. These observations hold true for any Gaussian initial condition whose  $x$  component lay in the interval  $[0.05, 0.5)$  and whose  $y$  component was 0.5; see Figures 5.9a and 5.9b for relative error plots and Figures 5.10 and 5.11 for snapshots of the Gaussian trajectory. Gaussians whose initial  $y$  value was further than 0.05 units away from 0.5 or whose  $x$  component was greater than 0.5 yielded visibly poor results.

We also performed reproductive tests to confirm that the larger training set does not

compromise the NODE surrogate accuracy for the individual members of that training set. For these test we found that the relative errors are kept on order  $O(10^{-2})$ , see Figures 5.6a and 5.6b. The modeled solution data using the NODE's flux prediction can be seen in Figures 5.7 and 5.8.



(a) Relative reproductive solution error for initial condition centered at  $(0.1, 0.5)$ . (b) Relative reproductive solution error for initial condition centered at  $(0.45, 0.5)$ .

Fig. 5.6: Relative NODE reproductive solution error (5.1) as a function of time when the NODE is trained using scaled flux data. The horizontal axis represents the time steps from 1 to 1866.

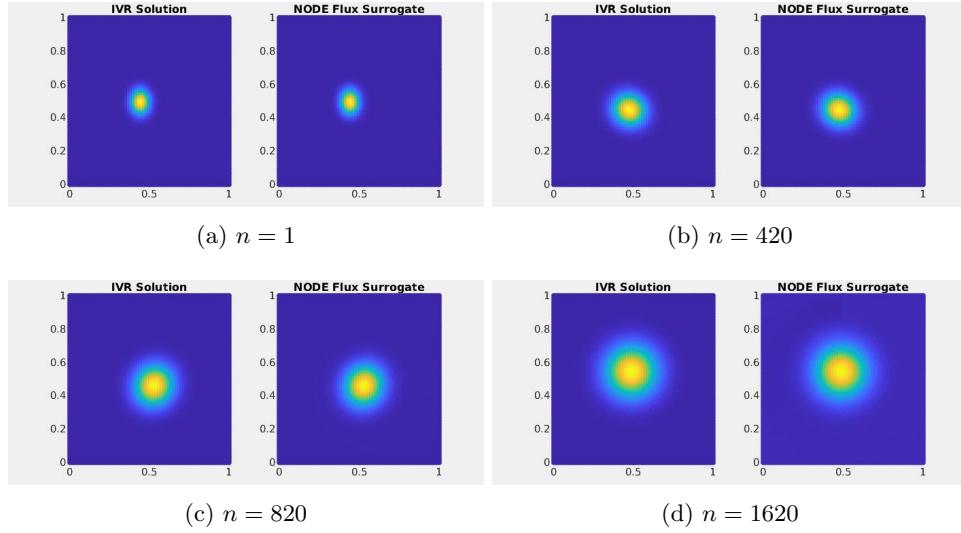


Fig. 5.7: Reproductive test for a solution trajectory with initial Gaussian data centered at  $(0.45, 0.5)$ . NODE flux surrogate trained on 9 solution trajectories and scaled flux data.

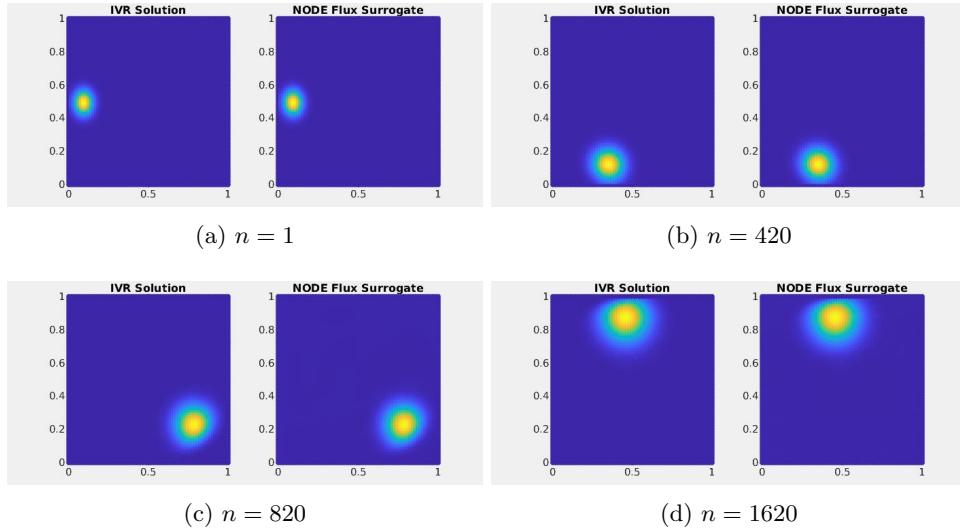
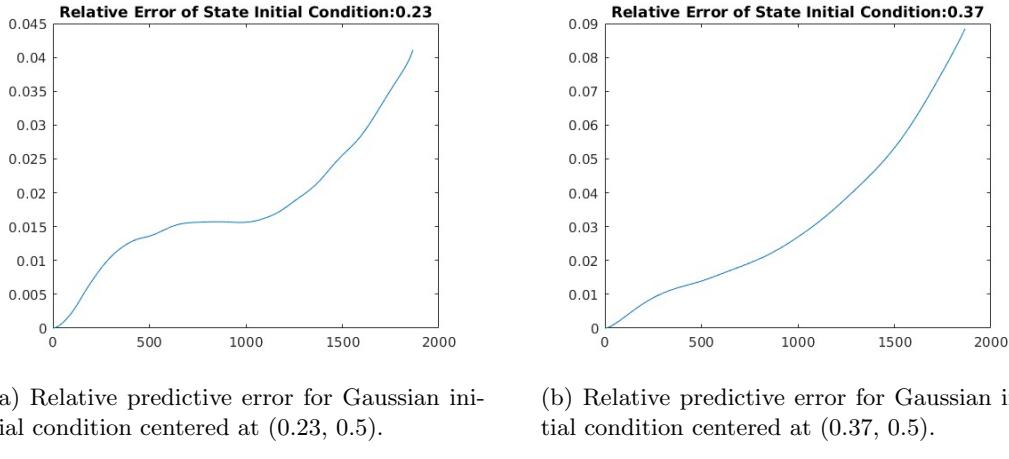


Fig. 5.8: Reproductive test for a solution trajectory with initial Gaussian data centered at  $(0.1, 0.5)$ . NODE flux surrogate trained on 9 solution trajectories and scaled flux data.



(a) Relative predictive error for Gaussian initial condition centered at  $(0.23, 0.5)$ .  
(b) Relative predictive error for Gaussian initial condition centered at  $(0.37, 0.5)$ .

Fig. 5.9: Relative NODE predictive solution error (5.1) as a function of time when the NODE is trained using 9 trajectories and scaled flux data and initial condition is centered within the training data range. The horizontal axis represents the time steps from 1 to 1866.

However, if one attempts to apply the NODE surrogate to simulate solution trajectories with Gaussian initial conditions centered outside the range of the centers of the training data, the accuracy of the NODE solution deteriorates significantly; see Figures 5.12a and 5.12b for relative NODE solution error plots and Figures 5.13 and 5.14 for solution trajectories computed by the NODE surrogate-based partitioned scheme in Algorithm 2.

These preliminary results are encouraging and indicate that it is possible for a NODE to

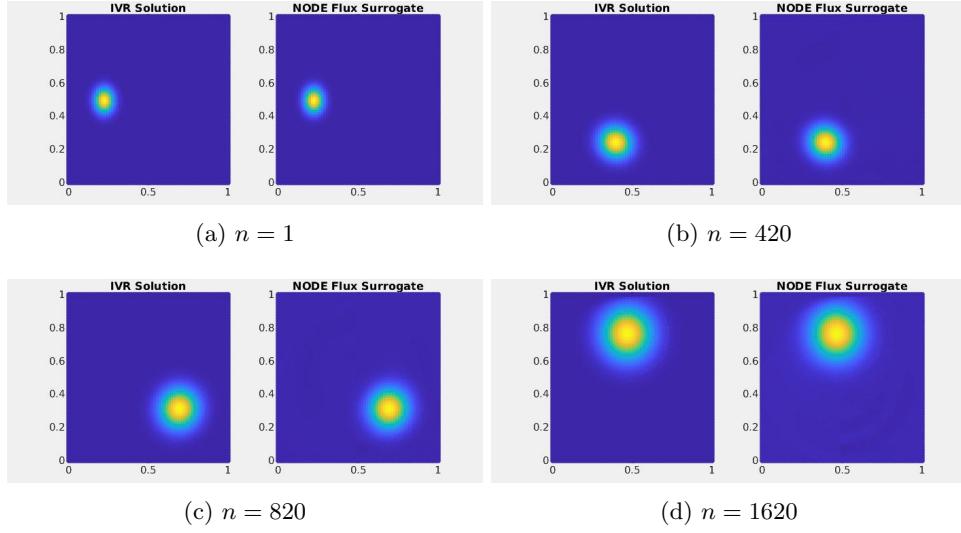


Fig. 5.10: Predictive test of the NODE flux surrogate-based scheme for Gaussian initial data centered at  $(0.23, 0.5)$ . NODE flux surrogate trained on 9 solution trajectories and scaled flux data.

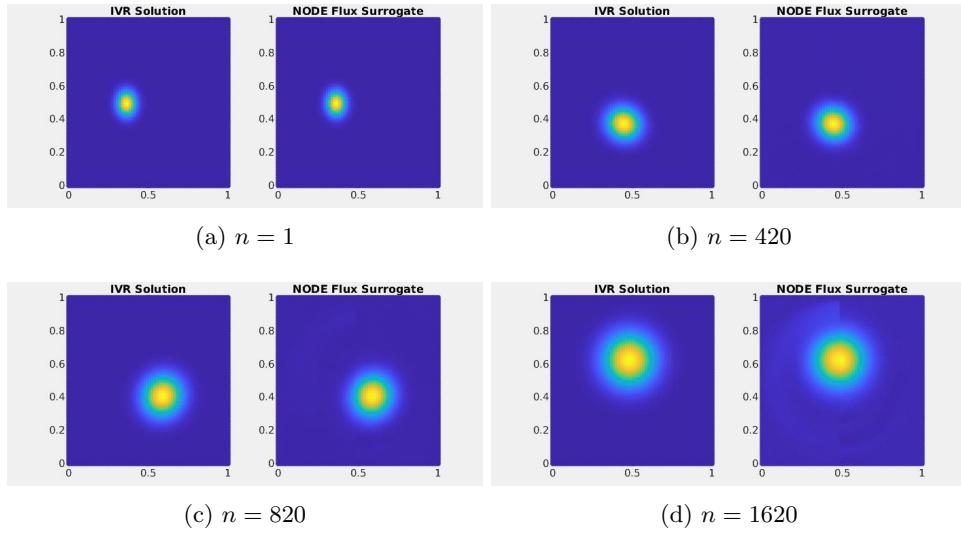


Fig. 5.11: Predictive test of the NODE flux surrogate-based scheme for Gaussian initial data centered at  $(0.37, 0.5)$ . NODE flux surrogate trained on 9 solution trajectories and scaled flux data.

learn the dynamics of the interface flux for more general initial conditions. Our results also suggest that the training data should be designed to represent well the possible locations of the initial conditions. For example, using a grid of initial conditions that sample from the

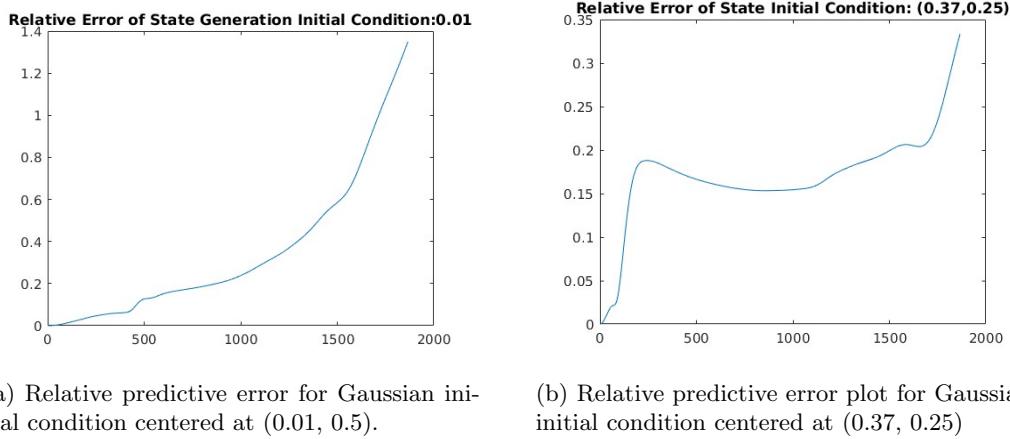


Fig. 5.12: Relative NODE predictive solution error (5.1) as a function of time when the NODE is trained using 9 trajectories and scaled flux data and the initial condition is centered outside the training data range.

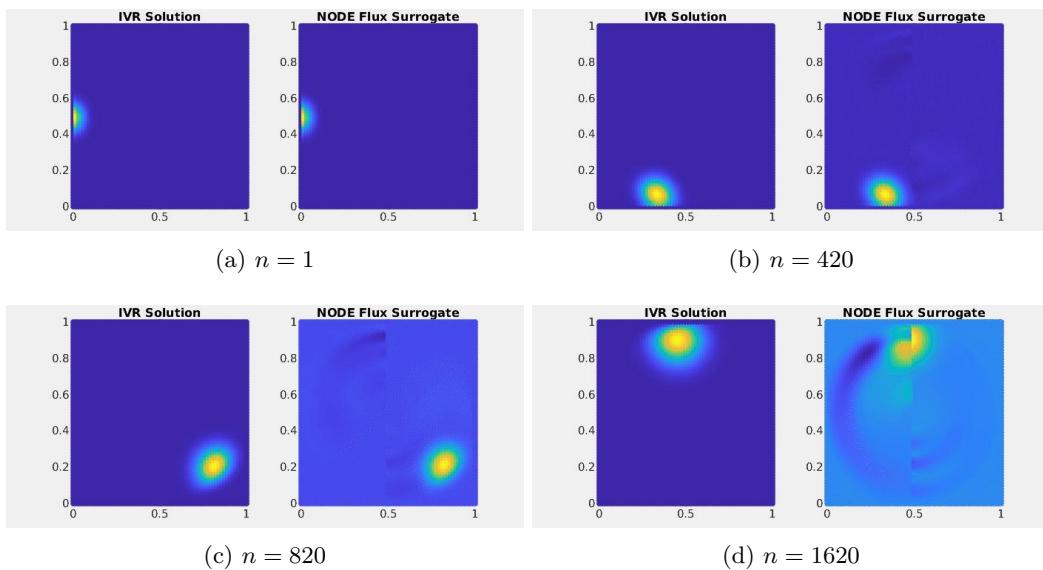


Fig. 5.13: Predictive test of the NODE flux surrogate-based scheme for Gaussian initial data centered at  $(0.01, 0.5)$ . NODE flux surrogate trained on 9 solution trajectories and scaled flux data.

entire domain, consequently both subdomains, is likely to yield better modeling results.

**6. Conclusions.** In this work we developed and demonstrated a surrogate-based partitioned scheme in which the interface flux is approximated by a NODE flux surrogate trained

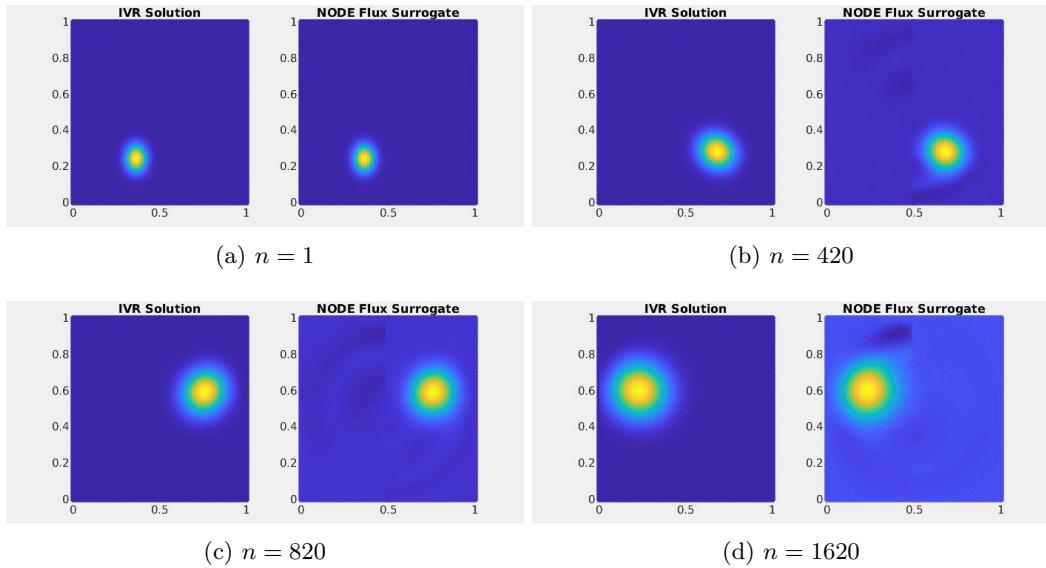


Fig. 5.14: For initial condition  $(0.37, 0.25)$ , the NODE's flux approximations are visually inaccurate.

on suitable solution data. Reproductive tests with the NODE surrogate trained on both single and multiple solution trajectories show acceptable accuracy and potential for the surrogate flux to serve as a more cost-effective substitute for, e.g., the Schur complement flux in the IVR scheme.

Predictive tests with the NODE surrogate reveal the importance of training data selection. Specifically, for an NODE trained using more diverse data, the surrogate-based partitioned scheme can accurately model the state data with previously seen initial conditions. The NODE is further able to generalize to unseen initial conditions, but these initial conditions must be centered close to those from the training data. If the initial condition centers are too far from the centers of the initial conditions in the training data, the accuracy of the surrogate-based partitioned scheme significantly deteriorates.

These observations suggest that for generalization to initial conditions outside of the training data, more varied data should be used in the training process. In particular, trajectories with initial conditions sampled throughout the subdomains should be used in the training process. Concomitantly, a different NODE architecture with more hidden layers and parameters might be required.

Additional, less obvious paths for improvement are also possible. These will be investigated in future works and include:

- Leveraging Properties of the Governing Equations: The NODEs trained in this report do not use a priori information of the process being modeled. Knowledge of the velocity field along the interface would provide information on how the solution and flux vectors evolve. Incorporating this information into the NODE Network would expedite training and improve modeling ability.
  - Generalized Scaling: Since scaling the flux data by  $10^4$  was an ad hoc method, a more general method is desirable because scaling by  $10^4$  isn't guaranteed to always

work. One approach to consider is to train a NODE that will learn the scaling parameters for each state variable and then using those scales for training.

- Utilization of logarithmic Mean Square Loss function: Such a function is appropriate for data fields that differ by orders of magnitude.
- Finer Sampled Data: Since error tends to increase once the Gaussian crosses the interface, training over more finely sampled data around the interface might yield better modeling results.
- Control Implementation: Implementing a control scheme is another possible improvement. One possible suggestion would have the input state be the solution vectors and have the control variables interact with the flux vector to obtain the next flux vector in time.

In addition to the aforementioned improvements to be investigated, we would also like to explore the ability of Algorithm 2 to generalize to parameterized PDEs.

## REFERENCES

- [1] R. T. CHEN, Y. RUBANOVA, J. BETTENCOURT, AND D. K. DUVENAUD, *Neural ordinary differential equations*, Advances in neural information processing systems, 31 (2018).
- [2] E. DUPONT, A. DOUCET, AND Y. W. TEH, *Augmented neural odes*, Advances in neural information processing systems, 32 (2019).
- [3] T. Z. JIAHAO, M. A. HSIEH, AND E. FORGOSTON, *Knowledge-based learning of nonlinear dynamics and chaos*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 31 (2021).
- [4] S. MASSAROLI, M. POLI, J. PARK, A. YAMASHITA, AND H. ASAMA, *Dissecting neural odes*, Advances in Neural Information Processing Systems, 33 (2020), pp. 3952–3963.
- [5] 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. Proceedings of the Eight International Conference on Numerical Methods for Multi-Material Fluid Flows (MULTIMAT 2017).

## MACHINE LEARNED INTERATOMIC POTENTIALS FOR GRAIN BOUNDARY SEGREGATION IN ALLOYS

KYREL POLIFRONE\*, HADIA BAYAT†, JAMES MICHAEL GOFF‡, AND WENWU XU§

**Abstract.** This study outlines the process of developing a machine-learned potential for a Nichrome alloy system. The goal is to study the atomistic dynamics which may lead to the phenomenon observed from laboratory TEM images where Cerium-rich precipitation occurs at grain boundaries and grain boundary junctions while under the influence of nano-current electrical pulses within a Nichrome system. Machine-learned interatomic potentials often suffer from spuriously large attractive forces when atoms are close together. This presents a problem for modeling NiCr alloys in electric fields where close interatomic separations may be induced by the field. We address this by building a comprehensive training data set and applying new regression methods. A diverse set of training data was generated and simulated with Quantum ESPRESSO based on simulating bulk modulus, equation of state, ab-initio molecular Dynamics, and genetic algorithm structures for both Nickel and Chromium. These simulation results were used in FitSNAP to create, as well as optimize the two generated potentials for Nickel and Chromium. These preliminary potentials have been tested and showed that spurious attractive forces can be resolved using regularization schemes that penalize large changes in attractive forces at short bond lengths. This resolves key problems for training ML potentials for NiCr in E-fields as well as other materials in extreme conditions.

**1. Introduction.** Laboratory experiments often give insight into further understanding the causes of specific effects seen within a system. However, sometimes an experiment may show results that are not easily understood. In situations like this, models of specific parts of a system can be used to isolate the region of interest for a deeper study into the cause of an effect. Laboratory Transmission Electron Microscope (TEM) image in Figures 1.1a and 1.1b, generated by Dr. Wenwu Xu and a team of researchers from San Diego State University revealed a particular phenomenon for Nichrome alloy with a composition of approximately Cr-20at%, Si-1.5at%, Ce less than 1-at%, with the balancing of Nickel for the rest. While under the influence of high-intensity electrical current nano-pulsing, the Nichrome solid solution alloy revealed Ce-rich precipitation along grain boundaries (GB) and junction, which is not typically observed under thermal-only processing. The phenomenon of Cerium-rich precipitation has driven our desire to conduct Molecular Dynamics (MD) simulations, aiming to model the migration of Cerium atoms within the Nichrome alloy. The objective is to gain a deeper understanding of the underlying causes of this behavior. However, the immediate execution of these MD simulations faced the challenge of a lack of a suitable potential to accurately represent the unique characteristics of Cerium atoms within the Nichrome system.

To overcome this limitation, our initial goal was to develop a potential capable of modeling a Nichrome alloy system accurately. While traditional MD simulation methods are available for potential development, recent years have seen the emergence of newer tools that leverage machine learning (ML) algorithms and models to optimize material systems more efficiently and comprehensively. In this context, density functional theory (DFT) based simulation methods serve as the foundation for the creation of the final potential file.

DFT simulations, grounded in the fundamental principles of quantum mechanics, enable precise calculations of the atomic and molecular structures of materials [3]. This study focuses on the initial stages of developing a potential for a Nichrome alloy system. It utilizes an ML-assisted potential constructed with training data generated by Quantum ESPRESSO (QE) and fine-tunes this potential to match the training data using FitSNAP. The overarching aim is to describe the material's properties under various conditions, including interactions with electrical fields [9]. The primary objective, aligned with the goal of developing the potential to compare results with laboratory TEM images, begins by creating a potential that models the atomistic properties of a simplified Nichrome

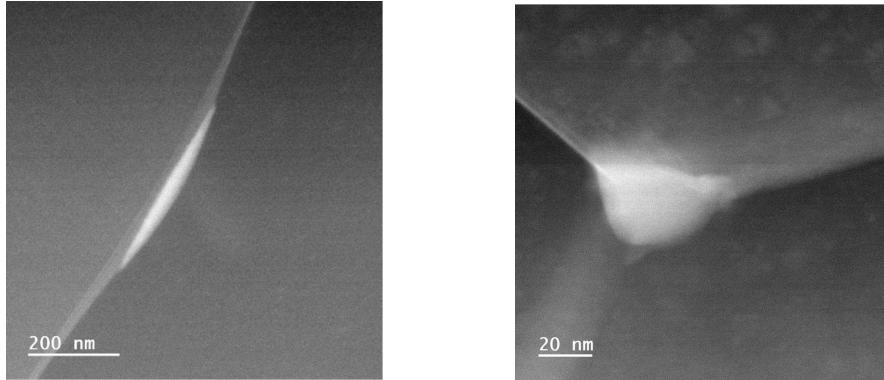
---

\*San Diego State University

†San Diego State University, hbayat7188@sdsu.edu

‡Sandia National Laboratories, jmgoff@sandia.gov

§San Diego State University, wenwu.xu@sdsu.edu



(a) Ce-rich precipitation along grain boundary by electrical current stressing.  
(b) Ce-rich precipitation at grain boundary junction by electrical current stressing.

Figure 1.1: TEM Images of Ce-rich Precipitate

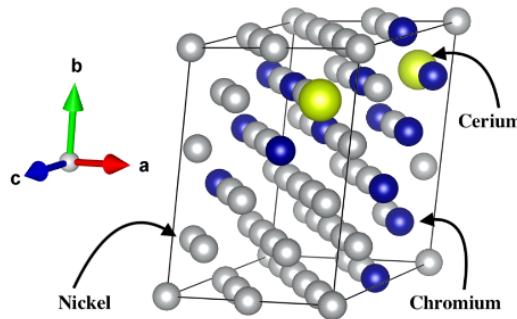


Figure 1.2: Special Quasirandom Structure for Nickel Alloy

system. This system comprises only Nickel and Chromium atoms. Subsequently, the process will extend to a more complex system that also includes Cerium and Silicon.

**2. Methodology.** The process of designing the desired potential is a four-part process where each step works along with one another to achieve the final product. The design of the potential involves: generating training data, forming the model, optimizing the potential, and testing results in a simulation engine.

To start the development process for the desired potential, the first step required generating training data. This process was done using the software Quantum ESPRESSO. QE is a set of codes developed to be used for calculating electronic structures and material modeling based on DFT, plane wave basis sets, and pseudopotentials for modeling interactions between electrons and ions [3]. DFT, density functional theory, attempts to solve the quantum mechanics equations used to describe fundamental properties differently than conventional methods. DFT uses one-body density as the fundamental variable, rather than conventionally using the many-body wave function [11]. DFT calculations are built on the foundation of Hohenberg-Kohn and Kohn-Sham theorems which describe the interactions of electrons and ions.

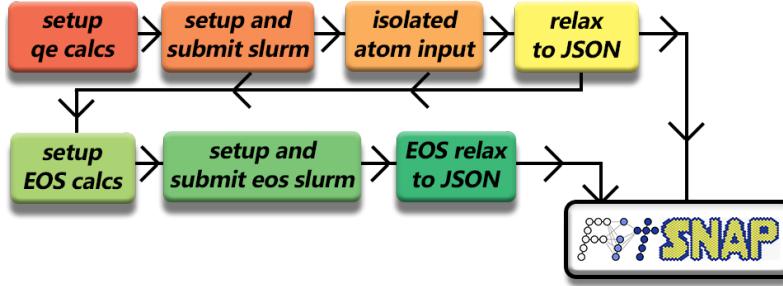


Figure 2.1: Training data process roadmap.

In ground-state DFT, the Hamiltonian can be used to describe the system of  $N$  number of interacting electrons with,

$$H\Psi(\mathbf{r}_1 \cdots \mathbf{r}_N) = E\Psi(\mathbf{r}_1 \cdots \mathbf{r}_N). \quad (2.1)$$

The approximate solution to this allows us to obtain accurate energy predictions for atomic structures to use as ground truth. The Hellman-Feynmann theorem allows us to evaluate accurate atomic forces to include in training. Though not explicitly written, Eq. (2.1) may be generalized to include spin. For all calculations, spin-polarized DFT was used with randomized initial spin states.

Inputs for QE also include the pseudopotential, which is a set of inputs used to describe the interaction between material valance electrons and ions [12]. A plane wave cutoff of 80 Ry was used as a basis to expand electronic wave functions and integrations over the Brillouin zone were performed on a uniform kpoint density in all unit cells (45 points per reciprocal space volume). In face-centered cubic cells, this is a  $10 \times 10 \times 10$  mesh, which was shown to converge the forces. In this research, the training data, and subsequent potentials of Nickel and Chromium were developed by using an input pseudopotential from PseudoDojo. Initially, tests were conducted using a set of pseudopotentials that were pre-installed with the version of QE on the local supercomputer cluster, FERMI, however, this proved to be troublesome during some later calculations and ultimately, the team chose to move forward with different pseudopotentials. The pseudopotentials available from PsuedoDojo have the advantage of being more accurate, as well as typically achieving results at lower cutoff energy, thus, being more stable [12].

The development process for the training data and potential fitting took place in two steps: training data was generated on FERMI, and then results were analyzed either on FERMI as well, or on local machines. The process for generating training data used a series of Python scripts to automate creating files and queuing simulations. The training data process involved nine different codes, Fig. 2.1 outlines the process and combines similar steps for simplicity. In this figure, several acronyms are defined to aid in the description of the process for generating training data. These acronyms include QE (Quantum Espresso), SLURM (Simple Linux Utility for Resource Management), JSON (JavaScript Object Notation), and EOS (Equation of State). The process for developing training data involved seven steps, where in total, two sets of training data were generated. The first step involved setting up the QE calculations. In this script setup, the 15 possible different crystal lattice structure orientations were generated with the desired element in the corresponding atomic positions. Along with this, the script would set up the directories to allow for simulations to be made with each crystal lattice structure as an input for relaxation calculations. The next two codes in the process built the SLURM files to queue on FERMI, as well as submitted the files to the system to generate MD simulation data. The isolated atom input file was generated and relaxed as well to generate a reference energy level for comparison to the complete QE relaxation simulations generated before. From here,

the data followed two paths: the files were transformed into JSON format, one set was stored for later use in FitSNAP, and the other set was employed as input for the EOS calculations. The EOS calculations performed similar relaxation calculations, except for the new EOS input data. Once complete, these results would also be converted to JSON, and saved for use in FitSNAP for use with SNAP and ACE models.

**2.1. Training.** The process of creating the training data was initially designed with a roadmap in mind. This roadmap aimed to generate a sufficient amount of data to comprehensively cover the various types of simulations for which the potential could be utilized. The roadmap was the following five steps:

1. Bulk modulus and equation of state
2. Ab-Initio Molecular Dynamics
3. Mixing and alloying systems
4. Genetic algorithm structures
5. Defects

With the following five steps, nearly all were completed during the project timeline. One thing to note, despite being shown as a series of steps, it is not strictly required to perform simulations in the order shown above.

**2.1.1. Bulk Modulus and Equation of State.** The initial set of simulations focused on generating bulk modulus data. These simulations aimed to determine the relaxed energy of the system and examine how this energy varied with changes in volume. The models are also trained on structures that have been expanded and contracted so that they may capture the correct bulk modulus. The inputs for these simulations consisted of the initial 15 lattice orientation structures, which were generated by the QE setup code. These structures served as input files for the relation code. In Fig. 2.1 this set for generating the bulk modulus data is shown in the top row of boxes, where the first four steps in the process generate an output to be used in both FitSNAP, as well as input for generating the next set of simulation output.

This next step of simulation output, also shown in Fig. 2.1 is for generating the EOS structures. The purpose of the EOS simulations is to properly define the relationship between pressure, volume, and temperature for a system. With these simulations, for each bulk modulus simulation generated, seven equations of state simulations were also generated. These simulations are performed for the 14 separate Bravais lattices including the face-centered cubic (FCC), body-centered cubic (BCC), and simple cubic (SC) phases so that we may be able to look at phase transformations in the future. Similarly to before, with the bulk modulus equations, these results were converted for use in FitSNAP.

**2.1.2. Ab-Initio Molecular Dynamics.** We conducted Ab Initio Molecular Dynamics (AIMD) simulations to further explore atomic dynamics based on quantum mechanical properties. These simulations not only provide insights into atom dynamics but also generate electronic structure information.

The AIMD simulations were performed using Quantum Espresso (QE) and were initiated with the room-temperature stable crystal lattice orientation. For example, in the case of Nickel, the input structure was set to an FCC lattice structure. To create the initial structure, we used a Python script that leveraged the Atomic Simulation Environment (ASE) libraries and functions for defining the atomic system [6].

In the Nickel system, which has an FCC ground-state crystal lattice structure, the initial setup comprised only 4 atoms. To expand the dataset for simulations, we increased the system size by a factor of 2 in each direction, resulting in a system composed of 32 atoms. Additionally, AIMD simulations were carried out at two different temperature values: room temperature (around 300K) and a higher temperature, approximately 30% above the melting point of Nickel, which corresponds

to 2000 K.

In addition to these simulations, we conducted a set of five different cell parameter simulations. The 'CELL\_PARAMETERS' variable in Quantum Espresso controls the size of the cell in which the atoms are placed. To diversify the training data, we selected values that were both 10% and 25% above and below the default value. This resulted in two sets of ten AIMD simulations: five simulations with varying cell parameters at 300K and five simulations with varying cell parameters at 2000 K.

Similar simulations were replicated for the Chromium system, with one notable difference: the crystal structure in the Chromium system was BCC, comprising a total of 16 atoms, as opposed to the 32 atoms in the Nickel system.

For the AIMD simulations, we executed two distinct sets of simulations to generate training data, each utilizing different thermodynamic ensembles. In the first set, as mentioned in the preceding paragraph, simulations were carried out under conditions of constant particle number, volume, and temperature (NVT ensemble). In the second set of simulations, we maintained constant values for the number of particles, pressure, and temperature (NPT ensemble). These were performed using the default thermostat and barostat settings for QE simulations of Ni and Cr.

**2.1.3. Mixing and Alloying Systems.** Training data for alloy models typically encompasses atomic structures with diverse chemical compositions, often requiring a substantial amount of data even for single-chemical structures. In our current work, we have diligently generated substantial and highly accurate data for both Nickel (Ni) and Chromium (Cr) systems independently.

For the two essential components, Ni and Cr, we have produced special quasirandom structures (SQS) with varying concentrations of both elements, alongside symmetrically inequivalent structures (SIS). These structures, contained within relatively small DFT-sized cells, serve two primary purposes:

1. The SQS procedure yields structures that closely approximate the correlations found in a randomly mixed alloy of arbitrary size.
2. The SIS structures comprehensively cover the entire spectrum of chemical compositions between pure Ni and pure Cr, with adjustable granularity for sampling.

DFT calculations for these structures are currently in progress, providing valuable initial results for the pure Ni and Cr potentials. These results are available even before training a mixed alloy potential on the SQS and SIS structures.

**2.1.4. Genetic Algorithm Structures.** The final category of training data involves genetic algorithm structures, which serve a crucial purpose in generating a diverse range of randomized amorphous structures and perturbed crystalline structures. These structures offer a broad spectrum of structural characteristics that may not be typically sampled in conventional approaches to data generation. Additionally, they address the need to incorporate defects into the training dataset, with perturbed crystalline structures acting as proxies for explicit defect calculations.

To create these genetic algorithm structures, we utilized a modified version of the 'genetic algorithm search for bulk crystal structures' code found in the Atomic Simulation Environment (ASE) [6]. This adapted code allowed us to generate structures according to specific criteria and within predefined boundaries. The process of generating and employing these genetic algorithm structures involved the following five steps:

1. The initial step involved establishing a database of structures using the 'ga\_bulk\_start.py' code.
2. Executing the 'ga\_bulk\_run.py' code initiated the relaxation process, resulting in the generation of the structure set.
3. The generated structures were then converted from the .traj file format to the .cif file format using a Python script, preparing them for use as inputs in Quantum Espresso (QE).

4. Subsequent stages of the process involved the use of Python scripts to set up QE calculations, similar to those used in bulk modulus simulations.
5. The same Python scripts for creating SLURM files and managing job queues streamlined the execution of tasks.

The generated structures were then converted from the .traj file format to the .cif file format using a Python script, preparing them for use as inputs in Quantum Espresso (QE). While we have initiated defect calculations for FCC phases due to their relevance, these simulations were temporarily deferred to prioritize other roadmap steps. Defect states are significant for applications related to chemical segregation within grain boundaries, as demonstrated in Fig. 1.1. It's worth noting that defect simulations partially overlap with genetic algorithm structures. The goal of defect simulations is to create structures with specific types of defects, including point, line, and surface defects. These simulations often encompass crystal surfaces with low Miller indices, such as (100, 110, 111), which are considered in the context of genetic algorithm structures. Although explicitly including these calculations in the training data would enhance the potential's ability to accurately simulate defects, our current approach relies on extrapolations from a model primarily trained on genetic algorithms and bulk structures.

**2.2. Fitting potentials.** Linear models are trained on data from DFT, and atomic environment descriptors are used to encode information about the atomic systems. Atomic environment descriptors, in addition to their use in ML models, may also be used for characterizing atomic systems[4]. In this work, we focus on using spectral neighbor analysis (SNAP) descriptors or atomic cluster expansion (ACE) descriptors for linear models. These linear models, an array of the DFT data comprised of energies and forces  $\mathbf{E}$  of dimension  $S + 3N_s = \# \text{ of structures plus force rows}$ , is predicted using the product of the linear coefficients  $\mathbf{c}$  of dimension  $n_d = \# \text{ of descriptors}$  with the corresponding descriptors matrix of descriptors  $\mathbf{A}$ . The dimension of this training matrix is  $(S + 3N_s, n_d)$ . This matrix contains the average values of their descriptors as well as the gradients of them per atom. Using this, we train on both the energies and Hellman-Feynmann forces. To obtain the linear coefficients, a simple linear set of equations, Eq. (2.2), may be solved with singular value decomposition or similar methods.

$$\mathbf{E} = \mathbf{c}\mathbf{A} \quad (2.2)$$

Least squares alone tend to lead to significant overfitting and poor extrapolation of models, especially for short interatomic distances. Since we expect extreme atomic environments induced by E-fields in NiCr alloys, we implement new regularization schemes that penalize overfitting in a physically intuitive way. We add L2 regularization to the loss function:

$$Q = |\mathbf{E} - \mathbf{c}\mathbf{A}|^2 + R_{smooth}(\lambda, \mathbf{c}) \mathbf{c}^T \mathbf{c} + R_{black\_hole}(\lambda, \mathbf{c}) \mathbf{c}^T \mathbf{c} \quad (2.3)$$

where  $R_{smooth}$  is given by:

$$R_{smooth}(\lambda, \beta = \{c_{nl}\}) = \lambda \sum_{nl} c_{nl} \sum_I^N \left| \left( \frac{\partial B_{nl}}{\partial r} \right)_I - \left( \frac{\partial B_{nl}}{\partial r} \right)_{I+1} \right| \quad (2.4)$$

and  $R_{overlap}$  is given by:

$$R_{overlap}(\lambda', \beta = \{c_{nl}\}) = \lambda' \sum_{nl} c_{nl} \sum_I^{N_{short}} \max \left[ \left( \frac{\partial B_{nl}}{\partial r} \right)_I, 0 \right]. \quad (2.5)$$

In the context of ridge regression, two of these equations play crucial roles. Above, the first equation ensures that the dimer curve produced by the potential is smooth, and the second ensures that

spurious atomic overlap is penalized. Overlapping atoms is often the result of overfitting and poor extrapolation. These effects may be characterized by extreme values, discontinuities, or other irregularities in the data. These effects can disrupt the accuracy and reliability of the predictions or fit, and they are usually considered unwanted noise in the analysis in the context of ridge regression and modeling. We can modify our ridge regularization to penalize particularly larger coefficients that are contributing to the poor extrapolation through our addition of an L2 regularization term, as observed in Eq. (2.3).

A dimer is a compound formed by the union of two radicals or two molecules of a simpler compound. The dimer curve is a graph of potential energy as a function of bond length. The dimer model emerged as an initial attempt to describe the adsorption of diatomic molecules on crystal surfaces, effectively explaining the behavior of partially dissolved crystals in equilibrium. Over time, it has found applications in various physical systems and has gained considerable attention in the mathematical community[2]. Ridge regression, on its own, may not fully eliminate these instances of overlapping atoms. However, as highlighted in the results, the inclusion of these adaptive regularization penalties proves effective. A more comprehensive description of ridge regression can be found in the Fitting Methods section.

**2.3. Fitting Methods.** The open-source software package, FitSNAP, was used to train all models. This engine for training interatomic potentials uses the Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) as a backend for efficient evaluation of descriptors used in ML models. The models produced by FitSNAP are immediately compatible with LAMMPS to perform molecular dynamics simulations. FitSNAP can be utilized for training and testing a diverse array of atomistic simulation models and is not limited to the spectral neighbor analysis potential. The use of this software is ideal for this study because parameterization of both spectral Neighbor Potential (SNAP) and linear Atomic Cluster Environment (ACE) models are capabilities in FitSNAP[9].

**2.3.1. Spectral Neighbor Analysis Potential .** In this work, we harnessed linear SNAP and ACE models, implemented within FitSNAP. Our approach commenced with SNAP and subsequently incorporated ACE. Throughout the training process, FitSNAP fine-tuned model parameters to reduce discrepancies between predicted and actual properties for specific atomic configurations. These machine-learned interatomic potentials were initially trained using data on energy, stresses, and the corresponding forces, ensuring the reproduction of various properties tied to internal energy and force within reasonable bounds.

We opted for SNAP models because of their effectiveness and accuracy, particularly in metal systems, in contrast to ACE[5]. To set the stage, we employed several default parameter values outlined in LAMMPS software documentation, such as *rmin0* and *bzeroflag*, which we integrated into our system. Notably, these commands were computed as part of the ML-PACE package, exclusively enabled if it was built with LAMMPS [8].

Our approach encompassed the use of both SNAP and ACE methods, allowing us to explore various model and descriptor forms before selecting the most suitable one. The choice to employ linear ACE models also stemmed from the inherent limitations of traditional SNAP models, particularly the lack of control over bond distance sampling and the widely varying chemical interactions they encompass.

**2.3.2. Atomic Cluster Environment.** The explicit radial dependence of ACE is a valuable feature for addressing overlapping atoms and will be particularly useful for our future studies on alloy potentials. This is because ACE allows us to specify the nature of short- and long-ranged interactions for each bond type, such as Ni-Ni, Ni-Cr, and Ce-Si.

The ACE potential involves calculating energy contributions from atomic clusters, enabling a detailed and accurate representation of atomic interactions based on a quantum mechanics (QM) training dataset. To capture the energy landscape and interactions among atoms in a material, ACE

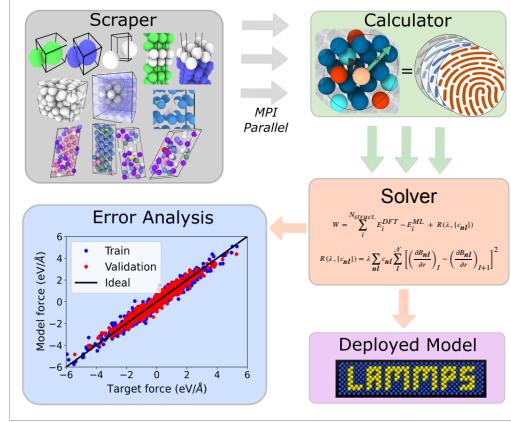


Figure 2.2: Fitting Process

relies on cluster expansion techniques. In contrast, SNAP is based on machine learning and spectral analysis methods for modeling atomic interactions. [5].

After fitting this model, we conducted an error analysis and deployed potential models within LAMMPS to assess their physical correctness and stability in dynamic simulations. Another metric used to assess stability is checking for overlapping atom attractions and correcting dissociation behavior. Models trained on energy and small amounts of force data enabled high-throughput optimization, focusing on achieving accurate extrapolations and interpolations for relevant atomic systems. Before implementing the data model, several parameters were fine-tuned to optimize its performance. These parameters include the radial cutoff ( $rcutfac$ ) and the parameter  $2jmax$ , which is instrumental in determining the number of bispectrum components integrated into the model's framework. The radial cutoff was determined through exhaustive Latin hypercube sampling, with the goal of minimizing force errors and ensuring the model's ability to reproduce physical behavior, particularly for ion dissociation. In this context, bispectrum components refer to measurement values that describe the arrangement of neighboring atoms in terms of both distance and angle [8].

For the case of nickel, we intentionally chose a  $2jmax$  value of 6, aligning with well-documented values in existing literature [10]. This choice strikes a balance between the predictive accuracy of the ACE model and the computational resources required for its implementation. While  $2jmax$  values can be adjusted beyond this threshold, doing so comes at the cost of increased computational resources required for MD simulations.

The radial cutoff factor, representing the distance at which interactions between atoms are limited, was set to 4.67637 Angstroms. A larger cutoff distance can improve accuracy by considering a broader interaction region, but it also leads to increased computational costs, while a smaller cutoff distance provides the opposite effect [8].

The initial parameter that underwent adjustment was the spherical harmonic terms number,  $lmax$ , which determines the highest order of spherical harmonic terms used in representing atomic environments. Different values of  $lmax$ , capture varying levels of angular detail. Higher values of  $lmax$ , capture more intricate angular patterns, enhancing accuracy but increasing computational complexity [1]. For our model, we initially used the order 0-5-3-1 and later adjusted it to 0-2-2-1, which corresponds to the  $lmax$  value for 2-body, 3-body, 4-body, and 5-body terms. Each number indicates the highest order of spherical harmonics used in the expansion for that particular case.

Subsequently, the number of radial terms,  $nmax$ , which determines the number of radial functions used to sample bond distances, was adjusted from an order of 12-7-3-1 to 6-3-3-1. Higher

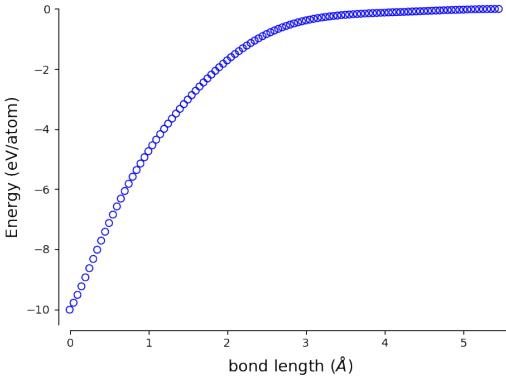


Figure 3.1: Overfitted Nickel Potential

$n_{max}$  values allow the model to capture more intricate variations in atomic interactions across different distance scales. However, excessively high  $n_{max}$  values can lead to overfitting, which is why we reduced it to a simpler model with fewer radial basis functions. This adjustment aims to enhance the accuracy of computed energies and align the results more closely with true or reference values.

Additionally, the remaining hyperparameters, including element weights and group weights, underwent meticulous calibration to optimize the overall performance of the models. We used genetic algorithms to optimize the weighting of the structures used in a weighted linear model fitting procedure. These hyperparameters have a smaller impact on the ML potential, but they were useful to make sure the correct phase (FCC Ni) was predicted by the potential under standard conditions. These hyperparameters collectively play a pivotal role in shaping the efficacy of the models and were subjected to comprehensive optimization efforts.

**3. Results.** The initial set of nickel potentials achieved a result for the root mean square that exceeded our initial search criteria. This outcome raised the possibility of overfitting, a phenomenon arising from the inadvertent encapsulation of stochastic noise and random perturbations, hindering the models' capacity to extrapolate effectively to previously unobserved data instances[7]. This can result in an inaccurate model, not being able to predict real-world scenarios. A possible remedy is through the addition of data, such as the mixing and alloying, or genetic algorithm structure systems that have yet to be utilized. This behavior can also be attributed to too many parameters being applied to our system, in our case it may be due to the system attempting to interpolate between data points.

For both SNAP and ACE, we employed a traditional RIDGE linear regression model to address overfitting and concurrently adjusted hyperparameters to minimize energy and force errors. This produced models that spuriously favored overlapping atoms. Favoring the overlapping atoms was generally less significant for ACE. For this reason, we then applied the new regularized regression scheme, by applying both Eqs. (2.4) and (2.5) when solving for the linear parameters of the ACE models. The result is a regularization scheme that penalizes both rapid changes in the derivative of the dimer curve as well as attractive forces at very short interatomic distances. Since the definition of these regularization terms depends on the solution, they must be solved iteratively. By tuning the strength of these regularization parameters, we were able to reduce spurious attractive forces for overlapping atoms and obtain a qualitatively correct dimer dissociation curve for Ni. These adjustments were confirmed to have a positive impact on our root mean square values when compared to the initial data we obtained. The outputted dimer plot reveals the stability of the chemical

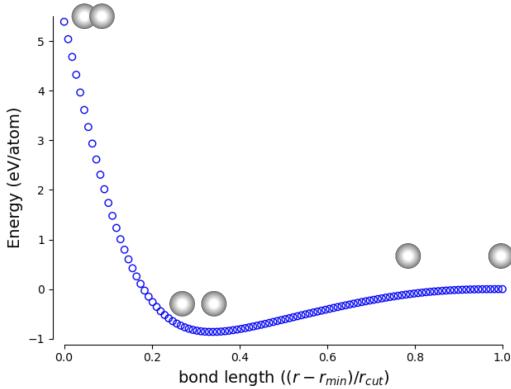


Figure 3.2: Optimized Potential Energy Curve

bonds at the different bond lengths and shows how the potential energy changes as the atoms get closer and farther apart. This physical behavior was only achieved after applying the new regularization function in Eq. (2.2). Prior to this, we observed overlap in atomic behavior, characterized by attractive forces at short bond distances that were spuriously high. We suspect that this poor behavior in Fig. 3.1 is due to overfitting and poor extrapolation. With our new approach, we have a better ability to extrapolate trends from our data. This issue, which significantly affects the development of machine-learned potentials, poses challenges in accurately capturing atomistic interactions. However, our findings indicate that employing certain regression schemes could potentially offer solutions to mitigate this problem. By selecting appropriate regression techniques and fine-tuning the model parameters, it might be possible to address the challenge of spuriously strong attractive forces and enhance the predictive capabilities of machine-learned interatomic potentials.

**4. Conclusions.** We have gathered a diverse training data set by incorporating bulk modulus, equation of state, AIMD, and genetic algorithm structure simulations. This data set allows us to model a wide range of simulation scenarios for both Nickel and Chromium. We have nearly completed the training data set for both of these materials. Although we sampled a full-range NiCr system, our focus has been on an 80% Ni and 20% Cr composition for experimental comparison. In the future, when Cerium (Ce) and Silicon (Si) are included, they will be present in trace amounts. Using this training data, potential files have been generated for Nickel, and atomistic simulations have been carried out to validate the accuracy of these potentials. These potentials have been verified to be stable for dynamics through LAMMPS simulations, which are displayed in Fig. 4.1. Although we haven't yet produced a Chromium potential within the allocated time frame, the next steps in the project include using the generated training data for Chromium to obtain a potential.

Considering the current progress, several key tasks are necessary to achieve the project's ultimate goal. First, we need to complete the Nickel and Chromium training data, including the required genetic algorithm simulations for Chromium and defect simulations for both materials. Once this is done, we will have covered the two extremes of the final system: Nickel and Chromium.

The next significant task involves creating an alloy system. This entails generating numerous SQS (Special Quasirandom Structures) and SIS (Substitutional Solid Solution) systems at various concentrations of Nickel and Chromium, as previously discussed. Subsequently, the results from these training datasets will be combined into a single potential, and simulations will be conducted to validate the NiCr potential. Additional training data or optimization may be necessary.

The next major milestone involves incorporating Silicon and Cerium into the model or system,

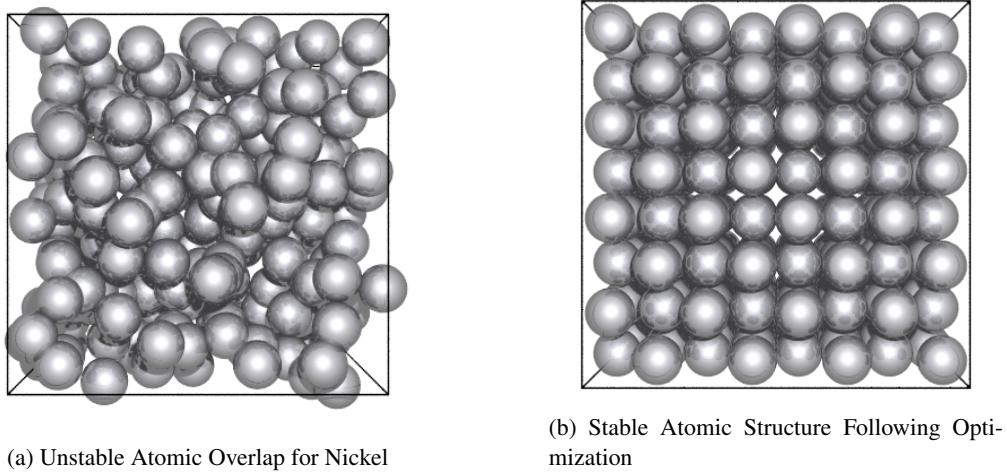


Figure 4.1: MD Simulations of Nickel

based on the training data, and optimizing it to create a NiCrSiCe potential. The final step will encompass conducting simulations of various systems with compositions similar to those observed in TEM images and testing for Cerium precipitation. Through these simulations, our goal is to gain a deeper understanding of the dynamics behind grain boundary segregation observed in our nichrome alloy when subjected to nano-current electrical pulses.

## REFERENCES

- [1] J. M. CHRISTIANSEN-SALAMEH, *MACHINE-LEARNING INTERATOMIC POTENTIAL FOR ULTRA-WIDE BANDGAP SEMICONDUCTOR AND X-RAY SCATTERING STUDY OF CHARGE DENSITY WAVE*, PhD thesis, Cornell University, 2002.
- [2] M. GARTNER, *Classical and quantum dimer models*, (2014).
- [3] P. GIANNOZZI, *Quantum espresso: a modular and open-source software project for quantum simulations of materials*, Journal of Physics: Condensed Matter, 21 (2009).
- [4] R. GUHA AND E. WILLIGHAGEN, *A survey of quantitative descriptions of molecular structure*, Current topics in medicinal chemistry, 12 (2012), pp. 1946–1956.
- [5] D. P. KOVACS, C. V. D. OORD, J. KUCERA, A. E. A. 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, 17 (2021), pp. 7696–7711. PMID: 34735161.
- [6] A. H. LARSEN, J. J. MORTENSEN, J. BLOMQVIST, I. E. CASTELLI, R. CHRISTENSEN, M. DUŁAK, J. FRIIS, M. N. GROVES, B. HAMMER, C. HARGUS, ET AL., *The atomic simulation environment—“a python library for working with atoms*, Journal of Physics: Condensed Matter, 29 (2017), p. 273002.
- [7] S. MUTASA, S. SUN, AND R. HA, *Understanding artificial intelligence based radiology studies: What is overfitting?*, Clinical imaging, 65 (2020), pp. 96–99.
- [8] S. PLIMPTON, A. THOMPSON, S. MOORE, ET AL., *LAMMPS Documentation (2 Aug 2023 version) – LAMMPS documentation*, 2023.
- [9] C. SIEVERS, M. CUSENTINO, D. M. O. DE ZAPIAIN, ET AL., *Fitsnap: Atomistic machine learning with lammps*, Journal of Open Source Software, 8 (2023), p. 5118.
- [10] E. SIKORSKI, M. CUSENTINO, M. McCARTHY, J. TRANCHIDA, M. WOOD, AND A. THOMPSON, *Machine learned interatomic potential for dispersion strengthened plasma facing components*, The Journal of Chemical Physics, 158 (2023).
- [11] E. G. STEFAN KURTH, M.A.L. MARQUES, *Density-functional theory*, Encyclopedia of Condensed Matter Physics, (2005), pp. 395–402.
- [12] M. VAN SETTEN, *The pseudodojo: Training and grading an 85 element optimized norm-conserving pseudopotential*

*table*, Computer Physics Communications, 226 (2018), pp. 39–54.

## A DEEP LEAST-SQUARES METHOD FOR THE STOKES EQUATIONS

TEDDY MEISSNER\*, EDWARD HUYNH†, PAUL KUBERRY‡, AND PAVEL BOCHEV §

**Abstract.** Physics Informed Neural Networks (PINNs) seek approximate solutions to PDEs as neural network functions minimizing the PDE residual over a set of collocation points. Thus, PINNs are an instance of a mesh free collocation least-squares method. The ease of data integration, scalability to higher dimensional problems, and flexibility to adapt the same network architecture to many equations has made PINNs an attractive option. Yet, when compared to traditional least-squares methods, PINNs tend to struggle in both computational time and accuracy. As a residual based method, one of the primary problems in these types of solvers is the choice of weights in the loss function. Many current PINNs use mean squared error (MSE) for each term and aim to find the correct weighting coefficients based on ad-hoc methods or experiments. In this work we use elliptic regularity theory to define a norm-equivalent loss function that involves combinations of Sobolev space norms reflecting the ellipticity properties of the differential operator. We term this approach deep least-squares because it relies on the same mathematical basis as conventional least-squares methods. To illustrate the approach we formulate a deep least-squares method for the incompressible Stokes equations. Numerical results confirm that the choice of a norm-equivalent loss function greatly improves the accuracy of PINNs.

**1. Introduction.** Physics Informed Neural Networks is a numerical method that seeks an approximate solution to a Partial Differential Equation (PDE) problem by minimizing its residual over a finite dimensional function space comprising neural networks with a prescribed architecture, width and depth. As such, PINNs belong to the class of least-squares numerical methods for PDEs. Because minimization is usually performed over a set of collocation points, PINNs are an instance of a mesh-free collocation least-squares method. However, in contrast to traditional residual minimization methods for PDEs, PINNs employ a globally defined trial space whose regularity is determined by the regularity of the activation function employed by the neural network. As a result, PINNs rely on automatic differentiation to compute the PDE residual at the collocation points. Furthermore, for smooth activation functions one does not have to first transform the PDE into an equivalent first-order system, which is a common practice in, e.g., least-squares finite elements [4].

Some of the early work on using neural networks to solve differential equations dates back at least two decades [8, 11] but its broader adoption suffered from the lack of sufficient computational resources to efficiently train the networks. Owing to the advances in computational power and automatic differentiation, there has been increased attention on this subject since the revival of such methods in [12], where the term PINNs was coined. While the concept of a PINN can be used in both the forward simulation and data assimilation contexts, in this paper we focus on the former, i.e., we view PINN as a counterpart to traditional numerical methods for PDEs such as finite element and finite difference methods.

PINNs have many appealing features such as a relatively small memory footprint, compared to traditional numerical methods, ability to avoid the curse of dimensionality [10, 5], and an inherent mesh-free nature. Yet, when it comes to a comparison between traditional methods such as FEM and FD, PINNs still tend to under perform in both accuracy and computational time [9]. Some of the reasons for this include the abundant choice of hyperparameters for optimization procedures and architectural design, along with the lack of theory for defining the optimal loss function and accounting for boundary conditions [6].

In this work we focus on the latter question and draw upon mathematical theories

---

\*University of Arizona, tmeissner@arizona.edu

†University of Arizona, huynh3@math.arizona.edu

‡Sandia National Labs, pakuber@sandia.gov

§Sandia National Labs, pbboche@sandia.gov

supporting the formulation of conventional optimally accurate residual minimization-based methods, such as least-squares finite elements, to guide the development of loss functions. To that end, we consider the incompressible Stokes equations in two dimensions written as an equivalent first-order Velocity-Vorticity-Pressure (VVP) system, augmented with the velocity boundary condition. We then use the Agmon-Douglis-Nirenberg (ADN) elliptic regularity theory [1] to formulate a loss function that is consistent with the ellipticity properties of the resulting boundary value problem (BVP). We term this approach deep least-squares method because it relies on the same mathematical basis as conventional least-squares methods to define norm-equivalent loss functions. We compare numerically the deep least-squares with PINN formulations employing standard MSE loss functions and show that the new approach leads to significant improvements in the accuracy of the solution.

Our results provide a more rigorous basis for recent empirical findings suggesting to include higher order derivatives in the definition of the loss. In particular, [7] shows that minimizing target derivatives in addition to target values can increase model accuracy, while [13] finds that training PINNs in higher order Sobolev spaces can lead to more accurate models with faster convergence.

**2. The Stokes equations.** Let  $\Omega \subset \mathbb{R}^2$ , denote a bounded open region with boundary  $\Gamma$ . The steady incompressible flow of a Newtonian fluid in  $\Omega$  can be modeled by the Stokes equations

$$\begin{aligned} -\nu \Delta \mathbf{u} + \nabla p &= f \text{ on } \Omega \\ \nabla \cdot \mathbf{u} &= 0 \text{ on } \Omega \\ \mathbf{u} &= g \text{ on } \Gamma \end{aligned} \tag{2.1}$$

where  $\mathbf{u}$  and  $p$  are the velocity and the pressure fields, respectively,  $\nu$  is the viscosity,  $f$  is the body force, and  $g$  is a function prescribing the velocity field on the boundary of the domain.

We recall that in two dimensions there are two curl operators acting on vector fields  $\mathbf{u} = (u, v)$  and scalar functions  $\phi$ , defined as

$$\nabla \times \mathbf{u} = v_x - u_y \quad \text{and} \quad \nabla \times \phi = \begin{pmatrix} \phi_y \\ -\phi_x \end{pmatrix}.$$

respectively. Using the incompressibility constraint  $\nabla \cdot \mathbf{u} = 0$  along with the vector identity

$$\nabla \times \nabla \times \mathbf{u} = -\Delta \mathbf{u} + \nabla(\nabla \cdot \mathbf{u})$$

it is easy to see that (2.1) is equivalent to the following first-order velocity-vorticity-pressure (VVP) system

$$\begin{aligned} \nabla \times \mathbf{u} + \nabla p &= 0 \text{ on } \Omega \\ \nabla \times \mathbf{u} - \omega &= 0 \text{ on } \Omega \\ \nabla \cdot \mathbf{u} &= 0 \text{ on } \Omega \\ \mathbf{u} &= \mathbf{g} \text{ on } \Gamma \end{aligned} \tag{2.2}$$

where  $\omega = \nabla \times \mathbf{u}$  is the vorticity field.

**2.1. A deep least squares method for the VVP Stokes equations.** To discuss residual minimization methods for (2.2) we need some additional notation. We recall the space  $L^2(\Omega)$  of all square integrable functions on  $\Omega$  with norm and inner product denoted by  $\|\cdot\|_0$  and  $(\cdot, \cdot)_0$ , respectively. The symbol  $L_0^2(\Omega)$  will stand for the subspace of all zero

mean functions in  $L^2(\Omega)$ . As usual,  $H^k(\Omega)$  will denote the Sobolev space of order  $k$  on  $\Omega$  and  $H^{k-1/2}(\Gamma)$  will be the associated trace space. We denote the norms and the inner products on these spaces by  $\|\cdot\|_k$ ,  $(\cdot, \cdot)_k$ ,  $\|\cdot\|_{k-1/2,\Gamma}$ , and  $(\cdot, \cdot)_{k-1/2,\Gamma}$  respectively.

A formal prerequisite for a well-posed, optimally convergent residual minimization formulation for PDEs is the norm-equivalence of the loss functional. This property implies  $V$ -ellipticity (strong coercivity) of the resulting first-order optimality condition (Euler-Lagrange equation) and existence of a unique minimizer. A norm-equivalent loss function for (2.2) can be formulated by using the ADN elliptic regularity theory [1]. Let  $q \geq 0$  be an integer regularity index and assume that  $\mathbf{u} \in [H^{q+2}(\Omega)]^2$ ,  $\omega \in H^q(\Omega)$  and  $p \in H^q(\Omega) \cap L_0^2(\Omega)$ . The ADN theory then asserts the existence of a constant  $C$  such that

$$\begin{aligned} \|\mathbf{u}\|_{q+2} + \|\omega\|_{q+1} + \|p\|_{q+1} &\leq C \left( \|\nabla \times \omega + \nabla p - f\|_q \right. \\ &\quad \left. + \|\nabla \times \mathbf{u} - \omega\|_{q+1} + \|\nabla \cdot \mathbf{u}\|_{q+1} + \|\mathbf{u}\|_{q+3/2,\Gamma} \right). \end{aligned} \quad (2.3)$$

The significance of (2.3) for residual minimization methods is that the upper bound in this inequality defines the combination of norms for the residuals of the equations and the boundary conditions that would yield a norm-equivalent loss function for (2.2). Specifically, setting  $q = 0$ ,  $\mathcal{V} = [H^2(\Omega)]^2 \times H^1(\Omega) \times H^1(\Omega)$ , and  $\mathcal{V} \ni U = (\mathbf{u}, \omega, p)$  one can show that

$$\mathcal{J}(U; f, \mathbf{g}) = \frac{1}{2} \left( \|\nabla \times \omega + \nabla p - f\|_0^2 + \|\nabla \times \mathbf{u} - \omega\|_1^2 + \|\nabla \cdot \mathbf{u}\|_1^2 + \|\mathbf{u} - \mathbf{g}\|_{3/2,\Gamma}^2 \right) \quad (2.4)$$

is a norm-equivalent loss function for the VVP system, i.e., there holds

$$C_1 (\|\mathbf{u}\|_2 + \|\omega\|_1 + \|p\|_1) \leq \mathcal{J}^{1/2}(\mathbf{u}, \omega, p; 0, \mathbf{0}) \leq C_2 (\|\mathbf{u}\|_2 + \|\omega\|_1 + \|p\|_1), \quad (2.5)$$

with some positive constants  $C_1$  and  $C_2$ ; see, e.g., [3]. It is easy to see that property (2.5) implies that the Euler-Lagrange equation for (2.4), defined by the bilinear form

$$\begin{aligned} \mathcal{B}(U, V) &= (\nabla \times w + \nabla p, \nabla \times \psi + \nabla q)_0 \\ &\quad + (\nabla \times \mathbf{u} - \omega, \nabla \times \mathbf{u} - \omega)_1 + (\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v})_1 + (\mathbf{u}, \mathbf{v})_{3/2,\Gamma}, \end{aligned} \quad (2.6)$$

where  $V = (\mathbf{v}, \psi, q) \in \mathcal{V}$  is coercive on  $\mathcal{V} \times \mathcal{V}$ .

In contrast, virtually all PINN formulations reported in the literature are based on minimization of empirically weighted MSE loss functions. The latter can be thought of as discrete equivalents of the loss function

$$\mathcal{J}_\alpha(U; f, \mathbf{g}) = \frac{1}{2} \left( \|\nabla \times \omega + \nabla p - f\|_0^2 + \alpha_1 \|\nabla \times \mathbf{u} - \omega\|_0^2 + \alpha_2 \|\nabla \cdot \mathbf{u}\|_0^2 + \alpha_3 \|\mathbf{u} - \mathbf{g}\|_{0,\Gamma}^2 \right) \quad (2.7)$$

in which all residuals, including the boundary residual, are measured in  $L^2$  norms scaled by empirically determined weights  $\alpha_i$ . Unlike (2.4), the functional (2.7) is not norm-equivalent, i.e., it gives rise to a bilinear form that is not strongly coercive.

While there have been many proposed weighting strategies to determine the weights  $\alpha_i$ , one can clearly see that in order for (2.7) to be norm-equivalent, the weights  $\alpha_i$  must be such that

$$\alpha_i \|\mathcal{N}(\theta)\|_0^2 \sim \|\mathcal{N}(\theta)\|_1^2 \quad \text{for } i = 1, 2 \quad \text{and} \quad \alpha_3 \|\mathcal{N}(\theta)\|_{0,\Gamma}^2 \sim \|\mathcal{N}(\theta)\|_{3/2,\Gamma}^2$$

where  $\mathcal{N}(\theta)$  is a neural network function with parameters  $\theta \in \mathbb{R}^N$ . These relations suggest that one may have to consider an adaptive weighting scheme, where  $\alpha_i$  depend on the neural network parameters  $\theta$ . This may intuitively explain some of the difficulty with finding the optimal weights in current techniques.

**3. A deep least-squares method for the Stokes equations.** While the norm-equivalent loss function (2.4) is compliant with the ellipticity properties of the underlying PDE, it cannot be used directly in conventional least-squares finite element methods because the associated bilinear form (2.6) requires test and trial functions that are at least  $C^1$ -regular. However, with the exception of Cartesian grids, construction of such functions is difficult and often impractical. In least-squares finite element methods this issue is dealt with by using properly weighted  $L^2$  norms as a substitute for the mathematically correct higher order Sobolev norms, which enables implementation using standard  $C^0$ -regular finite element spaces.

Neural networks present an opportunity to formulate a residual minimization-based method for (2.2) that uses directly the norm-equivalent loss function (2.4) instead of a weighted MSE version of that loss function. Indeed, since the regularity of a neural network is the same as the regularity of its activation function, any network employing a  $C^1$  or higher regular activation, such as  $\tanh$  is admissible as an argument in (2.4).

To define such a method for the Stokes equations (2.2), let  $\mathcal{V}(\mathcal{A}, \sigma, \Theta)$  denote a space of neural network functions

$$U_{\mathcal{N}} : \Omega \times \Theta \mapsto \mathbf{R}^4; \quad U_{\mathcal{N}} = (\mathbf{u}_{\mathcal{N}}, \omega_{\mathcal{N}}, p_{\mathcal{N}}), \quad (3.1)$$

with architecture  $\mathcal{A}$ , activation  $\sigma \in C^k(\Omega)$ ,  $k \geq 1$ , and parameter set  $\Theta \subset \mathbb{R}^N$ . Thanks to the regularity assumption on  $\sigma$  it follows that  $\mathcal{V}(\mathcal{A}, \sigma, \Theta)$  is a conforming approximation of  $\mathcal{V}$ , i.e.,  $\mathcal{V}(\mathcal{A}, \sigma, \Theta) \subset \mathcal{V}$ .

For notational simplicity in what follows we will drop the subscript  $\mathcal{N}$  when referring to elements of  $\mathcal{V}(\mathcal{A}, \sigma, \Theta)$ . The deep least-squares method for (2.2) is then given by the following minimization problem: seek  $U \in \mathcal{V}(\mathcal{A}, \sigma, \Theta)$  such that

$$\mathcal{J}(U; f, \mathbf{g}) \leq \mathcal{J}(V; f, \mathbf{g}) \quad \forall V \in \mathcal{V}(\mathcal{A}, \sigma, \Theta). \quad (3.2)$$

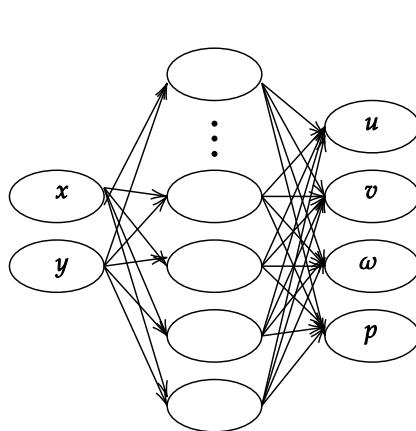
Since an element of  $\mathcal{V}(\mathcal{A}, \sigma, \Theta)$  is uniquely determined by a parameter set  $\theta \in \Theta$ , problem (3.2) is equivalent to

$$\theta = \arg \min_{\psi \in \Theta} \mathcal{J}(U(\psi); f, \mathbf{g}). \quad (3.3)$$

**3.1. Implementation of a deep least-squares method for the VVP Stokes system.** Implementation of (3.2) or (3.3) requires selection of a neural network architecture and numerical approximation of the loss function (2.4). In this work we examine two different neural network architectures. Both use the  $\tanh$  activation and define functions (3.1) mapping two dimensional spatial coordinates  $(x, y)$  to a four dimensional output  $(u, v, \omega, p)$ . The first architecture is a standard feedforward fully connected network in which all outputs share the weights of hidden layers. The second architecture comprises four separate feedforward networks, one for each variable. Figure 3.1 illustrates the two network architectures.

The motivation for the second architecture stems from ADN theory, which treats the solution of the PDE as a set of independent scalar coordinate functions belonging in different spaces. The functions therefore should only be connected through the governing equations of the PDE. While it may be possible to embed four separate neural networks into a single network, it would be difficult to impose no communication between the outputs and result in connected solution spaces which should otherwise be unconnected.

Single neural network



Four Separate neural networks

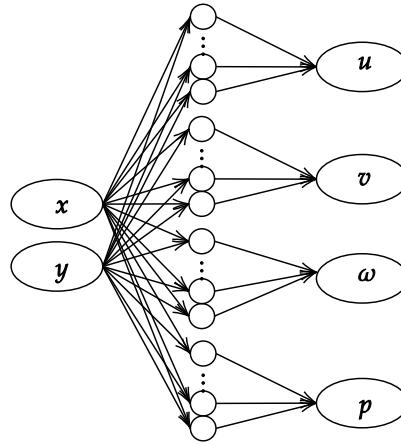


FIG. 3.1. Neural network architectures used in the deep least-squares method. Left: conventional fully connected architecture. Right: ADN-compliant architecture.

**3.1.1. Numerical Loss Function.** Solution of the deep least-squares optimization problem (3.3) requires evaluation of the loss function (2.4) for any given parameter set  $\theta \in \Theta$ . Except for some trivial cases the integrals in (2.4) cannot be computed in closed form. Thus, we approximate (2.4) using numerical quadrature. To that end we shall use Riemann sums over a point cloud

$$X(\Omega) := \{\mathbf{x}_i\}_{i=1}^{N_c} \in \Omega,$$

comprising  $N_c$  uniformly distributed points in  $\Omega$ , to approximate Sobolev norms on  $\Omega$ . To approximate trace norms on  $\Gamma$  we use another point cloud  $X(\Gamma)$  comprising  $N_b$  points on  $\Gamma$ . This approach circumvents the need to define a mesh, which would somewhat diminish the advantage of using mesh-free trial spaces in the method.

Given a smooth scalar function  $\phi(\mathbf{x})$  we approximate its  $L^2(\Omega)$  and  $H^1(\Omega)$  norms as:

$$\|\phi\|_0^2 \approx \|\phi\|_{0,X(\Omega)}^2 := \frac{1}{N_c} \sum_{i=1}^{N_c} \phi^2(\mathbf{x}_i) \quad \text{and} \quad \|\phi\|_1^2 \approx \|\phi\|_{1,X(\Omega)}^2 := \frac{1}{N_c} \sum_{i=1}^{N_c} \phi^2(\mathbf{x}_i) + |\nabla \phi(\mathbf{x}_i)|^2 \quad (3.4)$$

respectively. To compute (2.4) we also need to evaluate fractional order norms on the boundary. To that end we compute the norm on  $H^{k+1/2}(\Gamma)$  by interpolation between the Sobolev spaces  $H^k(\Gamma)$  and  $H^{k+1}(\Gamma)$ . We approximate the norms on these spaces by using their tangential derivatives on  $\Gamma$  as

$$\|\phi\|_{q,\Gamma}^2 \approx \|\phi\|_{k,X(\Gamma)}^2 := \frac{1}{N_b} \sum_{|\alpha| \leq q} \sum_{i=1}^{N_b} |D^\alpha \phi(\mathbf{x}_i) \cdot \mathbf{t}|^2; \quad q = k, k+1$$

where  $\mathbf{t}$  is the unit tangent vector to  $\Gamma$ . We then set

$$\|\phi\|_{k+\frac{1}{2},\Gamma} \approx \|\phi\|_{k,X(\Gamma)}^\beta \|\phi\|_{k+1,X(\Gamma)}^{1-\beta}$$

where the default value  $\beta = 0.5$  may be tuned additionally.

Using the numerical approximations of the Sobolev norms on  $\Omega$  and  $\Gamma$  we define the following family of discrete loss functions

$$\begin{aligned}\mathcal{J}_{\mathbf{a}}(U; f, \mathbf{g}) = & \frac{1}{2} \left( \|\nabla \times \omega + \nabla p - f\|_{a_1, X(\Omega)}^2 \right. \\ & \left. + \|\nabla \times \mathbf{u} - \omega\|_{a_2, X(\Omega)}^2 + \|\nabla \cdot \mathbf{u}\|_{a_3, X(\Omega)}^2 + \|\mathbf{u} - \mathbf{g}\|_{a_4, X(\Gamma)}^2 \right)\end{aligned}\quad (3.5)$$

where  $\mathbf{a} = (a_1, a_2, a_3, a_4)$  are Sobolev norm indices for the residuals of (2.2). The choice  $\mathbf{a}_1 = (0, 1, 1, 3/2)$  corresponds to the norm-equivalent loss function (2.4) that is the basis for the deep least-squares method. To examine the importance of using norms that are consistent with the ellipticity properties of the underlying PDE, we shall compare the performance of the deep least-squares method with three other methods employing loss functions (3.5) corresponding to the choices  $\mathbf{a}_2 = (0, 0, 0, 0)$ ,  $\mathbf{a}_3 = (1, 1, 1, 1)$ , and  $\mathbf{a}_4 = (0, 1, 1, 1)$ , respectively.

The second choice corresponds to a standard PINN employing an MSE loss function, the third choice can be viewed as a PINN based on Sobolev training [7, 13] in  $H^1(\Omega)$  spaces, and the last choice represents an approximation of  $\mathcal{J}_{\mathbf{a}_1}$  in which the fractional trace norm  $H^{3/2}(\Gamma)$  has been replaced by a weaker integer trace norm. The PINN case will serve as a baseline, whereas the other two cases will attempt to determine if choosing loss functions close to that of the theory can yield some improvements over the baseline case.

**4. Results and Discussion.** In this section we report results from a comparison study between four residual minimization methods for the incompressible Stokes equation corresponding to the four instantiations of (3.5) described above. The study uses the manufactured solution

$$\begin{aligned}u(x, y) &= \sin(\pi x) \cos(\pi y) \\ v(x, y) &= -\cos(\pi x) \sin(\pi y) \\ p(x, y) &= y(1 - y) \sin(\pi x)\end{aligned}$$

defined in [2]. Figure 4.1 shows the contour plots of each solution component.

To compare performance of the four different scheme we use the relative  $L^2$  error of the numerical solution defined as

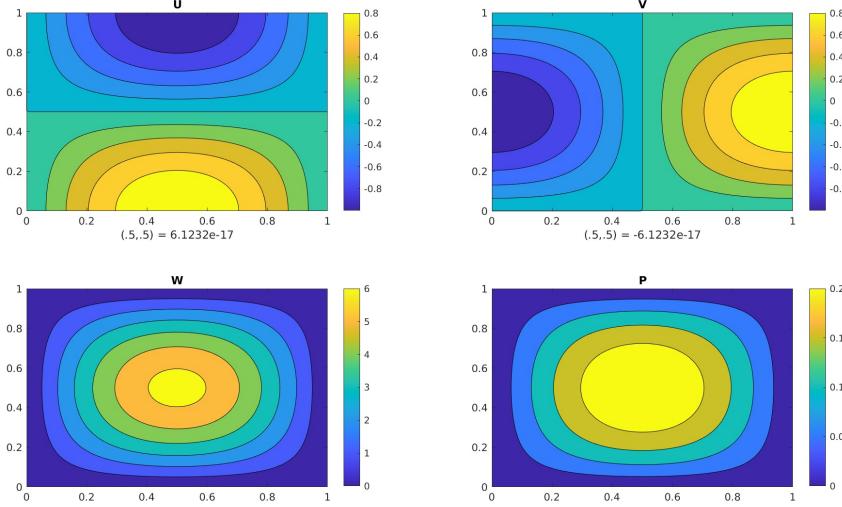
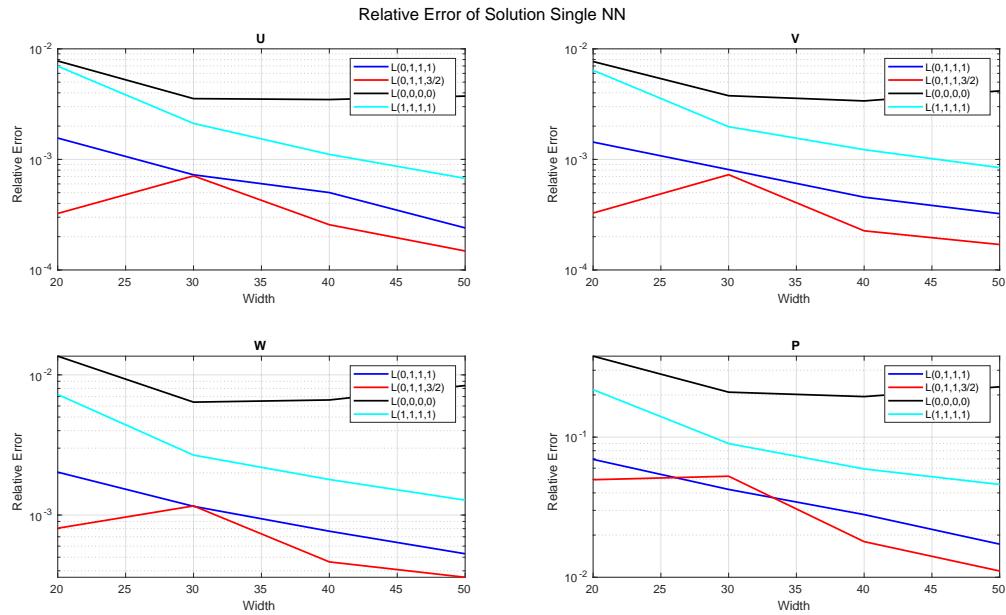
$$U_{error} = \frac{\|U - U_*\|_0}{\|U_*\|_0}$$

where  $U = (\mathbf{u}, \omega, p)$  and  $(\cdot)_*$  indicates the true solution. Because the residuals of the differential equations provide a measure of how well the numerical solution satisfies physical properties such as conservation of momentum and mass, we also report the MSE for each one of these terms. For example, the value of

$$\text{MSE}(\nabla \times \omega + \nabla p - f) = \|\nabla \times \omega + \nabla p - f\|_{0, X(\Omega)}$$

can be used to assess conservation of linear momentum by each one of the four schemes under investigation.

Figures 4.2-4.6 are all generated with 1 hidden layer and widths 20,30,40,50. Each combination of loss function and width was run for 10 samples while the lines represent the average error. All results were initially run with a 2000 Epochs using Adam optimizer with an equal learning rate and decay rate of .01. The training was then finished with 3000 iterations using limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). The batch

FIG. 4.1. *Contours of manufactured solution used in comparison plots*FIG. 4.2. *Average relative error over 10 samples of the velocity ( $u, v$ ), vorticity  $\omega$ , pressure,  $p$  using a single fully connected neural network. All plots are generated with 1 hidden layer and increasing width.*

sized used for Adam was 250 while  $N_c = 2500$  and  $N_B = 200$ . More discussion into the choice of hyperparameters can be found in Appendix A.

In Figure 4.2 we show a single fully connected neural network across the four loss functions described. In nearly every portion of the solution  $(u, v, \omega, p)$ , we see the loss function corresponding to  $\mathbf{a} = (0, 1, 1, 3/2)$  results in the lowest relative error while the

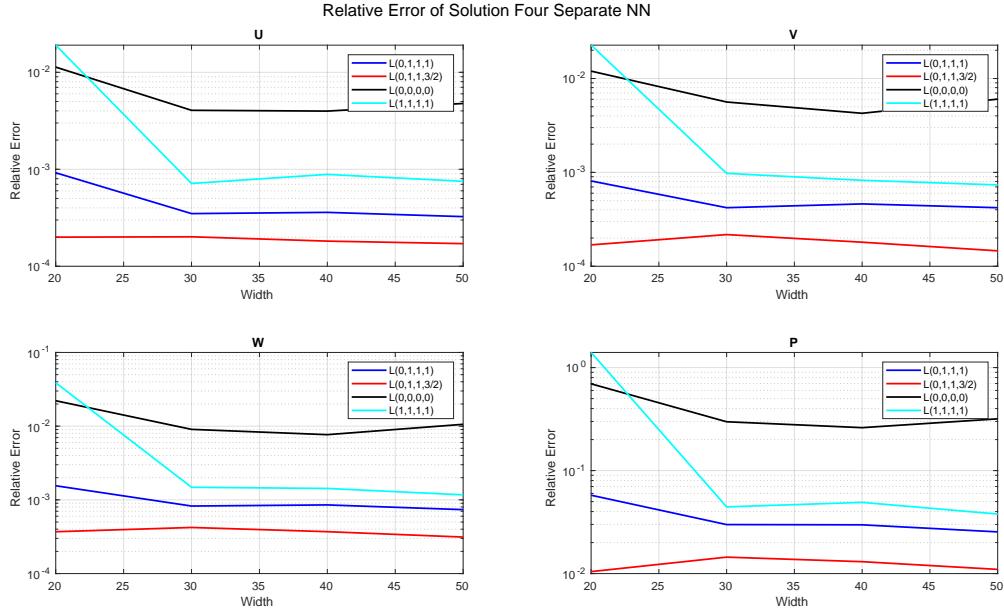


FIG. 4.3. Average relative error over 10 samples of the velocity ( $u, v$ ), vorticity  $\omega$ , pressure,  $p$  using four separate fully connected neural networks. All plots are generated with 1 hidden layer and increasing width.

baseline mean-squared error performs the worse and the error tends to stabilize after a width of 30. It is important to note that the initialization of the PINN can play a large role in the optimization and may be a component of the artifacts we see. Nonetheless, the results show a clear distinction between the error between each loss function.

In Figure 4.3 we show four separate fully connected neural networks tied to the four loss functions described. Similarly to Figure 4.2, we see across all solutions the loss function corresponding to  $\mathbf{a} = (0, 1, 1, 3/2)$  performs the best while the standard mean squared error performs the worse.

In Figures 4.4,4.5 we show the magnitude of the MSE for each term in the loss function for a single and four separate fully connected networks. Here we can clearly see the error in the solution is not necessarily connected to the smallest error in the loss function. For example,  $\mathbf{a} = (0, 1, 1, 1)$  outperforms  $\mathbf{a} = (0, 1, 1, 3/2)$  in three of the four terms, yet fails to result in lower solution error. This shows the power of the elliptic regularity theory where the relative importance of each residual term is accounted for correctly.

Lastly, Figure 4.6 shows the standard deviation of the four loss functions and two networks on the 10 samples used. We again see similar trends where the norm-equivalent loss function based on the ADN theory,  $\mathbf{a} = (0, 1, 1, 3/2)$ , results in the minimum standard deviation across all widths. This suggests that of all four methods considered, the deep least-squares not only achieves the best solution error, but does this in the most consistent manner.

The results are consistent with the findings in [7, 13], which suggest that training PINNs in higher order Sobolev spaces can result in more accurate results with faster convergence. The main difference is that while adding target derivatives to the loss function decreases the error, the correct combination of derivatives plays an important role.

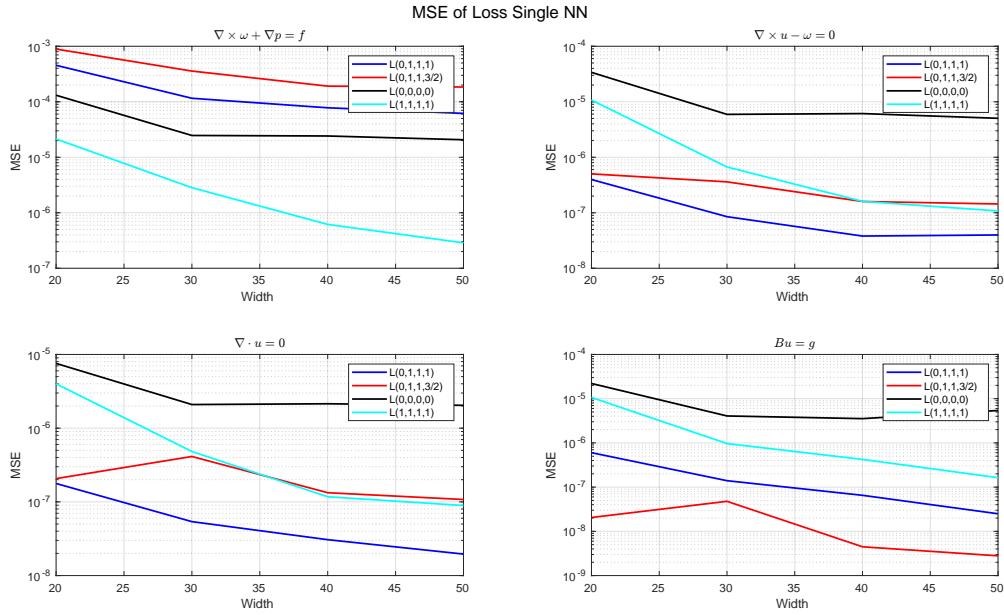


FIG. 4.4. Average MSE over 10 samples for each term in the VVP equations using a single fully connected neural networks. All plots are generated with 1 hidden layer and increasing width.

**5. Summary.** While PINNs have shown promise in their ability to generate mesh free approximations of PDEs, there are several significant hurdles. These include, but are not limited to, the abundant choice of hyper-parameters for optimization procedures and architectural design, lack of theory for defining the optimal loss function and accounting for boundary conditions [6]. In this work we addressed choosing a natural loss function that is consistent with the ellipticity properties of the underlying PDE. Our results show that by using the ADN elliptic regularity theory one can determine a combination of Sobolev norms for the loss function terms that greatly improves the accuracy and consistency of the resulting deep least-squares method.

**Appendix A. HyperParameters.** Quite often in neural networks, the manually tuning of hyperparameters can be the most time consuming component for the user. Motivated by time savings of previous papers reportings, below are our findings.

**A.1. Optimization methods.** The best combination of optimizers tended to be an initial set amount of epochs of Adam and then finishing the optimization with the L-BFGS, a quasi-Newton algorithm. This gives the initial flexibility of a stochastic optimizer for high dimensional optimization and then the power of quasi-Newton methods for convergence. Overall, for our work, we found around 2000 epochs of Adam and 3000 iterations of L-BFGS was sufficient. In some cases the loss would decrease after this point but the amount was minor.

**A.2. Activation Functions.** We tested tanh, swish, Relu and sine activation functions. Overall we found tanh, swish, and sine to be comparable, while ReLU did not perform well. This is consistent with the fact that of all activation functions considered here ReLU is the only one that is not  $C^1$ -regular, as required for the computation of the norm-equivalent loss function (2.4).

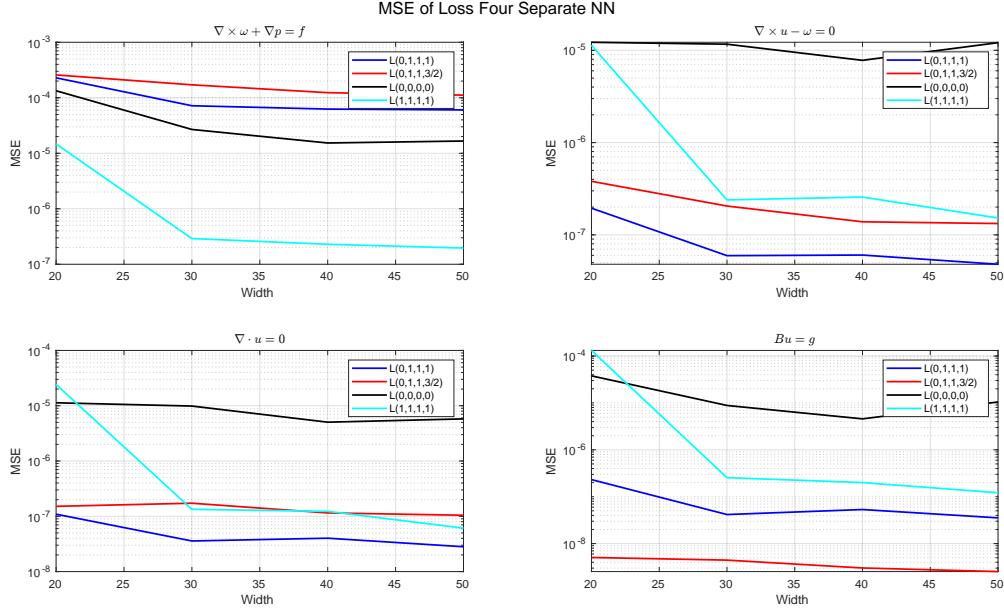


FIG. 4.5. Average MSE over 10 samples of each term in the VVP equations using four separate fully connected neural networks. All plots are generated with 1 hidden layer and increasing width.

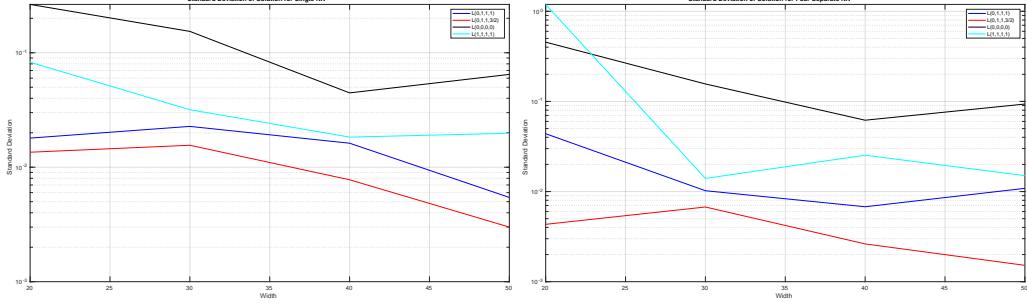


FIG. 4.6. Standard deviation of relative error over 10 samples, shown as the sum of all components. i.e.  $std(u) + std(v) + std(\omega) + std(p)$ . Left shows standard deviations of a single neural network and right shows four separate neural networks.

**A.3. Learning Rate.** We found a learning rate at  $10^{-2}$  or  $10^{-3}$  with an equal decay rate to give the best results. Noting that all experiments were run with 1 hidden layer, we did experiment with multiple hidden layers as well. Results were similar, but training becomes more sensitive to hyperparameters and the learning rate may have to be decreased even further. The intuition is that adding more layers adds higher complexity and the ability to have more oscillations which can make training more difficult.

**A.4. Batch-Size.** Batch sizes for Adam tended to be related to the amount of collocation points. We found the optimal Batch Size to be between 250 and 500 for 2500-5000 collocation points.

- [1] S. AGMON, A. DOUGLIS, AND L. NIRENBERG, Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions ii, Communications on pure and applied mathematics, 17 (1964), pp. 35–92.
- [2] P. B. BOCHEV, Analysis of least-squares finite element methods for the navier–stokes equations, SIAM Journal on Numerical Analysis, 34 (1997), pp. 1817–1844.
- [3] P. B. BOCHEV AND M. D. GUNZBURGER, Analysis of Least Squares Finite Element Methods for the Stokes Equations, Mathematics of Computation, 63 (1994), pp. 479–506. Publisher: American Mathematical Society.
- [4] ———, Finite element methods of least-squares type, SIAM Review, 40 (1998), pp. 789–837.
- [5] J. CHO, S. NAM, H. YANG, S.-B. YUN, Y. HONG, AND E. PARK, Separable pinn: Mitigating the curse of dimensionality in physics-informed neural networks, 2022.
- [6] S. CUOMO, V. S. DI COLA, F. GIAMPAOLO, G. ROZZA, M. RAISSI, AND F. PICCIALLI, Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next, Journal of Scientific Computing, 92 (2022), p. 88.
- [7] W. M. CZARNECKI, S. OSINDERO, M. JADERBERG, G. SWIRSZCZ, AND R. PASCANU, Sobolev training for neural networks, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017.
- [8] M. W. M. G. DISSANAYAKE AND N. PHAN-THIEN, Neural-network-based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.
- [9] T. G. GROSSMANN, U. J. KOMOROWSKA, J. LATZ, AND C.-B. SCHÖNLIEB, Can physics-informed neural networks beat the finite element method?, 2023.
- [10] Z. HU, K. SHUKLA, G. E. KARNIADAKIS, AND K. KAWAGUCHI, Tackling the curse of dimensionality with physics-informed neural networks, 2023.
- [11] I. LAGARIS, A. LIKAS, AND D. FOTIADIS, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9 (1998), pp. 987–1000.
- [12] M. RAISSI, P. PERDIKARIS, AND G. E. KARNIADAKIS, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378 (2019), pp. 686–707.
- [13] H. SON, J. W. JANG, W. J. HAN, AND H. J. HWANG, Sobolev training for physics informed neural networks, 2021.

## THE EFFECTS OF ORGANIZING TRAINING DATA FOR MACHINE LEARNED POTENTIALS

COREEN MULLEN\* AND EMBER L. SIKORSKI†

**Abstract.** Training data is one of the most important pieces to creating a Machine Learning algorithm. There is currently no predefined methodology for a scientist to select training data before making a machine learning potential. At the moment, the practice is that the scientist selects their training data by using their intuition. This project shows an attempt at sorting training data to see if there is an effect, whether negative or positive, on the results of the potential after sorting. The main work of this project was done by creating a Python script that is able to sort the training data and then run a fit on the data with the use of the FitSNAP python package. After getting the results from the script, a conclusion was made that with this certain script there is a negative effect on a Machine Learning Potentials results.

**1. Introduction.** The presence of Artificial Intelligence (AI) is growing exponentially each day with no signs of slowing down, Machine Learning (ML) is a direct subcategory of AI and supports the learning process of the AI system. Training Data is the true teacher for these two subjects, as training data provides the data that the ML potential will use to teach itself, it is comparable to a textbook and a student. The purpose of this research is to determine if there is an effect on the machine learning potentials results after sorting the data it is going to learn from.

Machine Learning has a wide outreach into many different fields of study, from Farming and Cosmetics, to Healthcare and Biology. According to a recent meta-study, from three of the study's looked at, 34 out of 36[1] AI systems were less accurate than one radiologist, and all were less accurate than two radiologists in screening for breast cancer. After reading this statistic, one might wonder how to decrease that number, to do that we have to increase the accuracy of the systems.

This paper uses data from the Materials Science field, Material Scientists study both the chemical and physical properties of structures for different materials. Machine Learning is helpful to Material Scientists because they are able to test the effects of different environments on the materials computationally. Material Scientists work together with experimentalist's to prevent them from running experiments that cost both time and money and which might provide results that are not beneficial for the experimentalist's research.

This paper serves as an attempt to study one possible solution in increasing the accuracy of Machine-Learning (ML) interatomic potentials (ML-IAP). The attempt starts below the Machine Learning level and looks at the training data that teaches the ML potential. The research involves organizing training data before it is implemented into a potential.

Training data can be set up as either unsupervised, supervised, or a mix of both. The data that this project is being tested on is using supervised learning. Supervised training data involves the selection of training data with human involvement, the data is labeled and then trains the algorithm. Once the algorithm is trained by this labelled data it can recognize new data that is put into it, which would be the data for the potential. The data that was used for this project is taken from a paper about the creation of FitSNAP, a ML package for LAMMPS, which is a molecular dynamics code[3]. The data includes information about different structures for a variety of elements. Some of the information included is the stress, force, and energy of the system. Training data can be set up as either unsupervised, supervised, or a mix of both. The data that this project is being tested on is using supervised learning. Supervised training data involves the selection of

---

\*Elizabeth City State University, cmmullen879@students.ecsu.edu

†Sandia National Labs, elsikor@sandia.gov

training data with human involvement, the data is labeled and then trains the algorithm . Once the algorithm is trained by this labelled data it can recognize new data that is put into it, which would be the data for the potential. The data that was used for this project is taken from a paper about the creation of FitSNAP, a ML package for Large-scale Atomic/Molecular Massively Parallel Simulator(LAMMPS), which is a molecular dynamics code[3]. The data includes information about different structures for a variety of elements. Some of the information included is the stress, force, and energy of the system.

The elements included in the data are all metals: Indium Phosphide, Tungsten Beryllium, Tantalum, and Iron. Testing on single elements like Tantalum and Iron was most likely done because they are basic tests to make fits for, creating fits for two element systems is a bit harder but it is still a common process that material scientists need to be able to do. The sorter that was created for this project is a python script that scrapes data from a folder of .json files and sorts them based on the information that is important to the scientist.

**2. Related Works.** For every unique set of training data that is going to be used for a Machine Learning algorithm, there is not just one answer for if it should be sorted or not before it is used. The opinions of different researchers varies, but the results for different data will also vary. In the majority of previous research, researchers have found that having a randomized mix of data is useful to avoid over or underfitting the data. Overfitting data happens when the model learns the data but it learns it so much that even a tiny fault will be detected as an error, Underfitting would be if the model itself was not capable of learning what it needed too from the data. Both of these situations would cause errors in the results of the model. In other cases there are researchers that are testing to see if Sorted Learning Algorithms [2] or sorting with the help of Neural Networks [4] would be useful for different data sorting scenerios. In learned sorting the data goes in unsorted and is sorted in its output, once this process is done the model would then be used on new unseen test data and those results are what would be compared to see any effects of sorting for the model.

**3. Sorter Overview.** The purpose of this Python script is to sort a set of folders containing a specific type of file ( .json ) and then sort those files based on the value of a specific keyword ( "Energy" ), so that the effects of sorting on training data can be examined. To get the results needed for comparison there are two separate scripts involved, the sorter itself, and the script that runs the FitSNAP fit on the output from the sorter. The sorter itself does two main things: It reads through the file for the value needed, and it sorts those values into chunks.

	ncount	mae	rmse	rsq
(*ALL, 'Unweighted', 'Energy')	363	0.112787	0.379769	0.994191
(*ALL, 'Unweighted', 'Force')	12672	0.0757576	0.160973	0.985504
(*ALL, 'Unweighted', 'Stress')	2178	68338.6	381744	0.939617

FIG. 3.1. The original metrics.md information, where "mae" is the metric being observed.

**3.1. Script.** The final data sorter script is written using the help of many different packages, for both sorting and file organization. At the beginning of the script there is an area where the file type and folder can be selected researcher. By changing the search term variables the person using the script will be able to select the folder or folders they are trying to sort through. It is possible to sort through more than one element when using this

	<b>ncount</b>	<b>mae</b>	<b>rmse</b>	<b>rsq</b>
(*ALL', 'Unweighted', 'Training', 'Energy')	363	1.58972	4.89882	0.033347
(*ALL', 'Unweighted', 'Training', 'Force')	12672	0.0655073	0.138075	0.989335
(*ALL', 'Unweighted', 'Training', 'Stress')	2178	393956	3.17452e+06	-3.17565

FIG. 3.2. *The sorted metrics.md information, where "mae" is the metric being observed.*

script so if needed, a scientist could compare in this specific data set the metrics for both Tantalum and Iron together.

The main function of this script is the function that finds the ‘item’ that the researcher wants to sort by. A .json file has the class type of string but the information inside the string is formatted as a dictionary. The json.loads() function turns the string into a dictionary so that it can then be searched through by the keys of a dictionary. In the specific data that this research is using, the function searches through either the first or second line in the file and stores the values of the search term into a variable. The search term from all of the files are then sorted into chunks, these chunks are sorted by the desired number of groups which is at the discretion of the scientist. After the data is sorted into chunks they are added to a csv file that the second script will use to perform the fit.

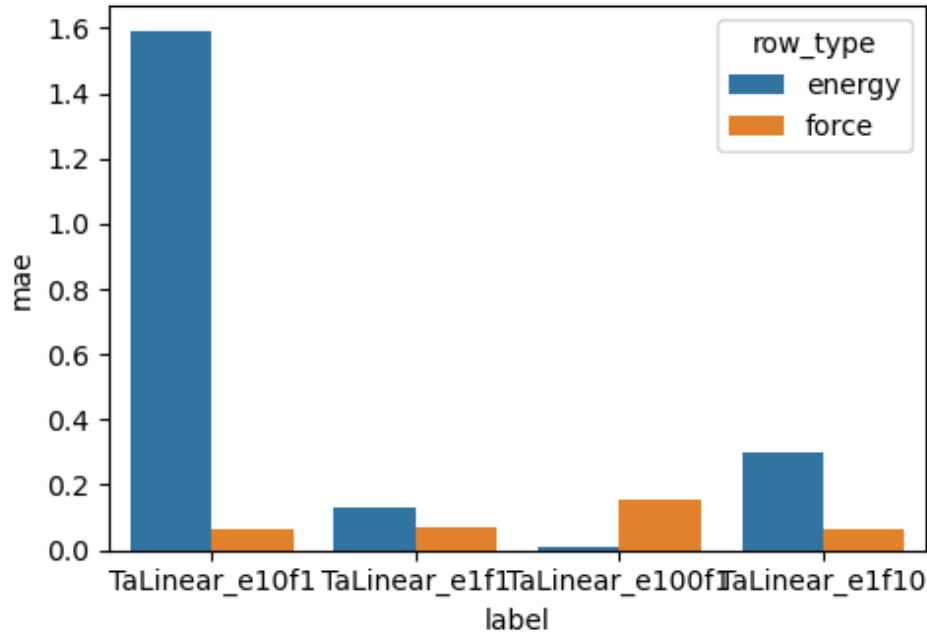


FIG. 3.3. *A barplot for the Mean Absolute Error of the sorted data from all the different environments. The labels show the energy and force levels in the name. For example TaLinear\_e10f1 has a energy of 10 and a force of 1.*

**3.2. Material Science Script.** The second script in this project is the one that runs the FitSNAP fit on the sorted data. This second script is most useful for the scientific portion of the research and as a way to see the effects on sorting when the data is put under

different conditions. As seen in Figure 3.3, Having a low energy creates a very high error in the results after being sorted. In the case of energy and force both being 1, the energy is the higher of the two. The reason for this is possibly due to the inclusion of the Liquid folder for TaLinear, the values of the Liquid energys are extreme outliers while the Forces in both the Liquid and other folders are more similar.

**3.3. Troubleshooting.** One of the largest blocks in creating the script was getting the value needed from the .json file due to different formatting. The specific variable that we worked with was “Energy”, this variable was inside of a nested dictionary in the file and would sometimes appear on a separate line. The data set that was used in this project had a comment line in the .json files, while it is more common for it to just be one line. A small section in the code was added to solve this potential problem by looking to see if the file had one line or two and then reading the data from the first or second line depending on the answer. During the process of making the script, trying to sort the data was also a difficult process, many different methods of sorting were attempted. Eventually the CSV file and DataFrame solution was made and worked the best for this testing. One original test was that the “Energy” values were being added to a Python list, and then sorted. After the list was sorted it became a challenge to sort the list into chunks so the script had to be reworked multiple times for all the different ideas for getting the information sorted.

```

1  # A test JSON file for QM data
2  {"Dataset": {"Data": [{"Stress": [[44613.38, 397.28000000000003, -242.23], [397.28000000000003, 46401.21, -102.86], [-242.23, -102.86, 43613.28000000006]], "Positions": [[

```

FIG. 3.4. The format of the Tantalum .json files.

```

1  # Header
2
3  {
4      "Dataset": {
5          "AtomTypestyle": "cheinallsymbol",
6          "data": [
7              {
8                  "AtomTypes": [
9                      "In",
10                     "In",
11

```

FIG. 3.5. The .json file format of Indium Phosphide

The script was mainly tested on the Tantalum data, it was also tested on the other elements, however for Indium Phosphide it was unable to sort. The .json files for Indium Phosphide are formatted differently than the other elements, the different forms of Indium Phosphide also had different lengths for their .json files. Due to this, one way to improve the script would be creating a way for it to do a type of .readlines() but for the entire file. It would make the script better not just for .json's but for all other file types. Figure 3.6 shows how much of an outlier Liquid was for the energy sorted TaLinear data, it also shows the range of the energys throughout the folders. There is one dot for each .json in all of the TaLinear set which had 363 .jsons in total. The size of TaLinear was smaller than the other elements, due to the sorter not being optimized it was also better to work with the smaller dataset. When the test was ran on the WBe element it took a considerably longer time to get the sorted data.

**4. Results.** The comparison of the data before and after sorting is made by looking at the speed of the FitSNAP run as well as the accuracy of the data compared to the target material properties. Mean Absolute Error and Root Mean Squared Error are two values that are included in the metrics.md, these values are two different ways of evaluating how accurate the fit is. From the results of the sorting on the TaLinear set, it can be seen that

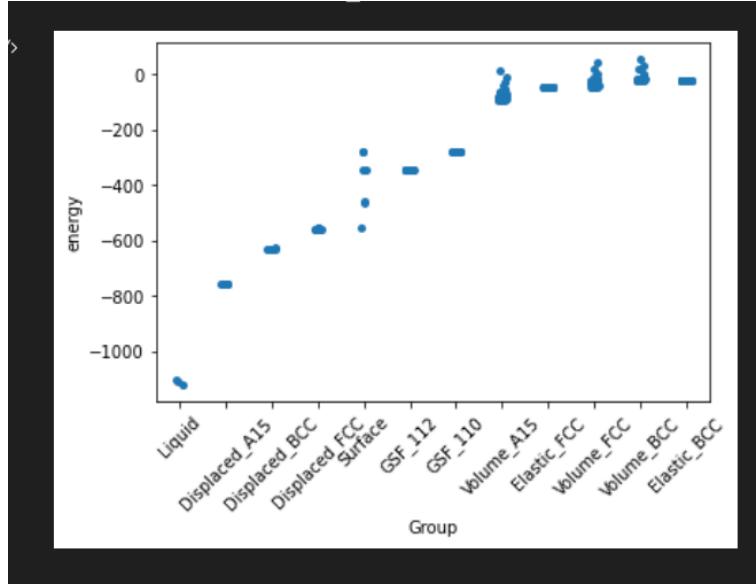


FIG. 3.6. A graph showing the energys for TaLinear based off of their type.

the MAE scores are quite high and do not show any positive results. Figure 4.1 shows the Mean Absolute Error of the data from folders with different eweights and fweights for TaLinear on a scatterplot. The accuracy of the data is read by seeing how close the score is to 0, in this plot it can be seen that the data can go from near 0 to 1.6 which is a very high error.

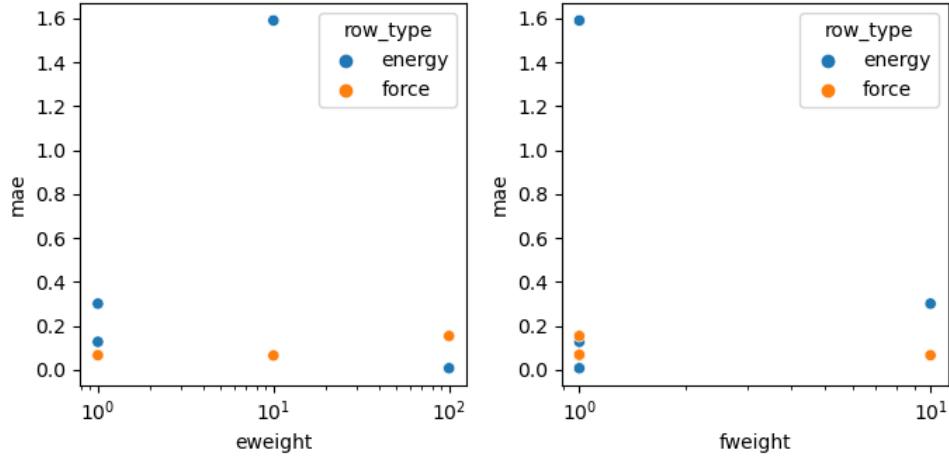


FIG. 4.1. A scatterplot for the Mean Absolute Error of the sorted data from all the different environments.

**5. Conclusion.** The purpose of this research was to determine if there was an effect on the results of a ML potential after sorting training data. The "Energy" results from this specific script gave a Mean Absolute error of 0.11 in the original test, and a Mean Absolute

error of 1.5 after being sorted. This shows an error increase of a little over 13x. Due to this, it can be concluded that in this specific scenario, sorting the data does not provide any benefit toward speed or accuracy in the data. The future work for this project would involve fully optimizing the script to make sure that the script isn't effecting the results of the tests too much. Another part of the future work is making the script even more usable for different file types and files of the same type with different formats. Even though the sorter did not provide a benefit in this specific case it is still something that can be tested on for different cases.

**6. Acknowledgements.** This research was funded by the Minority Serving Institution Partnership Program Nuclear Security Advanced Manufacturing Enhanced by Machine Learning (MSIPP NSAM-ML) program. I would like to sincerely thank Megan McCarthy for her coding knowledge and assistance with this project. The FusMAT ML team for their feedback on both the project and this paper. My fellow interns for their encouragement. As well as David Littlewood for helping me along this opportunity.

#### REFERENCES

- [1] K. FREEMAN, J. GEPPERT, C. STINTON, D. TODKILL, S. JOHNSON, A. CLARKE, AND S. TAYLOR-PHILLIPS, *Use of artificial intelligence for image analysis in breast cancer screening programmes: systematic review of test accuracy*, BMJ, 374 (2021).
- [2] A. KRISTO, K. VAIDYA, U. ÇETINTEMEL, S. MISRA, AND T. KRASKA, *The case for a learned sorting algorithm*, (2020), p. 1001–1016.
- [3] A. ROHSKOPF, C. SIEVERS, N. LUBBERS, M. CUSENTINO, J. GOFF, J. JANSSEN, M. McCARTHY, D. M. O. DE ZAPIAIN, S. NIKOLOV, K. SARGSYAN, D. SEMA, E. SIKORSKI, L. WILLIAMS, A. THOMPSON, AND M. WOOD, *Fitsnap: Atomistic machine learning with lammps*, Journal of Open Source Software, 8 (2023), p. 5118.
- [4] Q. ZHU, Y. ZHAO, D. HU, D. HUANG, Y. LIU, Z. YANG, L. MAO, C. LIU, AND F. ZHOU, *Sorting data via a look-up-table neural network and self-regulating index*, (2020).

## IDENTIFICATION OF MODEL FORM ERROR: A RANDOM BEGINNING

JOËD N. NGANGMENI \* AND KATHRYN A. MAUPIN †

**Abstract.** With the advent of computers, computational modeling became the third pillar of science, complementing the previous pair of theory and experimentation. Physics based models, derived and implemented according to first principles and calibrated to experimental data, inevitably rely on modeling choices made by subject matter experts. Thus, even the most reliable computational models include uncertainties in the experimental data, parameter values, and model form. Our long term goal is to obtain the ability to identify, isolate, and mitigate model form error. This will lead to more accurate results in modeling and simulation. The first step to doing this is to develop a model adept at recognizing data from a specified distribution. As a first step towards this goal, a random forest model was created and fit to given experimental data. One particular area of focus was to test the reliability of the random forest model to adequately generalize for a distribution, which was tested and compared by varying the number of trees in the algorithm.

**1. Introduction.** Computational modeling is crucial to modern scientific pursuits. Despite the continued advancement of computing power and resources, uncertainties lurk throughout the modeling process. Experimental data is subject to measurement error; numerical methods and approximations plague even the highest fidelity models; and model formulations, including model form and model parameters, may be inadequate. The goal of uncertainty quantification (UQ) as a whole is identify the degree to which we can have confidence in a model, function, or simulation. It is informational for scenarios in which we estimate some variable. In such cases, we can quantify either the uncertainty in the variable itself as an isolated end product or we can calculate this measurement by combining the uncertainties of all the measurements that culminated into the calculation of the target variable. The more accurate our quantification of uncertainty in the variable, the more confidence we can have in the extent to which our variable is descriptive of what it is supposed to measure.

Following this introduction, Section 2 provides an overview of how model form error is treated in this work, as well as a description of the machine learning model used in the example. The error metrics employed to refine the accuracy of the model are described in Section 3. Details regarding how the machine learning model is constructed are given using an illustrating example in Section 4 before concluding remarks are given in Section 5.

**2. Machine Learning and Model Form Error.** Contemporary machine learning practice makes use of some variation of the following function,

$$d(x) = m(\theta, x) + \varepsilon \quad (2.1)$$

where  $d(x)$  is experimental data,  $m(\theta, x)$  represents model output with  $\theta$  referring to model parameters, and  $\varepsilon$  represents error.

In this particular formulation of the function, every form of possible error, from experimental, to model, to other kinds of error are assumed to be accounted for by the  $\varepsilon$  variable. However, the formulation (2.1) is nondescript as to where the different pieces of information that go into the final calculation of  $\varepsilon$  are coming from. In other words, since  $\varepsilon$  is the culmination of multiple sources of error, we cannot know which or by how much each part of the modeling pipeline needs to be improved in order to enhance the predictivity of the model  $m$ .

---

\*Howard University, joed.ngangmeni@bison.howard.edu

†Sandia National Laboratories, kmaupin@sandia.gov

Examples of error include cases where experimental equipment may have some degree of bias, leading observations  $d$  to be slightly deviant from its expected value. The very machine learning model being used may have its own tendency to favor a specific outcome for reasons unintended by the designer. In the absence of an analytical solution, calibration methods are, unavoidably, approximate. In such cases, it will significantly improve confidence in the model, and subsequently the experiment as a whole, if we are able to characterize and classify the error stemming solely from the model itself, otherwise known as the **model form error**.

To address this, we slightly adapt the original formula, (2.1), into

$$d(x) = m(\theta, x) + \delta(x) + \varepsilon. \quad (2.2)$$

where  $\delta(x)$  is the model form error, isolated from the other forms of error. Rearranging this equation,

$$\delta(x) = d(x) - m(\theta, x) - \varepsilon. \quad (2.3)$$

the  $\delta(x)$  variable gives us a window through which we can effect the accuracy of the original model; and by so doing, we effect the uncertainty of its product. By identifying and eventually mitigating model form error in the high-fidelity model that generates our data, we can be more confident that the generated data is within a particular error bound of the distribution we are targeting.

Our long term goal is to develop the ability to identify, isolate, and mitigate the model form error  $\delta(x)$ . As a first step, the present work focuses on the development of a random forest regression model. In order to be able to generate data within a specific distribution, we have to create a model capable of discriminating that distribution, to within some pre-specified error bounds. By creating a keen random forest model to do that for the problem at hand, it can subsequently be used as the basis for a discriminator in a generative model.

Random forest regression models have been chosen due to several desirable features. They demonstrate robust performance in the presence of noise [1]. At the very least, experimental noise is unknown, though its presence is undeniable in all applications. Random forest models are able to handle both classification and regression tasks [1, 3], which is useful for identifying and characterizing model form error. Furthermore, a random forest model's generalization error approaches a limit as trees are added [1], thereby mitigating an additional uncertainty contribution.

**3. Sampling Method and Error Measurement.** Random Forest models can make use of a variety of sampling methods, including feature sampling and bagging, otherwise known as bootstrap sampling or bootstrapping. We relegated this experimental model to only use bootstrapping as its sampling method in order to utilize the **Out-Of-Bag** (OOB) error. For OOB to work, a random forest algorithm samples the initial dataset at random, with replacement, and then uses the newly randomly sampled data set as a basis from which trees in the forest were grown. Due to the nature of this method, an estimated 1/3 of data is excluded from the creation of each tree [1]. We can then use this excluded data as a test set to evaluate the performance of the model. Of the four error metrics detailed here, OOB is the only one specific to random forest models. It enables the calculation of internal estimates of the generalization error, classifier strength, and dependence without the need for typical cross validation [1].

The second error measurement featured here is the **Mean Absolute Error** (MAE). The MAE is the average of the absolute differences between predictions  $\hat{y}_i$  and actual values

$y_i$ ,

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|, \quad (3.1)$$

where  $n$  is the total number of observations. MAE provides a linear penalty for each unit of difference between the predicted and actual values. It is used to describe how far model predictions are, on average, from the experimental or high-fidelity simulation data.

A similar metric is the **Mean Squared Error**, which is the average of the squared differences between predictions and actual values,

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.2)$$

where, as before,  $y_i$  are the experimental or high-fidelity observations,  $\hat{y}_i$  are the corresponding model predictions, and  $n$  is the total number of observations. MSE provides a quadratic penalty for prediction errors.

The fourth and final error metric considered in this experiment with respect to the number of trees in a random forest model was the **Root Mean Squared Error**. The RMSE is the square root of the MSE. Since RMSE is in the same unit as the response variable, it may be considered to be more interpretable than MSE. Typically formulated as

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}, \quad (3.3)$$

the RMSE gives higher weight to larger errors. It is arguably the most widely used metric for regression tasks and offers a good balance between interpretability and sensitivity to large errors.

**4. Example.** As a motivating application, we consider a re-entry vehicle subject to extensive atmospheric forces. Experimental data is available from wind tunnel experiments, conducted at Calspan University at Buffalo Research Center. For the purposes of the analysis herein, we use high-fidelity simulation data produced by the Sandia Parallel Aerodynamics and Reentry Code (SPARC) [4]. This data was meant to be representative of organic experimental data. However, the SPARC model sometimes produces data that not only falls outside of the error bounds we set for it but is also divergent by a large margin from the distribution of experimental data. See reference [4] for further details.

The experimental data is parameterized by the freestream conditions of the flow: the density  $\rho$ , the velocity  $U$ , and the temperature  $T$ . Two key quantities of interest are measured at several spatial locations: the heatflux  $q$  and the pressure  $p$ . In the present work, we consider a single set of experimental conditions, with  $\rho = 0.3711g/m^3$ ,  $U = 2130m/s$ , and  $T = 44.8K$ , and a single quantity of interest, the heatflux. A sample data set of simulations was produced by sampling the parameter space near these nominal values and running SPARC. It is over these samples that the random forest model is constructed.

The Python library Scikit-learn was used to build the random forest model. To initialize the model, the `RandomForestRegressor` function was used with some initial conditions. The number of trees in the forest, `n_estimators`, was initially set to 1 so as to record a baseline for the performance of the model. The number of features to choose at each split, `max_features` was set to be the square root of the number of features in our data. This is standard practice for basic models as it caters well to both large and small datasets. Furthermore, prior

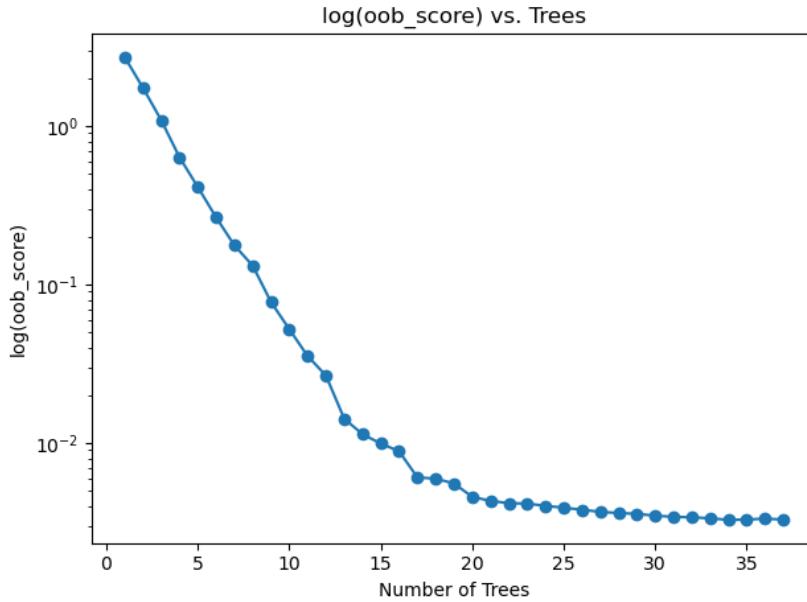


FIG. 4.1. This figure displays  $\log(\text{OOB})$  error versus the number of trees in the random forest. A tolerance of 0.0001 was chosen to determine stability, achieved with less than 40 trees.

research [1, 2] suggests that model performance with two arbitrary features at each split should be optimal or near optimal [1]. In order to enable Out-of-Bag scoring, `oob_score` was set to `True`. Finally, to make sure that the model updated its error measurements as more trees were added, `warm_start` was also set to `True`. With these settings in place, the random forest was ready to fit data and be used for the analysis detailed below, using `numpy` and `pandas`.

We employed an iterative scheme in which trees are added individually until the chosen error metrics converge, implying the convergence of the forest structure and complexity. Note that for more complex problems, this procedure may be used by adding groups of trees at each iteration. In this study, we consider a forest to be converged when the given metric is stable for 10 consecutive iterations.

When OOB is used as a metric, a tolerance of 0.0001 was chosen to determine stability. Figure 4.1 shows the convergence of OOB as trees are added to the forest. Figure 4.2 similarly shows the convergence of MAE as iterations progress. The stability point was set to be when the absolute value of the change in MAE was less than 1 for a consecutive 10 trees. Of the four metrics considered in this study, the MSE took the most iterations to stabilize, as shown in Figure 4.3. The forest was considered stable when the MSE was less than 1200 for 10 consecutive iterations. The convergence of the forest when the RMSE is used as the error metric is shown in Figure 4.4. Stability was reached when the the absolute value of the change in RMSE was less than 1.5 for 10 consecutive iterations.

Surprisingly, the Out-Of-Bag score incurred the fastest convergence of the random forest algorithm. As depicted in Figure 4.5, its initial slope is much steeper than any of the other metrics. It is possible that although the OOB score gives us an internal measurement of the stability of our algorithm, the other measures more harshly scrutinize pointwise comparisons. For example, the curve of the Mean Average Error is noticeably less steep than the others, likely due to the linear penalty for each unit of difference between the predicted and actual

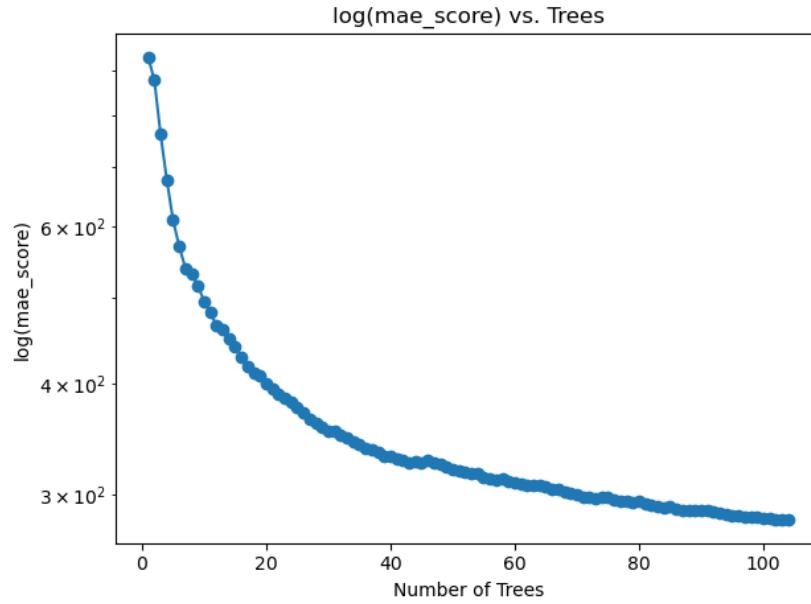


FIG. 4.2. This figure displays  $\log(\text{MAE})$  error versus the number of trees in the random forest. The forest is deemed stable when the absolute value of the change in MAE is less than 1 for 10 consecutive trees.

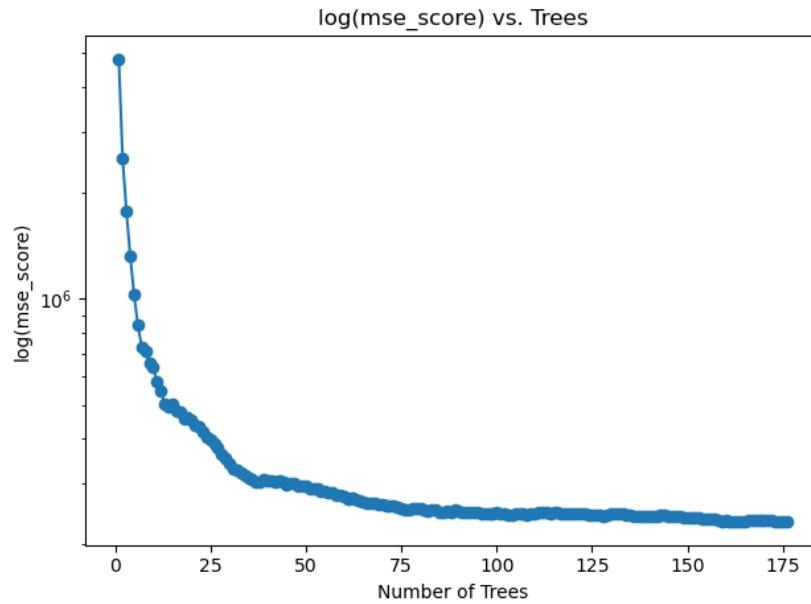


FIG. 4.3. This figure displays  $\log(\text{MSE})$  error versus the number of trees in the random forest. The forest is considered stable when the MSE is less than 1200 for 10 consecutive iterations.

values. It therefore takes a longer time to reduce when compared to other measures that reduce quadratically.

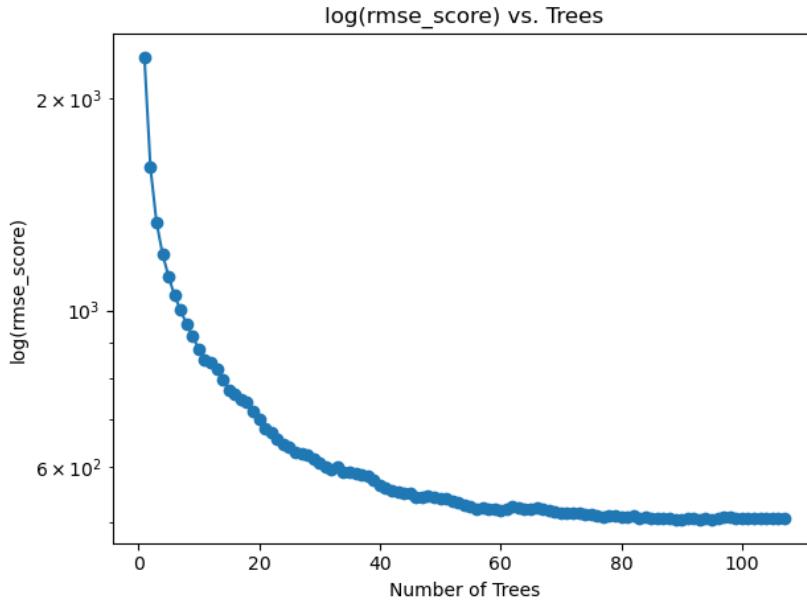


FIG. 4.4. This figure displays  $\log(\text{RMSE})$  error versus the number of trees in the random forest. The forest is deemed stable when the absolute value of the change in RMSE is less than 1.5 for 10 consecutive iterations.

Trials began with a focus on a singular target variable, i.e. the error metrics. Each time a tree was added to the forest, the error metrics for that specific instance were recorded. This process was repeated until the target variable stabilized. The result of all the error metrics for that singular batch were recorded. This was repeated 50 times for each target variable. Once 50 trials were run for each target variable, the results of the error metrics over those 50 trials were averaged. These averages are displayed in table (4.1).

**5. Conclusion.** A random forest regression model's performance in predicting the heatflux of a re-entry vehicle was the primary driver of this experiment. We were able to vary the number of trees in the model, keeping all other variables constant, to understand how the number of trees in a forest affect its performance. To assess that performance, we looked at some of the most common error metrics, namely, Out-Of-Bag Error, Mean Average Error, Mean Squared Error, Root Mean Squared Error. The results showed that Out-Of-Bag Error approaches a limit at a faster rate than the other metrics. With 1947 trees needed to stabilize over 50 trial runs, OOB was almost 3.5 times more efficient, in

	OOB Run	MAE Run	MSE Run	RMSE Run	ALL Runs
Trees	1947	4783	6968	6143	20041
OOB	0.19829	0.08253	0.05754	0.06471	0.07883
MAE	427.04	350.45	326.79	335.51	344.46
MSE	695323	448904	384405	404583	434938
RMSE	771.84	626.25	582.69	596.19	614.85

TABLE 4.1  
Average Metrics (50 Runs Per Metric)

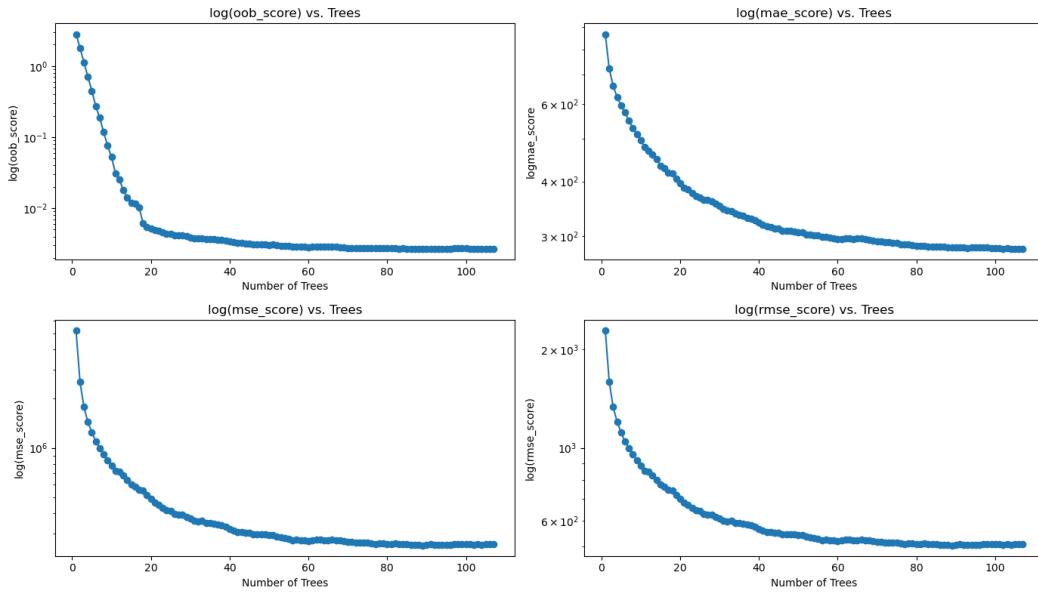


FIG. 4.5. This figure displays comparisons of convergence rates for four error metrics. The OOB score (top left) incurs the fastest convergence. The MAE (top right) is noticeably less steep than the other metrics. The MSE (bottom left) has a stronger penalty for larger errors. The RMSE (bottom right) is arguably the most widely used error metric for regression.

terms of trees needed to stabilize, than MSE which took the largest amount of trees to stabilize, 6968. By dividing 6968 by 50, we know that 150 trees is a safe amount to use for our re-entry problem to ensure that the accuracy of the random forest model is good enough.

#### REFERENCES

- [1] L. BREIMAN, *Random forests*, Machine learning, 45 (2001), pp. 5–32.
- [2] N. LAY, A. P. HARRISON, S. SCHREIBER, G. DAWER, AND A. BARBU, *Random hinge forest for differentiable learning*, 2018.
- [3] P. PROBST, M. N. WRIGHT, AND A.-L. BOULESTEIX, *Hyperparameters and tuning strategies for random forest*, WIRES Data Mining and Knowledge Discovery, 9 (2019), p. e1301.
- [4] J. RAY, S. KIEWEG, D. DINZL, B. CARNES, V. G. WEIRS, B. FRENO, M. HOWARD, T. SMITH, I. NOMPELIS, AND G. V. CANDLER, *Estimation of inflow uncertainties in laminar hypersonic double-cone experiments*, AIAA journal, 58 (2020), pp. 4461–4474.

## DOMAIN DECOMPOSITION-BASED COUPLING OF PHYSICS-INFORMED NEURAL NETWORKS VIA THE SCHWARZ ALTERNATING METHOD

WILLIAM SNYDER\*, IRINA TEZAUR†, AND CHRISTOPHER R. WENTLAND‡

**Abstract.** Physics-informed neural networks (PINNs) are appealing data-driven tools for solving and inferring solutions to nonlinear partial differential equations (PDEs). Unlike traditional neural networks (NNs), which train only on solution data, a PINN incorporates a PDE’s residual into its loss function and trains to minimize the said residual at a set of collocation points in the solution domain. This paper explores the use of the Schwarz alternating method as a means to couple PINNs with each other and with conventional numerical models (i.e., full order models, or FOMs, obtained via the finite element, finite difference or finite volume methods) following a decomposition of the physical domain. It is well-known that training a PINN can be difficult when the PDE solution has steep gradients. We investigate herein the use of domain decomposition and the Schwarz alternating method as a means to accelerate the PINN training phase. Within this context, we explore different approaches for imposing Dirichlet boundary conditions within each subdomain PINN: weakly through the loss and/or strongly through a solution transformation. As a numerical example, we consider the one-dimensional steady state advection-diffusion equation in the advection-dominated (high Péclet) regime. Our results suggest that the convergence of the Schwarz method is strongly linked to the choice of boundary condition implementation within the PINNs being coupled. Surprisingly, strong enforcement of the Schwarz boundary conditions does not always lead to a faster convergence of the method. While it is not clear from our preliminary study that the PINN-PINN coupling via the Schwarz alternating method accelerates PINN convergence in the advection-dominated regime, it reveals that PINN training *can* be improved substantially for Péclet numbers as high as  $1.0 \times 10^6$  by performing a PINN-FOM coupling.

**1. Introduction.** In recent years, physics-informed neural networks (PINNs) [23] have emerged as a scientific machine learning (ML) technique advertised to solve partial differential equations (PDEs) without requiring any training data or an underlying mesh. This is accomplished by constructing and minimizing a loss function that includes the residual of the underlying PDE, sampled at a finite number of collocation points within the physical domain on which the PDE is posed. While the aforementioned properties of PINNs make them appealing within the modeling and simulation community, these models unfortunately suffer from several deficiencies. First, PINNs are notoriously difficult to train for problems having a slowly-decaying Kolmogorov  $n$ -width, as discussed in [19] and the references therein. This class of problems includes advection-dominated flow problems in which the solution exhibits sharp boundary layers and/or shocks. Second, since PINNs are trained for a given geometry with a given set of boundary conditions, they require retraining when applied to different geometries or boundary data [28].

In this paper, we present and evaluate a promising technique that can mitigate both difficulties described above: the Schwarz alternating method [25] for domain decomposition-based coupling. The Schwarz alternating method is based on the simple idea that if the solution to a PDE is known in two or more regularly-shaped subdomains comprising a more complex domain, these local solutions can be used to iteratively build a solution on the more complex domain, with information propagating between subdomains through boundary conditions imposed on the subdomain boundaries. In several of our recent works [21, 22], the overlapping version of the Schwarz alternating method was pioneered as a mechanism for performing concurrent coupling of subdomains discretized in space by disparate meshes and in time by different time-integration schemes with different time-steps. In those works, attention was restricted to the coupling of high-fidelity full order models (FOMs) in quasistatic and dynamic solid mechanics. Our recent work [2] extended the approach to the coupling of

---

\*Virginia Polytechnic Institute, swilli9@vt.edu

†Sandia National Laboratories, ikalash@sandia.gov

‡Sandia National Laboratories, crwentl@sandia.gov

projection-based reduced order models (ROMs) with each other and with FOMs following either an overlapping or a non-overlapping domain decomposition (DD) of the underlying geometry. We additionally demonstrated in [13, 11] that it is possible to transform the non-overlapping Schwarz alternating method into a novel contact enforcement algorithm which exhibits remarkable energy conservation properties.

The present work explores the use of the Schwarz alternating method for coupling PINNs with each other and with FOMs, following an overlapping DD of the underlying physical domain. For a detailed literature overview on existing methods which combine ML and domain decomposition methods, the reader is referred to [10]. While it is possible to couple pre-trained subdomain-local PINNs or NNs, as in [28], our initial study focuses primarily on evaluating the Schwarz alternating method as a means to facilitate PINN training. The proposed approach is similar to the deep domain decomposition method (D3M) and the deep domain decomposition (DeepDDM) methods, proposed recently in [15] and [16], respectively. Unlike the D3M method, which is based on a deep Ritz method PINN formulation in which the loss function is derived from the weak variational form of the governing PDEs [5, 26], our loss function incorporates the PDE residual in strong form. Unlike the DeepDDM method, which imposes Schwarz and other boundary conditions weakly through a boundary loss contribution to the loss function being minimized, we consider both strong and weak formulations of the Dirichlet boundary conditions (DBCs) within our algorithm, as well as a combination of the two. For the former strong DBC enforcement, we borrow ideas from the Finite Basis PINN (FBPINN) literature [20, 4]. While the authors of [15, 16] focus their development and demonstrations on the Poisson equation, our work considers a much more difficult boundary value problem (BVP), namely the advection-diffusion equation in the advection-dominated (high Péclet) regime. Finally, to the best of our knowledge, ours is the first coupling formulation that enabling the creation of “hybrid” models through the coupling of PINNs with conventional numerical models, such as those constructed using the finite element, finite difference, or finite volume method.

The remainder of this paper is organized as follows. Section 2 defines the model problem considered herein, a one-dimensional (1D) advection-diffusion equation on the unit interval. Section 3 summarizes the overlapping version of the Schwarz alternating method for domain decomposition-based coupling in the specific context of our advection-diffusion model problem. Section 4 provides an overview of PINNs and several PINN variants, namely PINNs in which DBCs are enforced strongly, rather than weakly, and PINNs in which a data loss term is incorporated into the loss function being minimized. In Section 5, we extend the general Schwarz formulation described in Section 3 to the specific case of PINN-PINN and PINN-FOM coupling. The proposed method is evaluated as a mechanism for facilitating PINN training via domain decomposition-based coupling on our model advection-diffusion problem in Section 6. Whereas a definitive conclusion on whether or not PINN training can be facilitated through PINN-PINN coupling using the Schwarz alternating method cannot be made from the present study, our numerical experiments reveal that this goal *can* be achieved by performing a Schwarz-based PINN-FOM coupling. Finally, we provide some conclusions and directions for future work in Section 7.

**2. Model problem.** As a numerical example to test our coupling method and PINNs, we choose the 1D steady-state advection-diffusion equation defined in an open bounded domain  $\Omega = (0, 1)$  with boundary  $\partial\Omega = \{0, 1\}$ :

$$-\nu \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} - 1 = 0 \quad \text{in } \Omega = (0, 1), \quad (2.1)$$

with boundary conditions

$$u(0) = u(1) = 0. \quad (2.2)$$

From this point forward, we will refer to the DBCs in (2.2) as the “system DBCs” or “system boundary conditions”. In (2.1),  $\nu$  is the diffusion coefficient, which defines the Péclet number of the problem, given by  $Pe := \nu^{-1}$ . Our focus here is primarily on advection-dominated regimes for this problem, i.e.,  $10 \leq Pe \leq 10^6$ . Figure 2.1 shows a plot of the solution to (2.1)–(2.2) as a function of the Péclet number. The reader can observe that the solution develops a sharp gradient near the right boundary of the domain, whose steepness is a function of  $Pe$ .

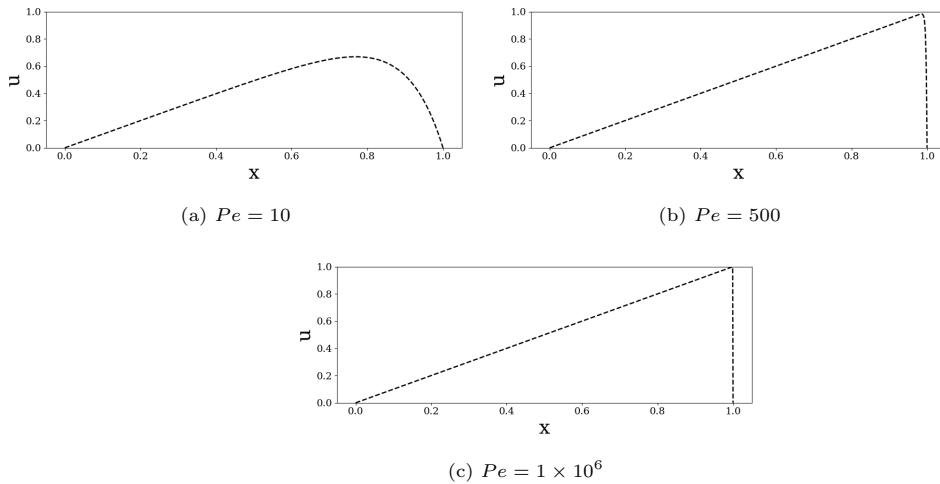


Fig. 2.1: Solutions to our model advection-diffusion problem for different Péclet numbers  $Pe$ .

**3. The Schwarz alternating method for domain decomposition-based coupling.** The Schwarz alternating method is the oldest known domain decomposition method, first proposed in 1870 by H. Schwarz [25]. The method is based on the simple idea that the solution to a PDE on a complex domain  $\Omega$  can be obtained via an iterative procedure on subdomains comprising  $\Omega$ , with information propagating via transmission (or boundary) conditions imposed on subdomain boundaries. While the Schwarz alternating method can be formulated for both overlapping [21, 22, 17] and non-overlapping [2, 13, 11] DDs, we restrict our attention to the overlapping variant of the method, which has been shown to have favorable convergence properties with respect to its non-overlapping counterpart [2].

Consider an overlapping DD of a geometry  $\Omega$  into two subdomains, as shown in Figures 3.1(a) and (b) in one and two spatial dimensions, respectively. We present the basic overlapping Schwarz algorithm on our targeted model problem (2.1) with system boundary conditions (2.2). Suppose the governing domain for our model problem is partitioned into two overlapping domains:  $\Omega = \Omega_1 \cup \Omega_2$ , with  $\Omega_1 = (0, \gamma_1)$  and  $\Omega_2 = (\gamma_2, 1)$ , with  $0 < \gamma_2 < \gamma_1 < 1$ , so that  $\Omega_1 \cap \Omega_2 \neq \emptyset$ , as shown in Figure 3.1(a). Equations (2.1)–(2.2) can be solved using the Schwarz alternating method by performing the following iteration:

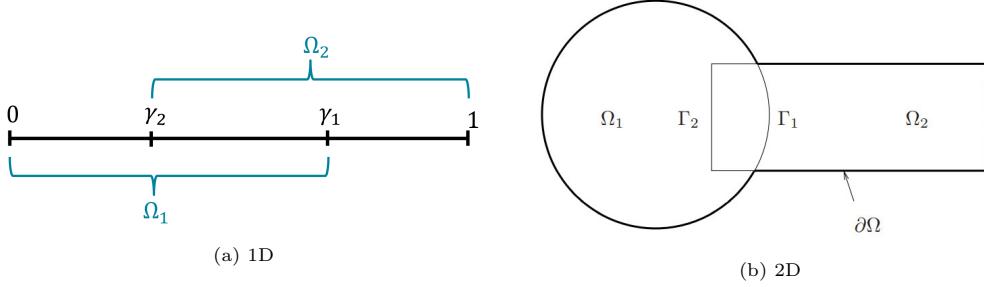


Fig. 3.1: Example overlapping domain decompositions for Schwarz alternating method in 1D (left) and 2D (right).

$$\begin{cases} -\nu \frac{\partial^2 u_1^{(n+1)}}{\partial x^2} + \frac{\partial u_1^{(n+1)}}{\partial x} = 1, & \text{in } \Omega_1, \\ u_1^{(n+1)} = 0, & \text{at } x = 0, \\ u_1^{(n+1)} = u_2^{(n)}, & \text{at } x = \gamma_1, \end{cases} \quad \begin{cases} -\nu \frac{\partial^2 u_2^{(n+1)}}{\partial x^2} + \frac{\partial u_2^{(n+1)}}{\partial x} = 1, & \text{in } \Omega_2, \\ u_2^{(n+1)} = 0, & \text{at } x = 1, \\ u_2^{(n+1)} = u_1^{(n+1)}, & \text{at } x = \gamma_2, \end{cases} \quad (3.1)$$

for Schwarz iterations  $n = 0, 1, 2, \dots$ . In (3.1),  $u_i$  denotes the solution in  $\Omega_i$  for  $i = 1, 2$ . The reader can observe that the transmission boundary conditions on the Schwarz boundaries  $x = \gamma_1$  and  $x = \gamma_2$  are of the Dirichlet type. The iteration (3.1) continues until convergence is reached. Generally, convergence of the method is declared<sup>1</sup> when  $\|u_i^{(n+1)} - u_i^{(n)}\|_2 / \|u_i^{(n)}\|_2 < \delta$ , for  $i = 1, 2$  and for some specified tolerance  $\delta$ . In the remainder of this paper, the boundaries  $\gamma_i$  will be referred to as the “Schwarz boundaries” and the boundary conditions applied at these boundaries will be referred to as the “Schwarz boundary conditions”. It can be shown [17, 21] that the Schwarz iteration procedure (3.1) converges to the solution of (2.1)–(2.2) provided the overlap is non-empty ( $\Omega_1 \cap \Omega_2 \neq \emptyset$ ). It is straightforward to extend the Schwarz iteration procedure (3.1) to an arbitrary number of overlapping subdomains. The present work builds on recent extensions of the Schwarz alternating method to concurrent multi-scale coupling in quasistatic [21] and dynamic [22] solid mechanics, and to the coupling of projection-based reduced order models (ROMs) with each other and with high-fidelity full order models (FOMs) [2].

**Remark 1.** The Schwarz iteration procedure described above in (3.1) is often referred to as the multiplicative Schwarz algorithm [7]. In this algorithm, the solution in subdomain  $\Omega_2$  at iteration  $n+1$  depends on the solution in subdomain  $\Omega_1$  also at iteration  $n+1$ . This contemporaneous dependence prevents one from solving the  $\Omega_1$  and  $\Omega_2$  problems in parallel. The Schwarz iteration can be modified to achieve what is commonly referred to as additive Schwarz [7] by applying boundary conditions from the  $n^{th}$  Schwarz iteration procedure in  $\Omega_1$  to the  $(n+1)^{st}$   $\Omega_2$  sub-problem. If this change is made, the Schwarz iteration sequence within can be parallelized over the number of subdomains, as done in [15, 16]. While we do not consider the additive variant of the Schwarz method in the present work, preliminary

<sup>1</sup>Herein, for the data-driven models coupled via the Schwarz alternating method, we also require the solution to be within a given tolerance from the known analytical or high-fidelity solution to the given problem; more details are provided in Section 6.

results have suggested that the method does not reduce solution accuracy and can achieve speed-ups if parallelized appropriately.

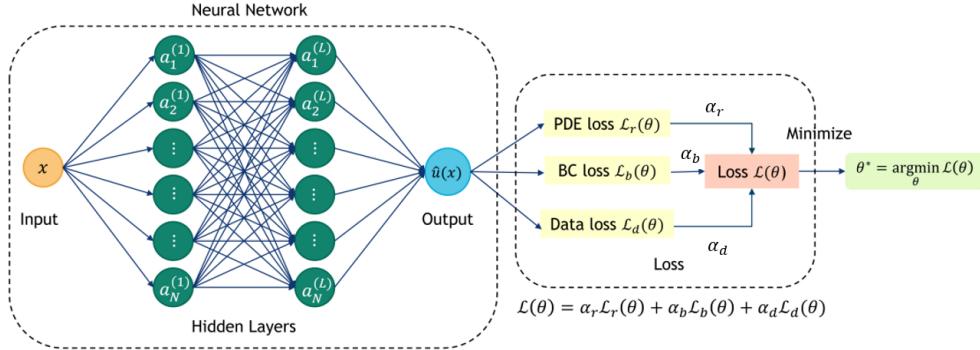


Fig. 4.1: Illustration of our PINN architecture for  $L = 2$  layers each having  $N = 6$  nodes per layer.

**4. Physics-informed neural networks (PINNs).** PINNs, introduced in the seminal paper [23], are constructed via a least-squares collocation method applied to the governing PDEs, the solution of which is approximated by a neural network function, denoted by  $NN(x; \theta)$ . Let  $x$  denote the input and  $\hat{u}$  denote the output of a typical feed-forward fully-connected NN<sup>2</sup> with  $L$  hidden layers and  $N$  neurons in each layer, as shown in Figure 4.1. The output of the network is computed as

$$NN(x; \theta) = \sigma \left( \sum_{i=1}^N w_{1,i}^{(L)} a_i^{(L)} + b^{(L)} \right), \quad (4.1)$$

where

$$a_n^{(j)} = \sigma \left( \sum_{i=1}^N w_{k,i}^{(j-1)} a_i^{(j-1)} + b^{(j-1)} \right), \quad (4.2)$$

for  $j = 2, \dots, L$  and  $k = 1, \dots, N$ , and

$$a_n^{(1)} = \sigma \left( w_{k,1}^{(0)} x + b^{(0)} \right), \quad (4.3)$$

for  $k = 1, \dots, N$ . In (4.1)–(4.3),  $\sigma(\cdot)$  denotes a nonlinear function known as the activation function,  $a_i^{(j)}$  is referred to as the activation of neuron  $i$  in layer  $j$ , and the parameters  $\theta := (w_{1,1}^{(0)}, \dots, w_{1,N}^{(L)}, b^{(0)}, \dots, b^{(L)})^T$  include the weights ( $w_{i,j}^{(l)}$ ) and biases ( $b^{(l)}$ ) of the NN. The parameters  $\theta$  are computed by minimizing a pre-defined loss function  $\mathcal{L}(\theta)$ :

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\theta). \quad (4.4)$$

A PINN is differentiated from a typical NN by the definition of the loss function  $\mathcal{L}(\theta)$ . Whereas a NN loss function measures the difference between the NN outputs and a corresponding set of solution data taken as the ground truth, a PINN does not require any such

<sup>2</sup>From this point further, we will use the term NN and PINN interchangeably, since our most general PINN formulation (4.6) includes a data loss term. Please see Remark 2 for a comment on this.

solution data. Instead, a PINN uses the residual of the PDEs it is attempting to solve as the loss function (in this case, equation (2.1)). This is achieved by first taking as inputs a set of collocation points  $x \in \Omega$ . In its most basic form, the loss function defining the PINN minimization problem is given by the residual loss, that is, by

$$\mathcal{L}(\theta) = \mathcal{L}_r(\theta) := \frac{1}{M} \sum_{i=1}^M (-\nu \nabla_x^2 \hat{u}(x_i, \theta) + \nabla_x \hat{u}(x_i, \theta) - 1)^2, \quad (4.5)$$

where the  $x_i \in \Omega$  for  $i = 1, \dots, M$  are the collocation points used to evaluate the residual loss and  $NN(x; \theta) = \hat{u}(x; \theta) \approx u(x)$  represents the PINN approximation of the solution as a function of  $x$  and the training parameters  $\theta$ . Once the loss function (4.5) is defined along with the PINN architecture (Figure 4.1), a gradient-based optimization algorithm is used to solve (4.4), with derivatives computed via automatic differentiation.

In the following subsections, we describe several PINN variations that allow for the incorporation of boundary conditions as well as available snapshot data. In this more general formulation, the loss being minimized takes the form

$$\mathcal{L}(\theta) = \alpha_r \mathcal{L}_r(\theta) + \alpha_b \mathcal{L}_b(\theta) + \alpha_d \mathcal{L}_d(\theta), \quad (4.6)$$

for some relaxation parameters  $\alpha_r, \alpha_b, \alpha_d \in \mathbb{R}$ , often normalized to sum to be between 0 and 1. In (4.6),  $\mathcal{L}_b(\theta)$  and  $\mathcal{L}_d(\theta)$  are the boundary and data losses, respectively. These terms are discussed in more detail below, in Sections 4.1 and 4.3, respectively.

**Remark 2.** While, in the original PINN paper by Raissi *et al.* [23], PINNs were presented as completely data-free, i.e.,  $\alpha_d = 0$  in (4.6), subsequent PINN formulations allowed the addition of the data-loss term  $\mathcal{L}_d(\theta)$  for cases where some training data (from high-fidelity simulations or experiments) are available. For more information, the interested reader is referred to [14, 9] and the references therein.

**4.1. Weak Dirichlet BCs (WDBCs).** The traditional way to impose boundary conditions in PINNs is through a weak formulation, via the loss function. To this effect, the loss function being minimized (4.6) takes the form

$$\mathcal{L}(\theta) = \alpha \mathcal{L}_r(\theta) + (1 - \alpha) \mathcal{L}_b(\theta), \quad (4.7)$$

where  $\alpha \in (0, 1]$ , and

$$\mathcal{L}_b(\theta) := \frac{1}{b} \sum_{i=1}^b (\hat{u}(x_i, \theta) - u(x_i))^2. \quad (4.8)$$

In (4.8), the points  $x_i \in \partial\Omega$  for  $i = 1, \dots, b$  are the coordinates of boundary points at which Dirichlet boundary conditions are prescribed. In the 1D domain depicted in Figure 3.1(a),  $x_i \in \{0, 1\}$  for the domain  $\Omega$ , whereas  $x_i \in \{0, \gamma_1\}$  and  $x_i \in \{\gamma_2, 1\}$  for the subdomains  $\Omega_1$  and  $\Omega_2$ , respectively. For 1D subdomains, the maximum number of boundary points for which the weak boundary loss is computed (within each subdomain) is thus  $b = 2$ .

**4.2. Strong Dirichlet BCs (SDBCs).** While the majority of PINNs in the literature employ a weak enforcement of the boundary conditions via a boundary loss term (4.8), this approach has some disadvantages. First, it is unclear *a priori* how to select the relaxation parameters  $\alpha_r$  and  $\alpha_b$  in (4.6); often this is done by trial and error, and the values require retuning when the problem setup is modified. Second, since the BCs in (4.6) are imposed

weakly, the learned solution may be inconsistent with the underlying BVP (2.1)–(2.2). Moreover, some recent work has demonstrated that the PINN optimization problem (4.6) may be extremely stiff, leading to a lack of convergence as a result of the residual and boundary losses competing with each other in the loss function [29, 27].

Since information within the Schwarz alternating method propagates through the DBCs imposed at the subdomain boundaries and past convergence analyses [17, 21, 22] of the method have assumed a strong implementation of these BCs, we are interested in being able to implement SDBC<sub>s</sub> within our PINNs for the BVP considered.

Motivated by earlier related work for other flavors of collocation methods (e.g., [8] for Bayesian Gaussian processes), recent years have seen the development of several approaches for imposing BCs strongly within a PINN or NN. In the FBPINN approach [20, 4], instead of defining a NN to directly approximate the solution as discussed above ( $NN(x; \theta) \approx u(x)$ ), an ansatz of the form

$$\hat{u}(x; \theta) = g(x) + \psi(x)NN(x; \theta) \approx u(x), \quad (4.9)$$

where the functions  $g(x)$  and  $\psi(x)$  are derived such that  $\hat{u}(x; \theta)$  in (4.9) satisfies the prescribed DBCs exactly. Since DBCs are imposed strongly in the NN, it is not necessary to include the boundary loss  $\mathcal{L}_b(\theta)$  in the loss function; hence, the loss function reduces to the residual loss, i.e., (4.5). In the case of complex, multi-dimensional domains, the functional form of  $g(x)$  may be rather complex, and will often depend on the distance of a point  $x$  from the boundary  $\partial\Omega$ , as discussed in [28]. An alternate approach that uses distance functions and geometry-aware trial functions within a Deep Ritz PINN (i.e., a PINN in which the residual loss is given in weak variational rather than strong form) is the work of Sukumar *et al.* [26].

In order to keep the discussion as general as possible for our 1D model problem (2.1), consider a generic overlapping decomposition of  $\Omega$  into  $n_D$  overlapping subdomains, so that  $\Omega = \cup_{i=1}^{n_D} \Omega_i$ , where  $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1})$ , for  $i = 1, \dots, n_D$ , where  $\gamma_0 = 0$  and  $\gamma_{2n_D-1} = 1$ . We begin by defining the following subdomain-local BVP for our targeted 1D advection-diffusion equation

$$-\nu \frac{\partial^2 u_i}{\partial x^2} + \frac{\partial u_i}{\partial x} - 1 = 0, \quad \text{in } \Omega_i := (\gamma_{2i-2}, \gamma_{2i-1}), \quad (4.10)$$

with boundary conditions

$$u_i(\gamma_{2i-2}) = g_{2i-2}, \quad u_i(\gamma_{2i-1}) = g_{2i-1}, \quad (4.11)$$

for  $g_i \in \mathbb{R}$ , with  $i = 1, \dots, n_D$ . Let  $NN_i(x; \theta)$  denote a NN trained to represent the solution in  $\Omega_i$ . The reader can observe that the following function is guaranteed to satisfy strongly the DBCs (4.11) on  $\partial\Omega_i$ :

$$\hat{u}_i(x; \theta) = v_i(x)NN_i(x; \theta) + \phi_i(x)g_{2i-2} + \psi_i(x)g_{2i-1}, \quad (4.12)$$

where  $v_i(x)$  is a function such that  $v(\gamma_{2i-2}) = v(\gamma_{2i-1}) = 0$ ,  $\phi_i(x)$  is a function such that  $\phi_i(\gamma_{2i-2}) = 1$  and  $\phi_i(\gamma_{2i-1}) = 0$ , and  $\psi_i(x)$  is a function such that  $\psi_i(\gamma_{2i-2}) = 0$  and  $\psi_i(\gamma_{2i-1}) = 1$ . In the present work, we employ the following expressions for these functions:

$$v_i(x) = \tanh(k(\gamma_{2i-1} - x)) \tanh(k(x - \gamma_{2i-2})), \quad (4.13)$$

$$\phi_i(x) = 10^{-10(x-\gamma_{2i-2})}, \quad (4.14)$$

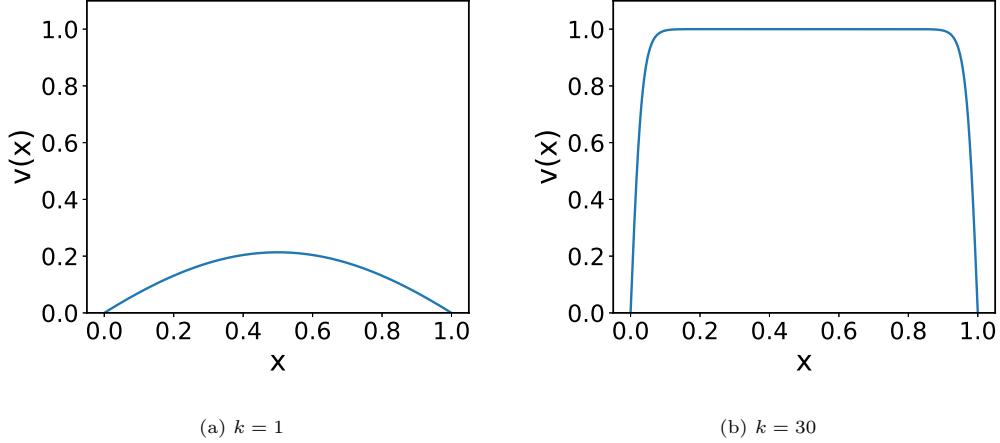


Fig. 4.2: Strong DBC scaling function (4.13) for two values of  $k$ :  $k = 1$  (left) and  $k = 30$  (right).

$$\psi_i(x) = 10^{10(x - \gamma_{2i-1})}, \quad (4.15)$$

for  $x \in (\gamma_{2i-2}, \gamma_{2i-1})$  and  $k > 0$ .

The scaling function (4.13) is plotted for two values of  $k$ ,  $k = 1$  and  $k = 30$ , in Figure 4.2. As  $k$  is increased,  $v_i(x)$  begins to resemble a step function on  $\Omega = (0, 1)$ . At first glance, it may seem as though employing a larger  $k$  in (4.13) should improve PINN convergence, as  $v_i(x) \approx 1$  in  $\Omega$  away from the boundary  $\partial\Omega$ , meaning that this scaling function minimally modifies the NN away from  $\partial\Omega$  (Figure 4.2(a)). However, our numerical experiments reveal that selecting higher values of  $k$ , e.g.,  $k = 30$  (Figure 4.2(b)), severely hinders the NN's convergence. We believe that this happens because  $v_i(x)$  exhibits sharp gradients when  $k \gg 1$ , requiring the NN to make rapid and substantial changes to the magnitudes of its outputs for a small subset of the spatial domain close to the boundaries, complicating the minimization of the loss function. For this reason, for the results presented in Section 6, a value of  $k = 1$  is used to define  $v_i(x)$  (4.13).

The additional scaling functions  $\phi_i(x)$  and  $\psi_i(x)$ , also for the interval  $\Omega_i = (0, 1)$ , are shown in Figure 4.3. We emphasize that, like the function  $v_i(x)$ , the inhomogeneous boundary condition functions  $\phi_i(x)$  and  $\psi_i(x)$  are not unique. While exploring alternate choices for these functions goes beyond the scope of the present paper, we remark that our preliminary numerical experiments suggested that a smoothly varying  $\phi_i(x)$  and  $\psi_i(x)$  function is in general more effective than a Dirac delta function, as it minimizes the presence of sharp gradients, which can hinder the convergence of a NN.

**4.3. Incorporation of data into a PINN.** While the appeal of PINNs is at least partly that solution data are not required for training, we remark that it is possible to incorporate available data into the NN by adding a so-called data loss term  $\mathcal{L}_d(\theta)$ , as in (4.6). This data loss term takes the form of

$$\mathcal{L}_d(\theta) := \frac{1}{d} \sum_{i=1}^d (\hat{u}(x_i; \theta) - u(x_i))^2, \quad (4.16)$$

where  $d$  is the number of collocation points used to evaluate the data loss, and  $u(x_i)$  is the snapshot data at  $x = x_i$ . While, as written, (4.16) compares the NN solution with

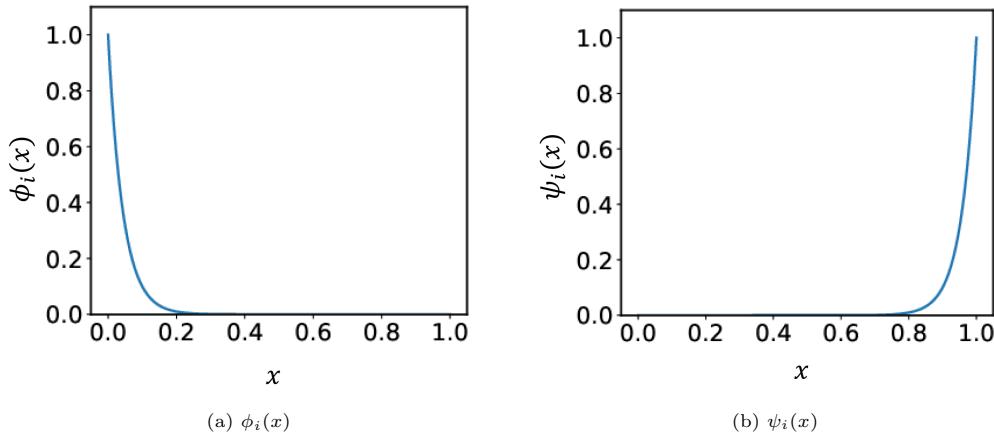


Fig. 4.3: Inhomogeneous boundary condition enforcement functions (4.14)  $\phi_i(x)$  and (4.15)  $\psi_i(x)$  for the interval  $\Omega_i = (0, 1)$ .

exactly one snapshot of the solution, additional snapshots can be obtained by generating solutions to the governing BVP (2.1)–(2.2) for different parameter numbers (e.g., different  $Pe$ ), different boundary conditions and/or different source terms.

**5. The Schwarz alternating method for PINN-PINN and PINN-FOM coupling.** Our goal in this paper is to extend the Schwarz alternating method, described earlier in Section 3, to PINN-PINN and PINN-FOM couplings, following an overlapping domain decomposition of the underlying spatial domain  $\Omega$ . Two coupling scenarios are possible:

- *Coupling Scenario 1.* Use DD and the Schwarz alternating method to facilitate training of PINNs (offline).
  - *Coupling Scenario 2.* Use DD and the Schwarz alternating method to couple pre-trained subdomain-local PINNs (online).

As discussed in Section 1, Coupling Scenario 1 has been considered in several recent works, most notably [15, 16, 10], whereas Coupling Scenario 2 was explored in [28]. Herein, attention is focused on Coupling Scenario 1. Specifically, we explore the hypothesis that, by creating and coupling (using the Schwarz alternating method) several subdomain-local PINNs with each other and/or with FOMs, it may be possible to accelerate PINN training. For the targeted advection-diffusion problem (2.1)–(2.2), we are particularly interested in the advection-dominated (high Péclet) regime, in which it is well-known that PINNs are particularly difficult to train [19]. We plan to explore Coupling Scenario 2 in a subsequent publication.

In order to develop a PINN-PINN coupling algorithm using the Schwarz alternating method, we combine ideas presented earlier in Sections 3 and 4. As in Section 4.2, suppose the subdomain  $\Omega = (0, 1)$  has been partitioned into  $n_D$  overlapping subdomains  $\Omega_i$ , so that  $\Omega = \cup_{i=1}^{n_D} \Omega_i$ , where  $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1}) \in \mathbb{R}$  for  $i = 1, \dots, n_D$ , with  $\gamma_0 = 0$  and  $\gamma_{n_D} = 1$ . Let  $\mathcal{L}_{r,i}$  and  $\mathcal{L}_{d,i}$  denote the residual and data losses, defined earlier in (4.7) and (4.16), respectively, but restricted to subdomain  $\Omega_i$ . Let  $NN_i(x; \theta)$  denote a NN within domain  $\Omega_i$ , and let  $\hat{u}_i(x; \theta) \approx u_i(x)$  be the approximation of the solution  $u(x)$  in  $\Omega_i$ . We provide some pseudo-code for our (offline) PINN training in Algorithm 1. The reader can observe that both the boundary loss term  $\mathcal{L}_{b,i}(\theta)$  and the approximate PINN solution  $\hat{u}_i(x)$  depend on

the type of DBC enforcement selected, namely the variable  $DBCtype$ , which can take on three values:

- Weak DBCs (WDBC), corresponding to weakly-imposed DBCs on all boundaries (section 4.2),
- Strong DBCs (SDBC), corresponding to strongly-imposed DBCs on all system and subdomain boundaries (Section 4.2),
- Mixed DBCs (MDBC), corresponding to strongly-imposed DBCs at the system boundaries and weakly-imposed DBCs at the Schwarz boundaries (Section 4.2 with  $\phi_i(x) = \psi_i(x) = 0$ ).

---

**Algorithm 1:** Alternating overlapping Schwarz PINN-PINN coupling algorithm

---

**Inputs:** overlapping DD of  $\Omega$  into  $n_D$  overlapping subdomains  $\Omega_i = (\gamma_{2i-2}, \gamma_{2i-1})$ ; relaxation parameters  $\alpha_r, \alpha_d \in [0, 1]$ ;  $DBCtype \in \{\text{WDBC, MDBC, SDBC}\}$ ; Schwarz convergence criteria.

---

```

Initialize  $\hat{u}_i(\gamma_{2i-1}, \theta) = 0$  for  $i = 1, \dots, n_D$ .
Set  $NN_{-i}(\cdot; \cdot) = NN_{n_D+1}(\cdot; \cdot) = 0$ .
while Schwarz method unconverged do
    for  $i = 1, \dots, n_D$  do
        Train PINN  $NN_i(x; \theta)$  in  $\Omega_i$  with loss
        
$$\mathcal{L}_i(\theta) := \alpha_r \mathcal{L}_{r,i}(\theta) + (1 - \alpha_r) \mathcal{L}_{b,i}(\theta) + \alpha_d \mathcal{L}_{d,i}(\theta),$$

        where
        
$$\mathcal{L}_{b,i}(\theta) = \begin{cases} \mathcal{L}_{b,i}^{sys}(\theta) + \mathcal{L}_{b,i}^{sch}(\theta), & \text{if } DBCtype = \text{WDBC}, \\ \mathcal{L}_{b,i}^{sch}(\theta), & \text{if } DBCtype = \text{MDBC}, \\ 0, & \text{if } DBCtype = \text{SDBC}, \end{cases}$$

        with
        
$$\mathcal{L}_{b,i}^{sys}(\theta) = \begin{cases} NN_i(0; \theta)^2, & \text{if } i = 1, \\ NN_i(1; \theta)^2, & \text{if } i = n_D, \\ 0, & \text{otherwise,} \end{cases}$$

        and
        
$$\mathcal{L}_{b,i}^{sch}(\theta) = \begin{cases} (NN_i(\gamma_{2i-1}; \theta) - \hat{u}_{i+1}(\gamma_{2i-1}; \theta))^2, & \text{if } i = 1, \\ (NN_i(\gamma_{2i-2}; \theta) - \hat{u}_{i-1}(\gamma_{2i-2}; \theta))^2, & \text{if } i = n_D, \\ (NN_i(\gamma_{2i-2}; \theta) - \hat{u}_{i-1}(\gamma_{2i-2}; \theta))^2 + \\ (NN_i(\gamma_{2i-1}; \theta) - \hat{u}_{i+1}(\gamma_{2i-1}; \theta))^2, & \text{otherwise.} \end{cases}$$

        Set
        
$$\hat{u}_i(x; \theta) = \begin{cases} NN_i(x; \theta), & \text{if } DBCtype = \text{WDBC}, \\ v_i(x)NN_i(x; \theta), & \text{if } DBCtype = \text{MDBC}, \\ v_i(x)NN_i(x; \theta) + \phi_i(x)\hat{u}_{i-1}(\gamma_{2i-2}; \theta) + \\ \psi_i(x)\hat{u}_{i+1}(\gamma_{2i-1}; \theta), & \text{if } DBCtype = \text{SDBC}, \end{cases}$$

        where  $v_i(x)$ ,  $\phi_i(x)$  and  $\psi_i(x)$  are given by (4.13), (4.14) and (4.15), respectively.
        Interpolate  $\hat{u}_i(x; \theta)$  onto  $x = \gamma_{2i}$ .
        Check convergence criteria for Schwarz iteration. Exit if converged.
    end
end

```

---

It is straightforward to modify Algorithm 1 to the case where a FOM is employed in one or more subdomains  $\Omega_i$ . While the details are omitted here for the sake of brevity, some numerical results for a PINN-FOM coupling are presented and discussed in Section 6.4.

**6. Numerical Results.** In this section, we perform some numerical experiments aimed at understanding to what extent DD combined with the Schwarz alternating method can assist with the training of PINNs for the target advection-diffusion BVP (2.1)–(2.2).

In order to perform the studies, we created a Python code<sup>3</sup> which invoked the `TensorFlow` library [1] for PINN training. All of the NNs evaluated use the same hyperparameters. Each network had a 1D input layer to receive the spatial data,  $x$ , two hidden layers each with 20 nodes per layer, and a 1D output layer for the approximations of  $u(x)$ , similar to the NN architecture shown in Figure 4.1. Each hidden layer employed the swish activation function, given by

$$\sigma(z) := \frac{z}{1 + e^{-\mu z}}, \quad (6.1)$$

in (4.1)–(4.3), for  $\mu = 1$ . When calculating the loss, the values of the relaxation parameters in (4.6) were  $\alpha_r = 0.25$ ,

$$\alpha_b = \begin{cases} 1 - \alpha_r, & \text{if using WDBCs or MDBCs,} \\ 0, & \text{if using SDBCs,} \end{cases} \quad \alpha_d = \begin{cases} 1 - \alpha_r, & \text{if using data loss,} \\ 0, & \text{otherwise,} \end{cases} \quad (6.2)$$

in all cases. The input data were not normalized, as they were already on the interval  $[0, 1]$ . We chose to use the Adam optimizer for all NNs, with a constant learning rate of 0.001. The number of training epochs for each NN per Schwarz iteration was 1024. For all experiments,  $M = 1024$  quasi-random uniform collocation points were selected within the full domain  $\Omega = (0, 1)$ , subdivided among the subdomains  $\Omega_i$ , as inputs to evaluate the residual loss (4.5). For experiments in which PINN-FOM couplings were evaluated (Section 6.4), the FOM was generated by discretizing the governing PDE (2.1) using a second-order accurate finite difference approach with a mesh resolution of  $h = \frac{1}{1024}$ . This finite difference solution was also considered the ground truth against which all PINN approximations were compared in relative error calculations. For experiments in which the data loss term (4.16) was included in the NN minimization problem, all FOM representations of this system were created using a second-order accurate backward finite difference approach with 1024 evenly spaced points on the domain  $x \in (0, 1)$ . These snapshot data were then transferred to each subdomain  $\Omega_i$  by interpolating the FOM solution to the collocation points in  $\Omega_i$ . The convergence criteria required both the Schwarz relative error to drop below a tolerance of  $\delta = 0.001$ , and the  $L_2$  relative error in the NN approximation with respect to the reference FOM solution to drop below a tolerance of 0.005. We allowed each model a maximum of 100 Schwarz iterations before cutting off training and declaring the model non-converged for a given problem case.

The study summarized herein is aimed at understanding the relative impact of the following PINN- and coupling-related parameters on both the accuracy and the efficiency of the PINN training process:

- the number of subdomains,  $n_D$ ,
- the size of the overlap region,
- the type of DBC enforcement in the PINN (WDBC, MDBC or SDBC), and
- the impact of including the data loss term  $\mathcal{L}_d(\theta)$  (4.16) in the loss function being minimized.

**6.1. PINN-PINN coupling: parameter sweep study.** We begin by investigating the impact of the size of the overlap region and the number of Schwarz subdomains  $n_D$  on both the accuracy and the efficiency of the resulting PINN-PINN coupling. To do this, we first

---

<sup>3</sup>This code is available on github at the following URL: <https://github.com/ikalash/Schwarz-4-Multiscale>.

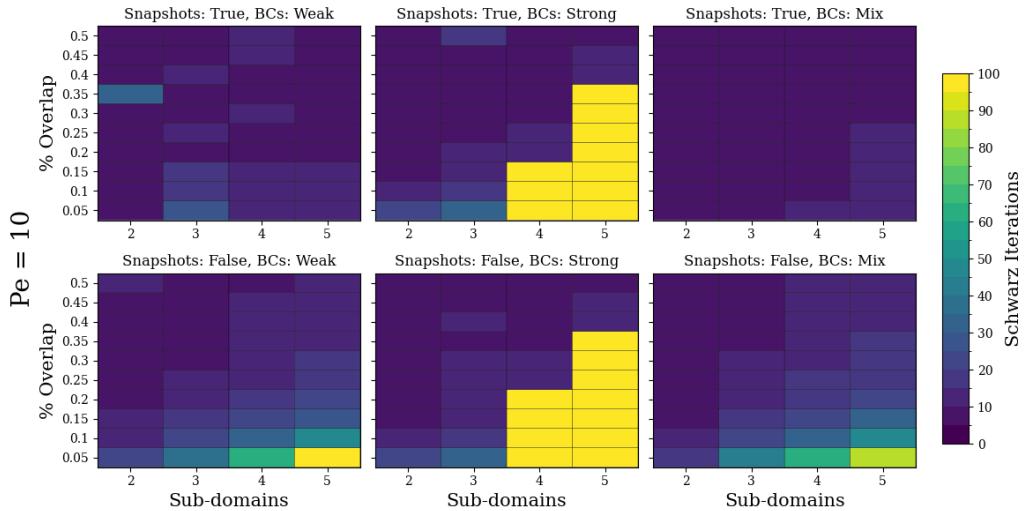


Fig. 6.1: Schwarz iterations required to achieve convergence for various  $n_D$  and percentage overlap, for  $Pe = 10$ . Each frame corresponds to various boundary conditions and data loss combinations.

perform a parameter sweep study in which we vary  $n_D$  between 2 and 5, and the percentage overlap of the subdomains between 5-50% in increments of 5%. To determine the boundaries of each subdomain, we calculate the size of each subdomain,  $S_D$ , and the size of the overlapping regions,  $S_O$ , as functions of the number of subdomains and the percentage overlap,

$$S_D = \frac{1}{n_D(1 - p_O) + p_O}, \quad (6.3)$$

$$S_O = p_O S_D, \quad (6.4)$$

where  $p_O$  is the percentage overlap expressed as a value between 0 and 1. Knowing  $S_D$  and  $S_O$ , the boundary points,  $\gamma_{2i-2}$  and  $\gamma_{2i-1}$ , which define each subdomain,  $\Omega_i$ , can then be determined by

$$\begin{aligned} \gamma_{2i-2} &= (i-1)(S_D - S_O), \quad \text{for } i = 1, \dots, n_D, \\ \gamma_{2i-1} &= \gamma_{2i-2} + S_D. \end{aligned} \quad (6.5)$$

We also investigate the impact of using SDBC vs. MDBCs vs. WDBC on the system and Schwarz boundaries, as well as the impact of including the data loss (4.16) in the loss function being minimized. We consider two Péclet numbers, namely  $Pe = 10$  and  $Pe = 100$ , in this study. Higher Péclet number problems are considered later in Sections 6.2 and 6.4.

To measure training efficiency and performance, both the number of Schwarz iterations required to satisfy the convergence criteria (Figures 6.1 and 6.3) and the final  $L_2$  relative error with respect to the FOM solution averaged across all subdomain models (Figures 6.2 and 6.4) are recorded. For the case of  $Pe = 10$ , Figures 6.1 and 6.2 exhibit a general decrease in the number of iterations required for convergence and average  $L_2$  relative error with increased percentage overlap. This is particularly true when SDBC are enforced, as many cases with low overlap percentages fail to converge entirely (the cells marked in

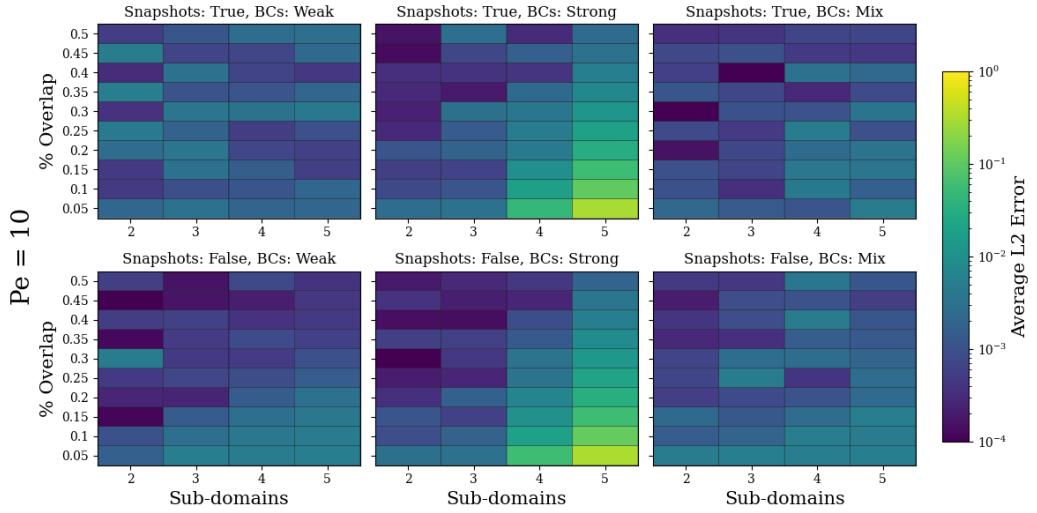


Fig. 6.2: Average  $L_2$  relative error at the end of the Schwarz iteration for various  $n_D$  and percentage overlap, for  $Pe = 10$ . Each frame corresponds to various boundary conditions and data loss combinations.

yellow in Figure 6.1), but succeed with increased overlap. This follows the expected trend demonstrated in a number of references, including [17, 21, 22, 16], wherein the number of Schwarz iterations (which typically correlates with the CPU time) is inversely proportional to the size of the overlap region. Conversely, increasing the number of subdomains seems to increase the required number of Schwarz iterations and the average  $L_2$  error. This effect is, again, particularly clear for the case of SDBC enforcement. For the case of  $Pe = 100$ , however, Figures 6.3 and 6.4 do not exhibit such clear relationships. Varying the number of subdomains and the percentage overlap appears to have a largely random effect, with many cases simply failing to converge within 100 Schwarz iterations (see the cells marked in yellow in Figure 6.3). This is a stark example of the historic difficulty in constructing robust PINNs for fluid systems undergoing strong advection (see [19] and the references therein for more details).

Noticeable trends with respect to the boundary condition enforcement and data loss inclusion are also observed in our results. Curiously, SDDBCs perform categorically worse than WDBCs or MDCBs in almost all cases. As noted previously, this is particularly true for the case of  $Pe = 10$  with a larger number of subdomains and lower percentage overlap, for which many cases failed to converge. This conclusion generally extends to the case of  $Pe = 100$ , with the exception of WDBCs without a data loss term, for which nearly all cases failed to converge. Strikingly, the inclusion of a data loss term with WDBCs and MDCBs mostly eliminates this convergence problem. In general, for both  $Pe = 10$  and  $Pe = 100$ , inclusion of the data loss term improves efficiency and accuracy. This is largely unsurprising, as one expects providing additional training data to improve the reconstructive accuracy of the neural network. The fact that including the data loss term in the PINN loss function being minimized has the largest improvement for the models with WDBCs and MDCBs is consistent with the literature [6, 24], which reports that a PINN optimization problem is not guaranteed to be well-posed with a weak implementation of the boundary conditions. We hypothesize that some of this ill-posedness is being encountered by our PINNs with WDBCs and MDCBs, and that the inclusion of the data loss term helps to regularize the optimization

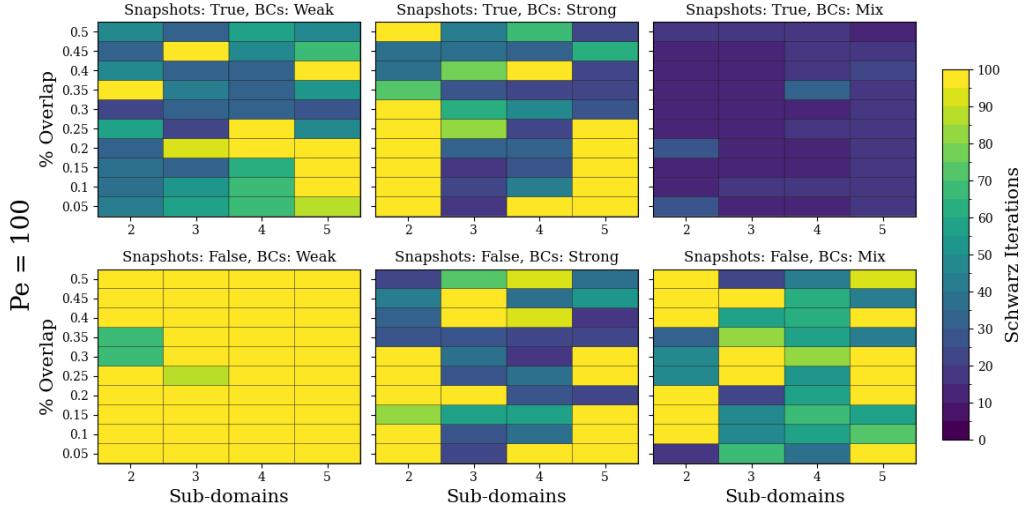


Fig. 6.3: Schwarz iterations required to achieve convergence for various  $n_D$  and percentage overlap, for  $Pe = 100$ . Each frame corresponds to various boundary conditions and data loss combinations.

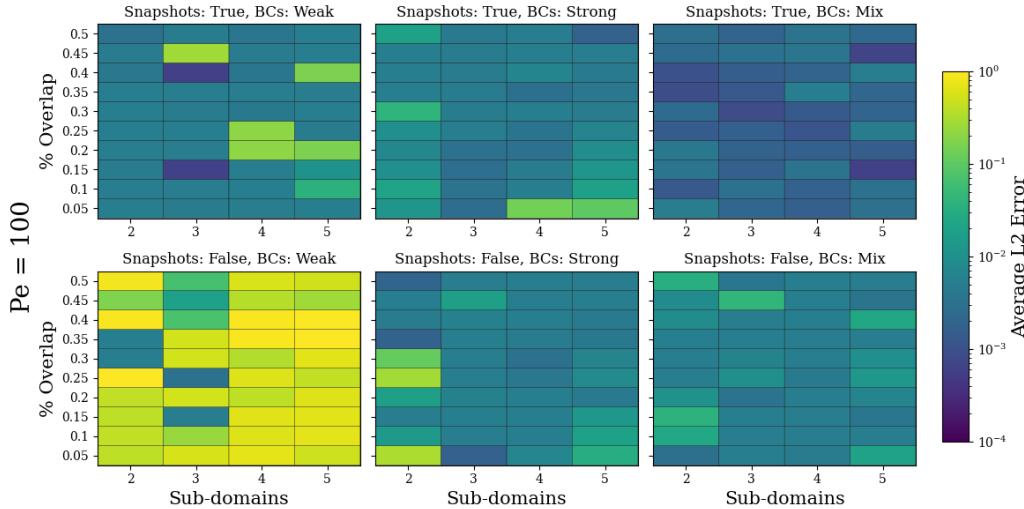


Fig. 6.4: Average  $L_2$  relative error at the end of the Schwarz iteration for various  $n_D$  and percentage overlap, for  $Pe = 100$ . Each frame corresponds to various boundary conditions and data loss combinations.

problem in each of these cases.

To examine the above results in greater detail, Pareto plots for a single intermediate percentage overlap of 35% display the average  $L_2$  error with respect to the number of Schwarz iterations in Figure 6.5. The various symbol colors indicate the number of subdomains, while the symbol shapes indicate the boundary condition enforcement type and inclusion of the data loss. The apparent error threshold for  $Pe = 100$  is an artifact of including the relative error in the convergence criteria. This broadly indicates the increased difficulty of efficient

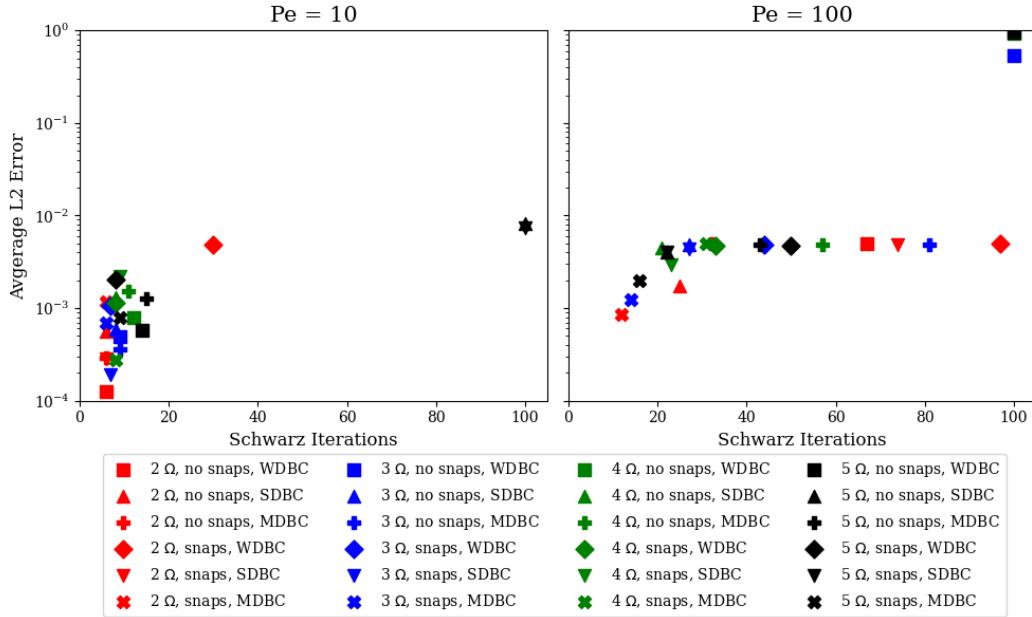


Fig. 6.5: Pareto plots depicting  $L_2$  relative error vs. Schwarz iterations for PINN-PINN couplings having different values of  $n_D$  and different DBC enforcement types, with and without a data loss term, for  $Pe = 10$  (left) and  $Pe = 100$  (right).

PINN training at higher  $Pe$ , where the predicted solution is slow to converge to the true solution, hence satisfying the Schwarz convergence criterion and yet struggling to satisfy the relative error criterion. These plots also draw attention to the relative success of the MDBCs in combination with the data loss term, indicated by an “ $\times$ ” symbol. All such cases succeeded in converging in fewer than 40 iterations.

**6.2. PINN-PINN coupling: the impact of different DBC implementations for higher  $Pe$  numbers.** Following the systematic parameter sweep studies described in the previous section, we performed a manual study exploring in more detail the impact of the boundary condition treatment on model convergence at high Péclet numbers. In this investigation, snapshot data are included in the loss function being minimized, and the relaxation term is set to  $\alpha_r = 0.2$ . For these trials, the Péclet number is fixed at  $Pe = 150$ , the number of subdomains is  $n_D = 3$ , and the percentage overlap is 20%. Attempts to obtain a PINN-PINN coupled solution or a single-domain PINN solution for  $Pe > 150$  were unsuccessful, as discussed in more detail in Section 6.3. As shown in Section 6.4, it is possible to circumvent this difficulty by performing a PINN-FOM coupling using the Schwarz alternating method.

Sample snapshots throughout model training are displayed for WDBCs in Figure 6.6, for MDBC in Figure 6.7, and for SDBC in Figure 6.8. The lower right panel in each figure represents the final solution upon achieving convergence. It is interesting to observe that, while all DBC enforcement strategies give rise to couplings that converged in fewer than 100 Schwarz iterations, the manner in which the solutions progress is very different. When imposing WDBCs, the subdomain-local PINNs very slowly converge to the true solution, and require additional iterations to closely match the system boundary conditions (Figure 6.6). In contrast, when imposing MDBC, the subdomain-local PINNs very quickly approach a solution that resembles the FOM solution within the first 20 iterations, satisfying the

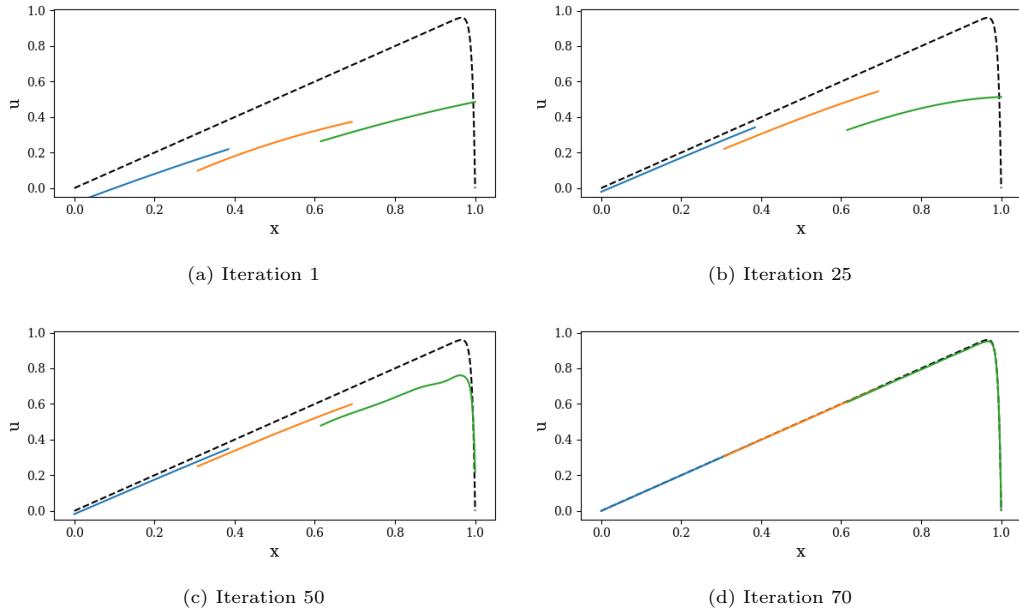


Fig. 6.6:  $Pe = 150$  alternating Schwarz-based converged solutions for a three subdomain all-PINN coupling with WDBCs for several Schwarz iterations. The method converged in 70 Schwarz iterations.

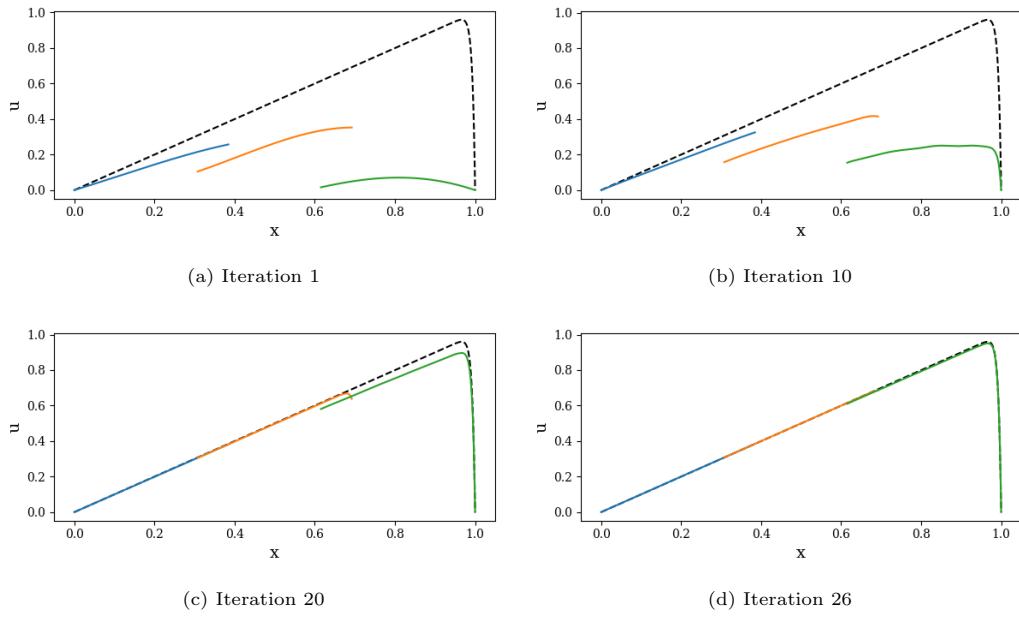


Fig. 6.7:  $Pe = 150$  alternating Schwarz-based converged solutions for a three subdomain all-PINN coupling with MDBCs (strong enforcement at system boundaries and weak enforcement at Schwarz boundaries) for several Schwarz iterations. The method converged in 26 Schwarz iterations.

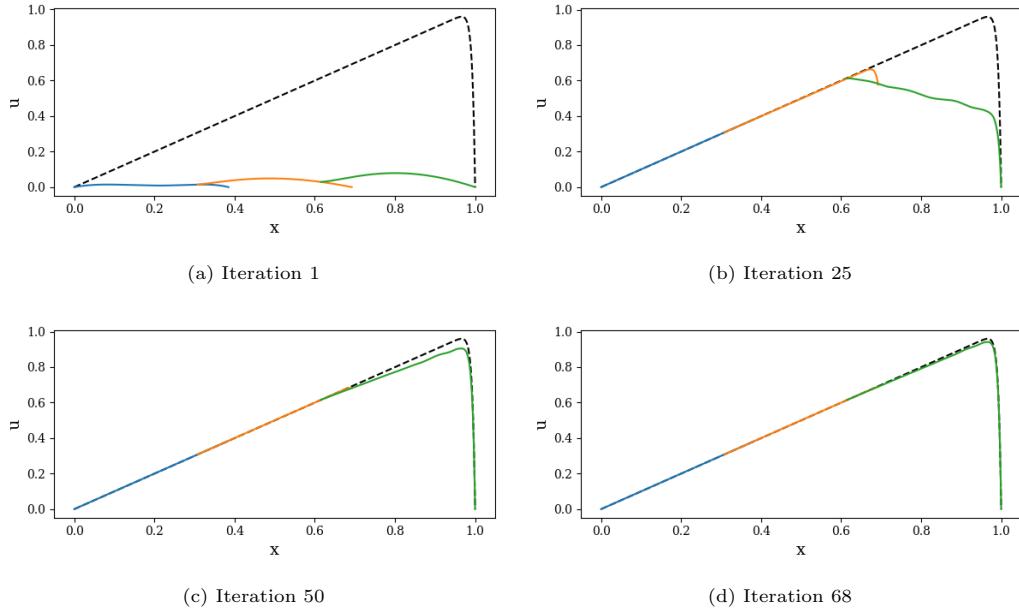


Fig. 6.8:  $Pe = 150$  alternating Schwarz-based converged solutions for a three subdomain all-PINN coupling with SDBC<sub>s</sub> for several Schwarz iterations. The method converged in 68 Schwarz iterations.

system boundary conditions throughout convergence (Figure 6.7). Unlike the WDBC and MDBC cases, which display disjoint solutions between the subdomains, the enforcement of SDBC<sub>s</sub> (Figure 6.8) generates a relatively smooth solution throughout the duration of training. Similar to the MDBC case, the SDBC enforcement quickly approaches a reasonable solution, but requires a significant number of additional iterations to meet the convergence criteria.

Pointwise relative error plots in Fig. 6.9 visualize the spatial accuracy induced by each boundary condition enforcement scheme. The error plots are trimmed to  $x \in [0.025, 0.975]$  to avoid division by very small numbers in the error calculations. As one might expect, the WDBC and MDBC schemes do not guarantee that the error distribution is continuous between subdomains, as evidenced by significant jumps in relative error from one subdomain to the next. Conversely, the SDBC enforcement ensures that the predicted solutions match at the subdomain interfaces, resulting in a relatively smooth error profile. Also notable is the relatively high error near  $x = 1$  when SDBC<sub>s</sub> are enforced, an artifact of the slow convergence of the solution in this region observed in Fig. 6.8. The pointwise relative error remains below 2% in each case, and is ultimately controlled by the solution error tolerance used to determine convergence.

While all DBC enforcement methods investigated here lead to comparable results when considering Schwarz iteration counts alone, it can be argued that the convergence trajectory taken by the MDBC and SDBC variants are preferable. These models rapidly achieve a plausible solution, but struggle to fall strictly below the relative error convergence threshold. It is possible that a hybrid DBC enforcement strategy, in which one first applies MDBC<sub>s</sub>/SDBC<sub>s</sub> for a number of Schwarz iterations, followed by WDBC<sub>s</sub> in the subsequent Schwarz iterations, may give the most desirably convergence result, especially for higher Péclet numbers.

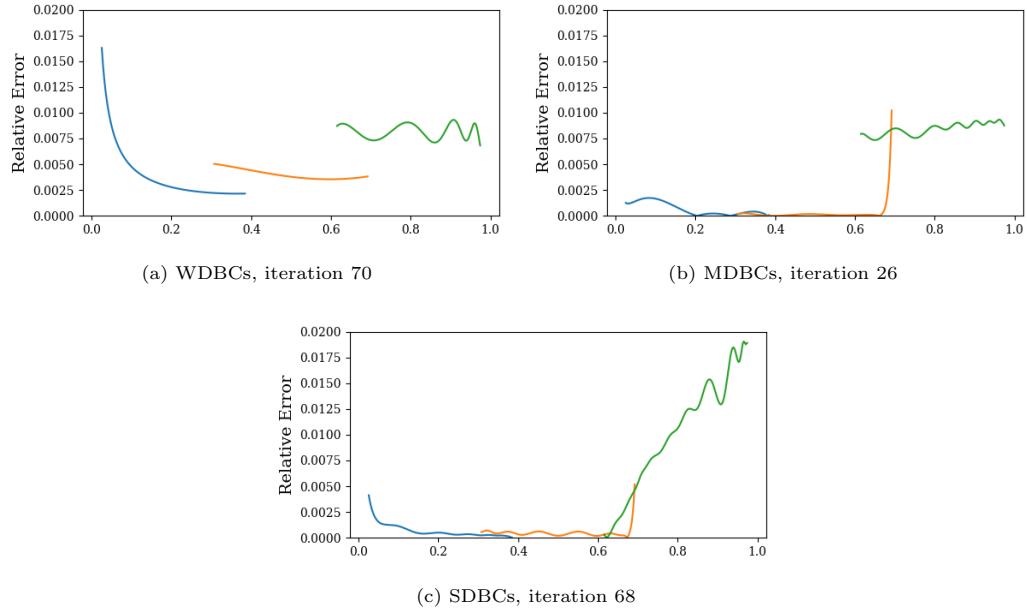


Fig. 6.9: Pointwise relative error in each subdomain for final  $Pe = 150$  converged solutions displayed in Figs. 6.6, 6.7, and 6.8, in color and measured against the left  $y$ -axis. All couplings produced a solution with a relative error of  $< 2\%$  with respect to the exact solution.

**6.3. PINN-PINN coupling vs. single-domain PINN.** One major question that has not yet been addressed is whether the proposed multi-domain coupling strategy for PINNs is actually beneficial for training such models in the advection-dominated regime when compared to a single-domain PINN having an equivalent architecture. To answer this question, we constructed a single-domain PINN for both forms of DBC enforcement and compared its convergence efficiency with the PINN-PINN couplings described above. Results were, however, largely inconclusive due to an observed extreme sensitivity to the model parameterization and initialization at high Péclet number, as previously displayed in Figure 6.3. We instead draw attention to broad trends. In general, strong enforcement of the system boundary conditions in the single-domain case resulted in faster convergence than those cases for which the boundary conditions were weakly enforced, in which case many models failed to converge entirely. For these high Péclet number systems, multi-domain PINNs with SDBC similarly tended to converge faster than equivalent models with MDBC or WDBC, though they did not, on average, appreciably improve training efficiency over their single-domain SDBC counterparts. Although this appears to be a negative result for the proposed coupling strategy as a means to accelerate PINN training, we remark that it is likely possible to improve our PINN-PINN coupling results by introducing additional parallelism into the Schwarz iteration by employing the additive variant of the Schwarz alternating method (Remark 1), and/or alternating WDBC and SDBC enforcements during the training process, as suggested at the end of Section 6.2. It may also be possible to attain improved results through PINN formulations especially designed for problems with a slowly decaying Kolmogorov  $n$ -width, such as the methods proposed in [19] and the references therein.

**6.4. PINN-FOM coupling.** A coupling strategy enabled by the Schwarz alternating method but not yet investigated is the coupling of PINNs with high-fidelity FOMs. Our numerical results reveal for this coupling case that it is possible to obtain a convergent coupled model for arbitrarily-high Péclet numbers in as few as 10 Schwarz iterations by coupling a PINN to a sufficiently-accurate finite difference model used to represent the sharp gradient within the boundary layer that forms. Consider a DD of  $\Omega = \Omega_1 \cup \Omega_2$  into  $\Omega_1 = (0, \gamma_1)$  and  $\Omega_2 = (\gamma_2, 1)$ , where  $\gamma_1$  and  $\gamma_2$  are calculated using (6.5) with  $n_D = 2$  and  $p_O = 0.1$ . Figure 6.10 shows the result of a PINN-FOM coupling in which a PINN is specified in  $\Omega_1$ , and subsequently coupled to a finite difference model with mesh resolution  $h = \frac{1}{1024}$  in  $\Omega_2$ . For this case, is possible to achieve convergence in just 10 Schwarz iterations for a Péclet number of  $Pe = 1 \times 10^6$ . We find that the type of DBC implementation (WDBC, MDBC or SDBC) has little effect on the accuracy and efficiency of the method.

The more interesting possibility that the PINN-FOM coupling strategy presents is the idea of pre-training several subdomain PINNs for different sections of a problem using FOM couplings on either side of the PINN. Thereafter, these pre-trained PINNs could be coupled online and may be capable of modeling more difficult problem cases than what is achievable with PINN-PINN coupled training, as discussed briefly in Section 6. We note, however, that this remains speculative, as we have not yet tested such a strategy.

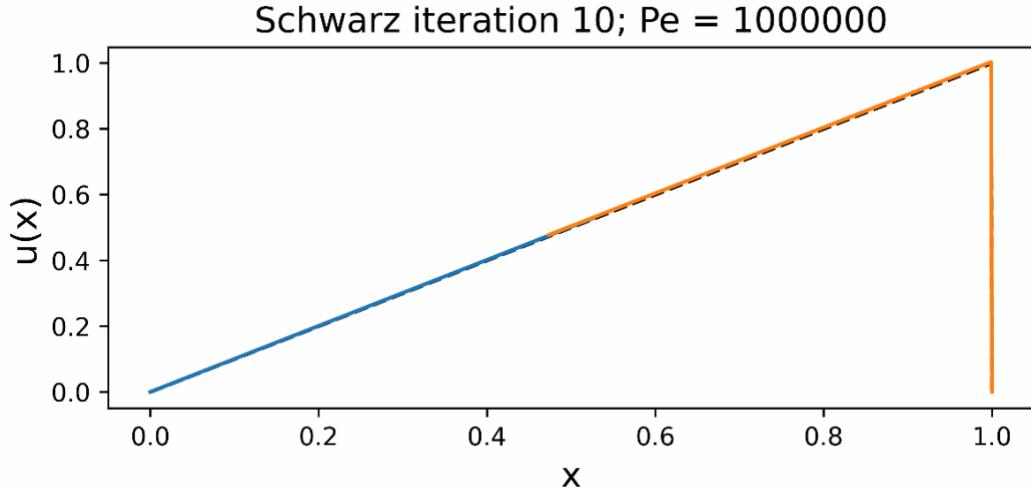


Fig. 6.10: Converged PINN-FOM coupled model with two subdomains and 10% subdomain overlap for the advection-diffusion equation with  $Pe = 1 \times 10^6$ . This model used WDBCs. The black dashed line indicates the FOM solution, the blue line indicates the PINN subdomain model solution, and the orange line indicates the FOM subdomain model solution.

**7. Conclusions.** In this paper, we explored the use of the Schwarz alternating method as a means to couple PINNs with each other and with conventional discretizations, following a decomposition of the spatial domain into overlapping subdomains. The bulk of our attention was focused on determining whether domain decomposition and the Schwarz alternating method can expedite the offline training stage of PINNs, to facilitate training these models in the slowly-decaying Kolmogorov  $n$ -width regime (referred to as *Coupling Scenario 1* herein). We explored three approaches for imposing Dirichlet boundary conditions within the overlapping Schwarz formulation at the system as well as the Schwarz boundaries: (i) a traditional weak formulation, in which a boundary loss term is added to the PINN loss

function being minimized (termed WDBC), (ii) a strong formulation, motivated by the FBPINN literature [20, 4], in which a transformation is applied to the neural network to ensure the solution satisfies the prescribed DBCs strogly (termed SDBC), and (iii) a “mixed” formulation, in which the system DBCs are imposed strongly whereas the Schwarz DBCs are imposed weakly via the PINN loss function (termed MDBC). We also explored the effect of incorporating a data loss term within the PINN loss function being minimized.

We evaluated the proposed coupling approach on a canonical 1D advection-diffusion problem, in which sharp gradients that are a function of the Péclet number ( $Pe$ ) form near the right boundary. For  $Pe = 10$  and  $Pe = 100$ , we performed a parameter sweep study in which we varied the number of subdomains, the size of the overlap region and the DBC implementation method. The results indicate that at lower  $Pe$ , the domain decomposition has a strong effect on model training efficiency and accuracy, with larger overlap regions and a smaller number of domains tending to results in the fastest convergence to the true solution. This is largely consistent with the traditional Schwarz decomposition literature. At higher  $Pe$ , the decomposition structure has a less consistent effect. In almost all cases, however, including the data loss term resulted in a significant performance improvement. Curiously, the SDBC implementation of the boundary conditions did not always lead to the fastest convergence of the Schwarz alternating method, as hypothesized *a priori*; the Pareto plot analysis reveals that the MDBC implementation leads to the best results in terms of accuracy and efficiency. While it is not clear from these results that Schwarz-based coupling of subdomain-local PINNs has an improvement on PINN training, especially at higher Péclet numbers, we demonstrate that PINN training *can* be improved through a PINN-FOM coupling enabled by the Schwarz alternating method. With this hybrid PINN-FOM coupling approach, it is possible to perform PINN training in as few as 10 Schwarz iterations for Péclet numbers as high as  $1.0 \times 10^6$ .

The initial exploration discussed in this paper has motivated several potential follow-on research efforts, itemized below.

- *Repeating the WDBC analysis with asymptotically-optimal norms and weights derived by Huynh, Meissner et al. [18, 12].* These weights and norms were derived by making analogies between PINNs and the Least Squares Finite Element Method [3].
- *Extending the formulation to non-overlapping subdomains.* The coupling of non-overlapping subdomains is important for multi-physics and multi-material problems, and requires the use of alternating Dirichlet-Neumann or Robin-Robin boundary conditions [7, 2].
- *Extending the formulation to multi-dimensional and time-dependent problems.* In multiple spatial dimensions, imposing Dirichlet BCs strongly is more challenging and will likely require a distance function-based approach [26, 28].
- *Prototyping of an additive Schwarz variant to improve efficiency.* In the additive Schwarz implementation, it is possible to solve multiple subdomains in parallel, rather than contemporaneously, which can lead to greater CPU-time savings [15, 16, 7].
- *Considering alternate convergence criteria for the Schwarz alternating method.* Combining the Schwarz convergence and  $L_2$  relative error in the convergence criteria may be sub-optimal, and alternative measures may be considered to improve predictive accuracy.
- *Devising a “hybrid” boundary condition enforcement strategy that switches between WDBCs and SDBCs.* The results presented in this paper suggest that such a strategy has the potential to improve accuracy and efficiency.
- *Exploring fully Coupling Scenario 2, in which pre-trained subdomain-local PINNs*

are coupled using the Schwarz alternating method. This work would leverage ideas from Wang *et al.* [28], in which a similar coupling scenario is investigated.

**Acknowledgement.** Support for this work was received through Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) program. The writing of this manuscript was funded in part by the second author's (Irina Tezaur's) Presidential Early Career Award for Scientists and Engineers (PECASE). 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. The authors wish to thank Alejandro Mota and Daria Koliesnikova for engaging in enlightening discussions related to details pertaining to the Schwarz alternating method for domain decomposition-based coupling, and Mamikon Gulian for offering invaluable insight into PINNs.

## REFERENCES

- [1] M. ABADI, P. BARHAM, J. CHEN, Z. CHEN, A. DAVIS, J. DEAN, M. DEVIN, S. GHEMAWAT, G. IRVING, M. ISARD, ET AL., *Tensorflow: A system for large-scale machine learning*, in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.
- [2] J. BARNETT, I. TEZAUR, AND A. MOTA, *The Schwarz alternating method for the seamless coupling of nonlinear reduced order models and full order models*. arXiv pre-print, math.NA, 2210.12551, 2022.
- [3] P. BOCHEV AND M. GUNZBURGER, *Least-squares finite element methods*, Springer Science and Business Media, vol. 166, 2009.
- [4] V. DOLEAN, A. HEINLEIN, S. MISHRA, AND B. MOSELEY, *Finite basis physics-informed neural networks as a Schwarz domain decomposition method*, 2023.
- [5] W. E AND B. YU, *The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, 2017.
- [6] S. A. FAROUGHI, N. PAWAR, C. FERNANDES, M. RAISI, S. DAS, N. K. KALANTARI, AND S. K. MAHJOUR, *Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing*, 2023.
- [7] M. J. GANDER, *Schwarz methods over the course of time*., ETNA. Electronic Transactions on Numerical Analysis [electronic only], 31 (2008), pp. 228–255.
- [8] T. GRAEPEL, *Solving noisy linear operator equations by gaussian processes: Application to ordinary and partial differential equations*, in Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML'03, AAAI Press, 2003, p. 234–241.
- [9] Z. HAO, S. LIU, Y. ZHANG, C. YING, Y. FENG, H. SU, AND J. ZHU, *Physics-informed machine learning: A survey on problems, methods and applications*, 2023.
- [10] A. HEINLEIN, A. KLAWONN, M. LANSER, AND J. WEBER, *Combining Machine Learning and Domain Decomposition Methods – A Review*. Universitat zu Köln Technical Report Series, Center for Data and Simulation Science. Technical Report ID: CDS-2020-9, 2020.
- [11] 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, 2021.
- [12] E. HUYNH, T. MEISSNER, AND P. BOCHEV, *A preliminary study for obtaining inverse Sobolev-type inequalities for tanh neural networks*. to appear in in Computer Science Research Institute Summer Proceedings 2023, 2023.
- [13] D. KOLIESNIKOVA, A. MOTA, I. TEZAUR, AND J. HOY, *The Schwarz alternating method for contact mechanics*. submitted to Int. J. Numer. Meth. Engng. (under review), 2023.
- [14] Z. K. LAWAL, H. YASSIN, D. T. C. LAI, AND A. CHE IDRIS, *Physics-Informed Neural Network (PINN) Evolution and Beyond: A Systematic Literature Review and Bibliometric Analysis*, Big Data and Cognitive Computing, 6 (2022), p. 140.
- [15] 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.
- [16] W. LI, X. XIANG, AND Y. XU, *Deep Domain Decomposition Method: Elliptic Problems*, Proceedings of Machine Learning Research, 107 (2020), pp. 269–286.
- [17] P. LIONS, *On the Schwarz alternating method I*. in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, 1988.

- [18] T. MEISSNER, E. HUYNH, P. KUBERRY, AND P. BOCHEV, *PINNs for Stokes: an elliptic regularity approach.* to appear in Computer Science Research Institute Summer Proceedings 2023, 2023.
- [19] R. MOJGANI, M. BALAJEWICZ, AND P. HASSANZADEH, *Kolmogorov  $n$ -width and Lagrangian physics-informed neural networks: A causality-conforming manifold for convection-dominated PDEs,* Computer Methods in Applied Mechanics and Engineering, 404 (2023), p. 115810.
- [20] B. MOSELEY, A. MARKAHM, AND T. NISSEN-MEYER, *Finite basis physics-informed neural networks (FBPINNs): a scalable domain decompositin approach for solving differential equations,* Advances in Computational Mathematics, 69 (2023).
- [21] 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.
- [22] A. MOTA, I. TEZAUR, AND G. PHILIPOT, *The Schwarz alternating method for dynamic solid mechanics,* Int. J. Numer. Meth. Engng, (2022), pp. 1–36.
- [23] M. RAISI, P. PERDIKARIS, AND G. KARNIADAKIS, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,* Journal of Computational Physics, 378 (2019), pp. 686–707.
- [24] C. RAO, H. SUN, AND Y. LIU, *Hard encoding of physics for learning spatiotemporal dynamics,* 2021.
- [25] H. A. SCHWARZ, *Ueber einen Grenzübergang durch alternirendes Verfahren,* Zürcher u. Furrer, 1870.
- [26] N. SUKUMAR AND A. SRIVASTAVA, *Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks,* Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114333.
- [27] L. SUN, H. GAO, S. PAN, AND J.-X. WANG, *Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data,* Computer Methods in Applied Mechanics and Engineering, 361 (2020), p. 112732.
- [28] H. WANG, R. PLANAS, A. CHANDRAMOLISHWARAN, AND R. BOSTANABAD, *Mosaic flows: A transferable deep learning framework for solving PDEs on unseen domains,* Computer Methods in Applied Mechanics and Engineering, 389 (2022), p. 114424.
- [29] S. WANG, Y. TENG, AND P. PERDIKARIS, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks,* SIAM Journal on Scientific Computing, 43 (2021), pp. A3055–A3081.

## MACHINE-LEARNED POTENTIAL DEVELOPMENT FOR HAFNIUM CERAMICS IN EXTREME CONDITIONS

VAYLE A. VERA CRUZ\* AND EMBER L. SIKORSKI†

### Abstract.

Hafnium carbide (HfC) is well known for its high melting temperature, even among the group IV transition metal. Its applications as an ultra-high-temperature ceramic (UHTC) coating for aerospace vehicles has been one of the main thrusts for researching the material. While HfC is theorized to have a melting point well above 3000°C, the material is also known to oxidize at low temperatures. The conditions that HfC would experience in the aerospace would expose it to high temperature atmospheric oxygen and degrade the material faster. In post mortem analyses of HfC exposed to similar conditions, the top layers of material are revealed to consist of Hafnia ( $\text{HfO}_2$ ), while the middle layers are composed of an amalgam of Hafnium Oxycarbide  $\text{HfO}_x\text{C}_y$ , and the bottom layers remain as pristine HfC. It is known that  $\text{HfO}_2$  is the main ablation product of HfC, but the interlayer between the  $\text{HfO}_2$  and HfC remains of interest as it is seemingly capable of preventing oxygen from diffusing further into the ceramic composite. Experimental studies of this composite system are limited by observable results at high temperature, however this can be bypassed through materials simulation. Generally, machine learned potentials use a single material structure as a starting point for simulations, and this ignores a variety of different compositions and structures of materials made from the same elements. In this work, we attempt to develop a machine learned interatomic potential (MLIAP) capable of describing oxygen diffusing into the ceramic composite  $\text{HfO}_2$ , HfC and  $\text{HfO}_x\text{C}_y$  by creating a robust database of over 300,000 diverse training data points.

**1. Introduction.** Over the last half century there has been a growing interest in group IV transition metal carbides, borides and nitrides. [21] These materials, known as UHTCs, are known for their high melting points (above 3000°C), which makes them quite attractive for applications as protective coatings in hypersonic and propulsion related aerosurfaces. These materials also exhibit relatively low densities, high hardness, high Young's modulus and excellent thermal conductivity, which makes them suitable for potential re-usable atmospheric re-entry vehicles or gas turbines.

Of these UHTCs, Hafnium Carbide (HfC) and Hafnium Dioxide ( $\text{HfO}_2$ ) have received quite a bit of attention due to their potential applications in standard gate dielectrics and high density logic and memory devices, plasma facing components in nuclear reactors, absorber coatings for solar cells, and of course the aforementioned aerospace surfaces. [22, 16, 8, 4] HfC in particular has been in discussions of ceramic matrix composites (CMCs), and several studies have been conducted on its addition to a carbon fiber-reinforced Silicon Carbide (SiC) matrix (C/SiC).

The introduction of HfC to C/SiC (forming C/SiC-HfC) mitigates the effects of reaching such extreme temperatures since HfC has a melting point of roughly 3890°C. [11, 10] On its own, HfC has the propensity to oxidize at temperatures as low as 500°C creating the main ablation product,  $\text{HfO}_2$ . [12, 23, 3] A downside to the formation of  $\text{HfO}_2$  is that it forms in a porous fashion, allowing high temperature atmospheric oxygen to infiltrate the system and degrade the material further [17]. When exposed to oxygen, the HfC ceramic forms three layers of different material. The initial layer is fully oxidized to  $\text{HfO}_2$ , the middle layer is a partially oxidized HfC layer containing some form of  $\text{HfO}_x\text{C}_y$ , and the final layer is pure HfC. [1]

As one might expect, experimental research on these materials at ultra-high temperatures is quite costly, as well as potentially dangerous. Additionally, experiments are limited by the maximum temperature that can be achieved by modern technologies. In the case of studying HfC, these composite ceramics are typically conducted at temperatures near

---

\*San Diego State University, vayleamelieveracruz@gmail.com

†Sandia National Labs, elsikor@sandia.gov

1600°C. We can however, bypass these physical limitations by simulating the materials. Over the last decade the rise in machine learning has led to the creation of many interatomic potentials capable of describing a range of materials at extreme conditions. These potentials are an attractive form of materials simulations because they can produce fairly accurate results at the billion atom scale, while remaining computationally cheap. In this study, we will detail the approach we used to study these UHTC composites at the Density Functional Theory (DFT) level and ab initio molecular dynamics level to generate training data. Additionally, we will focus on explaining the success and failures of different methods of developing MLIAPs using Spectral Neighbor Analysis Potential (SNAP), Genetic Algorithm SNAP (GA-SNAP), SNAP neural networks and Atomic Cluster Expansion (ACE).

Both SNAP and ACE methods uses bispectrum components as descriptors, which are representations of elements and structures. These bispectrum components turn neighboring atoms into 4D hyperspherical coordinates to describe local atomic environments. The SNAP method describes atoms and its environment using 4 bodies, while ACE is an N-body expansion. Rationalizing descriptor space for SNAP and ACE as a way to visualize how structures differ from one another, we want to make a robust training data set by changing atomic environments in as many different ways as possible. We surmised that including multiple stoichiometries and compositions can fill more descriptor space and better inform a MLIAP to create stable structures.

It has been documented that MLIAPs can benefit from the inclusion of structures that lie beyond domain expertise [18]. Most MLIAPs are trained off of a single known structure of a given material, which are considered domain expertise. Non domain expertise structures are those created through genetic algorithms, which are capable of constructing materials of the same elemental make up, but with various stoichiometries and different crystal structures. The inclusion of these non domain expertise structures can additionally fill the descriptor space of the potential far more than any one structural composition can. In this work, we focus on studying a diverse array of structures for  $\text{HfO}_2$ ,  $\text{HfC}$  and  $\text{HfOC}$  to show how different compositions of materials can affect an MLIAP.

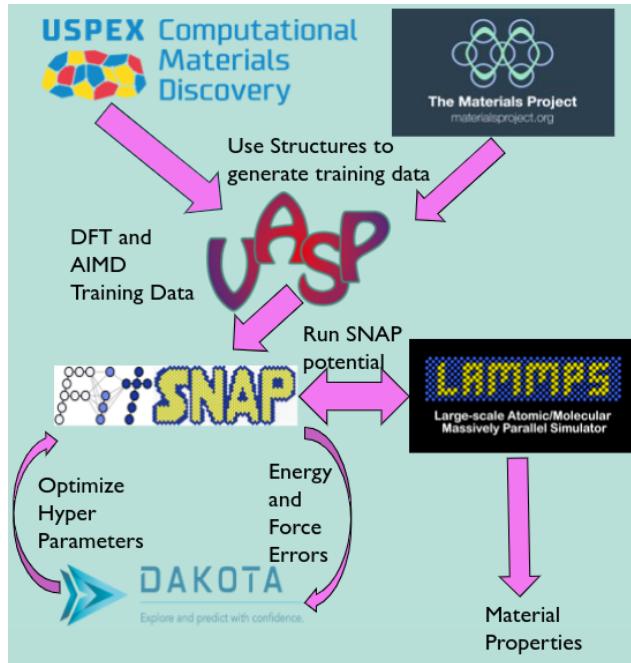
**1.0.1. Benchmarking Pseudopotentials.** The first steps taken to generate training data was to benchmark the performance of pseudopotentials in Vienna ab initio Simulation Package (VASP) 5.4 for Hf, O and C. To do this, known structures of cubic  $\text{HfO}_2$  and  $\text{HfC}$  were surveyed using the Projector-Augmented Wave DFT methods, with the Perdew-Burke-Ernzerhoff (PBE) exchange functional. [15] For  $\text{HfO}_2$ , there were 7 pseudopotentials tested for oxygen and 3 for hafnium. All 21 combinations were tested on cubic Fm3m  $\text{HfO}_2$  with a cutoff energy of 700 eV with a 4x4x4 k-point mesh centered at the gamma point. Cubic Fm3m  $\text{HfC}$  was tested in the same fashion, but with carbon having 6 available pseudopotentials for testing.

The pseudopotential combinations were benchmarked according to their ability to replicate the lattice constants and bulk modulus of the materials. The chosen pseudopotentials were the Hf-pv pseudopotential, which denotes 10 valence electrons from the d and p orbitals; O-s-GW which denotes an 6 electron valency; and the C-h which has a 4 electron valency but is known to have better performance in dimers and short bonds. The  $\text{HfO}_2$  calculation for the Hf-pv and O-s-GW pseudopotentials resulted in a lattice constant of 5.072 Å with a relative error of 0.84%, and a bulk modulus of 242.5 Gpa with a relative error of 2.21%. The  $\text{HfC}$  calculation for the chosen pseudopotentials resulted in a lattice constant of 4.636 and a relative error of 0.19%, with a bulk modulus of 245.9 and a relative error of 4.19%.

**2. Methods.** The general workflow for the creation of an MLIAP can be summed up as generation of training data, using a method to calculate hyperparameters that satisfy

objective functions to lower force and energy errors, and then test the potential in molecular dynamics. In this work, we used a variety of methods to generate structures for training data, which is detailed below. We used VASP to get DFT and AIMD training data and then feed them into different potential fits [7]. The potential fits are then optimized with genetic algorithms to lower force and energy errors, and in some cases fit to objective functions. Finally the potentials are run in LAMMPS to check their stability [20]. If a potential is unstable, meaning extrapolation errors are too great or MD simulations in LAMMPS are unable to retain atoms in the simulation cell, we return to this iterative process of adding new training data, refitting and re-optimizing until we get a stable potential.

FIG. 2.1. Schematic for the workflow of training data generation into Fits, optimization and testing the potential in LAMMPS.

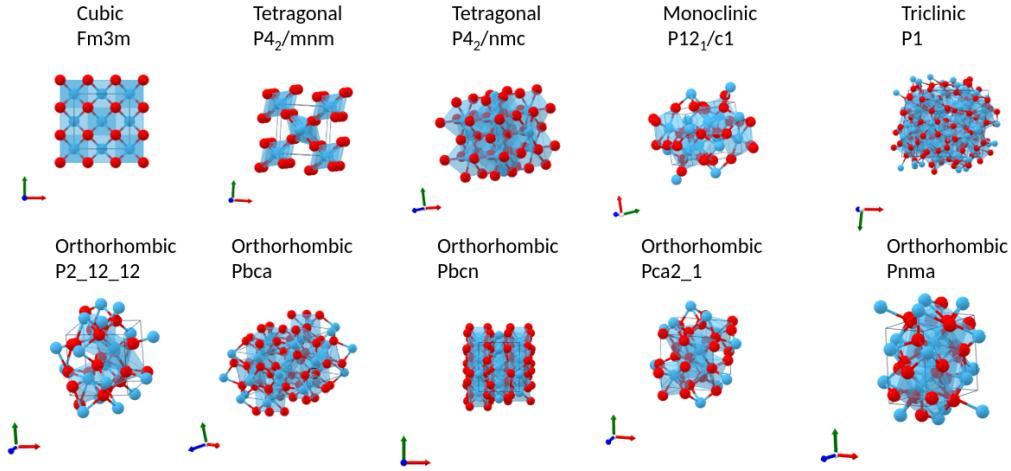


## 2.1. Generating Training Data.

**2.1.1. Existing Structures.** For the case of creating a rudimentary potential we surveyed The Materials Project for known structures of HfO<sub>2</sub> and HfC where we found 10 distinct crystal compositions for HfO<sub>2</sub> and 6 for HfC. [6] After relaxing the cells, the resulting final structures were turned into 2x2x2 supercells containing 96 atoms using an open visualization tool known as OVITO. [19] These materials were then run in AIMD using an isobaric ensemble and Langevin thermostat with gamma set to 3.0 ps<sup>-1</sup> for each element, and in a separate run they are run with an isochoric ensemble and the Nose-Hoover thermostat. The isobaric ensemble was chosen to capture thermal expansion since the precursor materials would be heated to extreme temperatures to form HfOC. AIMD calculations were run at three sampling temperatures, 2900 K, 3073 K, and 3573 K, which corresponds to HfO<sub>2</sub> in solid phase, at the melting point, and in liquid phase, respectively.

**2.1.2. Genetic Algorithms Structures.** To fill descriptor space with varied compositions, we used the Universal Structure Predictor: Evolutionary Xtallography (USPEX) genetic algorithms to create structures of HfO<sub>2</sub>, HfC and HfOC with varying stoichiometries

FIG. 2.2. Structures found for  $\text{HfO}_2$  using the materials project. Each structure was relaxed in DFT and run in isobaric and isochoric ensembles AIMD at 2900K, 3073K, and 3573K.



and topologies of the materials.[14] USPEX calculations were set to minimize energies for each genetic composition. This would provide the potential with structural information unseen from just the physical structures denoted earlier. Even though some of the structures generated through USPEX may not be physical compositions, their inclusion in the training data set can still be used to fill descriptor space of the potential.

Initial USPEX calculations were run by using variable compositions, leading to roughly 1500 unique structures, consisting of approximately 500 for  $\text{HfO}_2$ ,  $\text{HfC}$  and  $\text{HfOC}$ . Each of which is relaxed in DFT at 700 eV cutoff energy at  $4 \times 4 \times 4$  k points. Additionally, the compositions with the lowest formation energy were surveyed at the AIMD level by turning their unit cells into  $2 \times 2 \times 2$  supercells. Some of the structures were considered too unphysical for VASP to converge energies for, which led to some of the AIMD calculations being excluded from the data set.

**2.1.3. Defect and Interstitial Calculations.** Defects in the materials were also considered for training data, as objective functions for defect energies and interstitial atoms can be used as a parameter to fit potential candidates to. To create vacancy defects, we used existing  $2 \times 2 \times 2$  supercells of cubic Fm3m  $\text{HfO}_2$  and  $\text{HfC}$  and simply delete a single atom of either Hf, O or C and create a corresponding calculation for each vacancy at the DFT level with a 700 eV cutoff energy and 1 k point at the gamma point. Then AIMD calculations using the isobaric ensemble and Langevin thermostat are conducted at the three sampling temperatures stated previously. Substitutional defects are also considered for the training set, which were made by changing an atom of Hf, O, or C with one of the remaining two element types. In total there are 8 different substitutional defects tested across the two materials.

Interstitials were induced by using the Atomic Simulation Environment python module (ASE). [9] Supercells of  $\text{HfO}_2$  and  $\text{HfC}$  were used to create structures with an extra atom of either Hf, O or C in both materials, leading to 6 distinct interstitial structures. Reference structures for pure Hf, O and C need to be calculated to make objective functions of the defect energies. A cubic Hf material was used as the reference energy for hafnium, the oxygen reference used was a dimer of oxygen gas, and the reference for carbon was graphite.

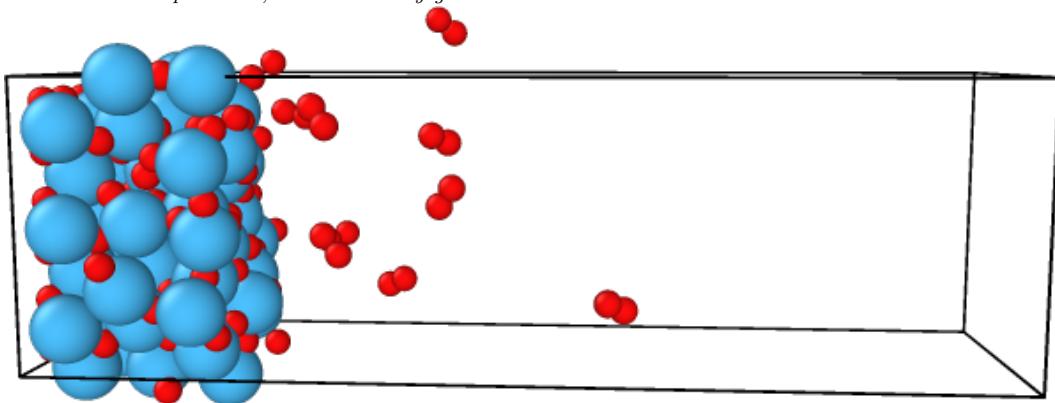
**2.1.4. HfOC Stoichiometries.** The exact structure of HfOC is unknown, as it is a partially oxidized interlayer of HfC, making simulations of the material quite difficult. The international crystallography database (ISCD) contained one structure, which was an Fm3m cubic structure of HfOC that is quite similar to HfC. The difference with this structure is that its oxygen and carbon sites are shared, which meant that simulating the file directly from the database results in failed calculations due to oxygen and carbon atoms being directly on top of each other. To fix this, we can simply modify the file to have either an O or C at each of the four possible sites. Since HfOC is a variable composition we can adjust the stoichiometry of the file to generate DFT and AIMD training data with varying percentages of oxygen and carbon.

This results in 0%, 25%, 50%, 75% and 100% Oxygen structures of HfOC. Each stoichiometry, is relaxed in DFT and simulated with AIMD in the same fashion. Additionally, vacancies and intersstitial calculations were run, following the same workflow used in creation and calculations for the  $\text{HfO}_2$  and HfC materials. This resulted in 28 different structures with exact percentages of O and C.

**2.1.5. Surface and Adsorbate Structures.** In addition to fitting against defect and interstitial energies, surface and adsorption energies can be used as an extra parameter to fit potentials to. Since the ceramic composite has a surface of  $\text{HfO}_2$ , we created surfaces of  $\text{HfO}_2$  with either a Hf or O terminating layer. These termination layers are necessary to prevent simulations from inducing a dipole across the  $\text{HfO}_2$  structure. The materials simulations must also include a new portion in the calculation setup, as not all of the atoms in the simulation should be allowed to move. Only the first few layers of atoms should be considered the surface and allowed to move, while the others in the bulk should be frozen.

To make these surface structures, a 3x3x3 topology of  $\text{HfO}_2$  was used with Atomsk to first create a vacuum in the z direction of the structure, and then to apply selective dynamics to specific atoms in the file. [5] The structures were then loaded in OVITO, and using the slice function, we were able to make the different terminating layers. The two surfaces were then run in DFT and npt AIMD using 1 k point at the gamma point, and run at 3073 K.

FIG. 2.3. Oxygen terminated surface of  $\text{HfO}_2$  run in an isobaric ensemble with at 3073K. The material melts at this temperature, and causes oxygen dimers to release.



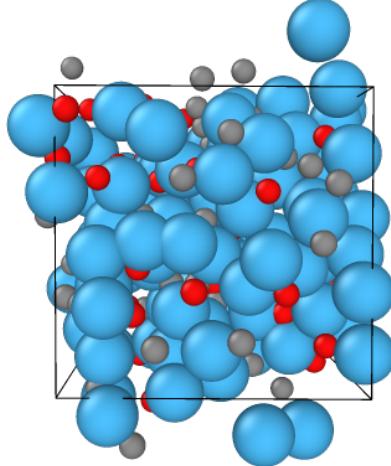
### 3. Results.

**3.0.1. Preliminary SNAP Potentials.** The earliest attempt at making an MLIAP for HfOC used linear SNAP method for fitting. It was first tested by fitting training data on cubic and tetragonal  $\text{HfO}$ . Hyperparameters were quick to calculate when making a simple

potential with objective functions for lattice constants, radial distribution functions (RDFs) and force and energy errors. The addition of more diverse training data, like USPEX structures and defects began to make calculation times for adequate hyperparameters to take much longer, and caused us to abandon using linear SNAP and GA-SNAP optimization methods for quick testable potentials. We moved to SNAP neural networks afterwards as a way to quickly see how additions of new training data affected stability of our potentials. The first Neural Network fit was not rigorous, but stable at 1000 K for 25 ps, which was on par with our last linear SNAP model. This first Neural Network fit also took a fraction of the time to calculate hyperparameters, compared to the linear SNAP method. We theorized that adding in "liquid" compositions into our training data would help our potential be stable at higher temperatures by including small RDFs, which prevent extrapolation errors in cases of MD simulations where those atomic environments are present.

**3.0.2. Accounting for Smaller RDFs with Liquid Structures.** To create liquid structures of  $\text{HfO}_2$  and  $\text{HfOC}$ , there were two ways to go about creating them. The first, and easiest, was to use packmol to generate a random output of  $\text{HfO}_2$  "molecules" within a set boundary condition. [13] This allowed for AIMD calculations using these structures to include training data on atomic interaction with very short distances ( $< 1.5 \text{ \AA}$ ). The structures were relaxed in DFT at 700 eV cutoff energy and at 1 k-point centered at gamma. AIMD calculations were run with the same methods, except only at 3073 K.

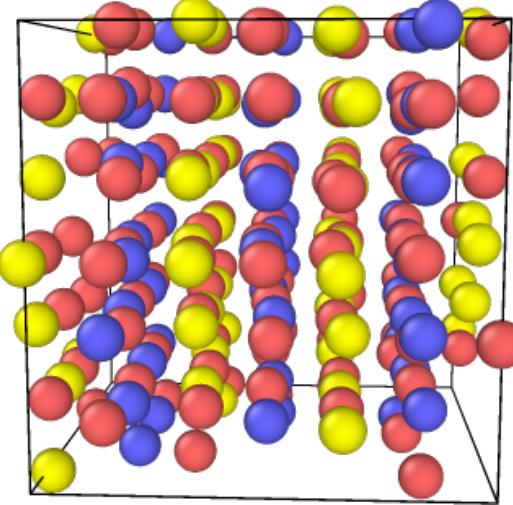
FIG. 3.1. *Liquid structure formed from running a rudimentary neural network potential in LAMMPS at 5000 K. After 300 timesteps the simulation crashed, and the resulting coordinates for atoms were turned into a file that is usable in VASP.*



The alternative route to constructing liquids for simulations used LAMMPS and a rudimentary potential for  $\text{HfOC}$ . The first step in the process was to develop a simple potential for  $\text{HfOC}$ . The potential used in production of these liquid structures will be discussed in more detail in the results section. Using LAMMPS and a data file of a 3x3x3 supercell of  $\text{HfOC}$ , we can run a calculation at higher temperatures and take the results of the first few timesteps to use as a "liquid" structure. For this method, we used the third iteration of the SNAP neural network fits made for  $\text{HfOC}$  and ran an isochoric MD simulation at 5000K and

10000K. The resulting structures were then subject to the same workflow as the packmol structures.

FIG. 3.2. A snapshot of the last neural network potential for HfOC tested on a 3x3x3 supercell of 50% Oxygen and Carbon at 1873 K. The red atoms represent Hf, the blue represents O, and the yellow represents C.



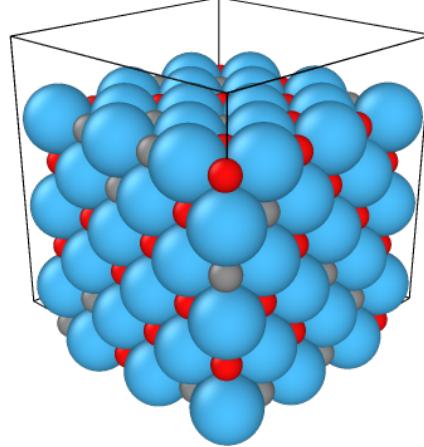
With these additions to the database, we used a SNAP neural network fit with a learning rate of 1e-4, 1000 epochs, and the inclusion of every single training data point separated into 106 training groups. This Neural Network fit used roughly 250,000 training data points and took 2 days to finish calculating hyperparameters. The resulting HfOC potential was stable at 1873 K across a 25 ps MD simulation for a cubic 5x5x5 simulation cell. This was tested with varying compositions of the simulation cell as well, using set stoichiometries of HfOC, as well as using a Monte Carlo randomized structure.

**3.0.3. Monte Carlo Structures.** The Monte Carlo structure used in testing the SNAP neural network fit was based off of a python function we made to randomize the oxygen and carbon placements in a cubic Fm3m unit cell. Our last effort made to generate a diverse training set was to make a script capable of generating structures with random placements of oxygen and carbon in an HfOC supercell. This was done by making a python function based on a simple unit cell of HfOC. The function is capable of building supercells of any size, given the right input. Using the sample unit cell, the function can discern atom type, and uses several other inputs to determine whether the non-Hafnium atoms are an oxygen or carbon within the lattice. The function is also capable of retaining the correct number of atoms for all sizes of supercells. An added functionality of this python function is its ability to change the number of layers that are entirely oxygens or carbons, as well as the percentage of oxygen and carbon throughout the cell. The boundary layers of oxygen and carbon can also be changed to allow for studies of more bulk structures in the interlayer, or even near the transition between HfO to HfOC or HfOC to Hfc.

This kind of random assortment was necessary for describing how the HfOC bulk materials force and energies change depending on local environments. The stoichiometric versions of HfOC tested prior do not contain any randomness to their structure as they are built from a single unit cell, which limits the ability of the MLIAP to accurately describe sections of the HfOC interlayer that do not conform to the set stoichiometric ratios. With this sort

of random assortment we can feed more variable training data to fill the descriptor space of the potential.

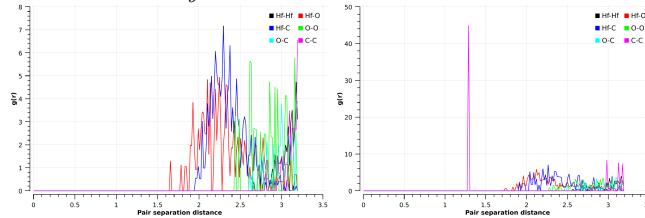
FIG. 3.3. This structure is a  $3 \times 3 \times 3$  supercell of  $\text{HfOC}$ , with 67% oxygen randomly distributed throughout the lattice.



Aside from being a way to generate new training data this method was also useful for making a data file for LAMMPS to read in and test our  $\text{HfOC}$  potentials on. This function was used primarily for testing how the SNAP neural network fit with 250,000 structures performed on different areas of the entire  $\text{HfOC}$  composite. No studies were made on pure  $\text{HfO}$  or  $\text{HfC}$  layers, but attempts at simulating the  $\text{HfOC}$  interlayer in the bulk and near the transitions to pure  $\text{HfO}$  and  $\text{HfC}$ . Each of which was stable at 1873 K, which lies at the experimental limit for  $\text{HfOC}$ .

**3.1. Monte Carlo AIMD Results.** Although this Monte Carlo randomize function was useful for benchmarking the performance of the  $\text{HfOC}$  potential, training data from these compositions were excluded from the training data set. This is because of an error that was noticed while observing the radial distribution functions (RDFs) of several supercells ran in AIMD. The compositions made with a layer of oxygen in the top most layer and a layer of carbon at the bottom have carbon-carbon (C-C) RDFs that indicate some sort of clumping of carbon. This is noticeable as the carbon percentage diminishes, as structures with a carbon percent above 50% have a C-C RDF between 2.5 and 3.5 Å, but as the carbon percent diminishes, and oxygen becomes the more prevalent atom, the C-C RDF lies between 1 and 2 Å.

FIG. 3.4. On the left is the radial distribution function of a  $3 \times 3 \times 3$  supercell with 40% oxygen and a layer of Oxygen on one end, and a layer of Carbon on the opposite. The left graph is the rdf of the same kind of supercell, but with 80% oxygen. As oxygen becomes more prevalent in this cell, the carbon-carbon rdf shifts to wards 1 Å and its intensity increases.

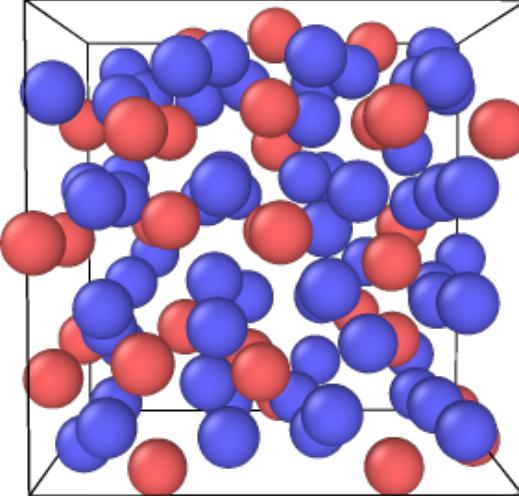


Interestingly, a composition with no boundary layers of oxygen or carbon is run in AIMD, the C-C RDF lies within 2 and 3.5 Å regardless of carbon percentage. Possible reasons for why this is occurring are because of periodic boundary conditions, and an induced dipole due to differences in terminating layers of the material. Assuming the former, this could potentially be fixed by adding a vacuum in the z direction to prevent any interaction amongst the layers.

**3.2. Stable ACE Potential for  $\text{HfO}_2$ .** An alternate method used to create an MLIAPI for the  $\text{HfO}_2$ ,  $\text{HfOC}$  and  $\text{HfC}$  system is atomic cluster expansion (ACE), which we used to create a potential for  $\text{HfO}_2$ . [2] This potential was made more as a proof of concept, as we can learn from making a stable ACE potential and then apply those lessons learned to the construction of an ACE potential for  $\text{HfOC}$ . Using the defect and interstitial structures of  $\text{HfO}_2$  and the structures of pure Hf, C and O we create objective functions for the defect and interstitial energies. Using Dakota algorithms, we optimize the hyperparameters and group weights to achieve minimal force and energy errors for potential candidates.

An additional inclusion in the ACE potential is the optimization of inner cutoff radius for element pairs. This stops the potential from overcoming the repulsive energies between atoms, and allows resulting structures to remain in the physical domain. Through this ACE potential we were able to achieve a stable  $\text{HfO}_2$  potential for a  $3 \times 3 \times 3$  supercell at 3000 K. We liken this stability to the inclusion of a repulsive wall added into the ACE potential that is present at interatomic distances smaller than the optimized inner RDF cutoff. By including this we are capable of simulating at extreme conditions that go far beyond the experimental limit.

FIG. 3.5. A snapshot of the ACE potential for  $\text{HfO}_2$  simulating a  $3 \times 3 \times 3$  supercell at 3000 K.



**4. Conclusions.** At the current juncture, the  $\text{HfOC}$ ,  $\text{HfO}_2$  and  $\text{HfC}$  training data set contains over 300,000 training data points. Current  $\text{HfOC}$  potentials using the SNAP neural network fits are capable of running simulations of all sorts of stoichiometries for  $\text{HfO}_x\text{C}_y$ . The potential can also run cells of  $5 \times 5 \times 5$  size without throwing atoms out of the cell, and run for 25 ps (50000 timesteps). Additionally, the SNAP neural netowrk potential using every  $\text{HfOC}$  and  $\text{HfO}$  structure available is stable enough to run at 1873 K, which meets the current experimental limit for observable results of  $\text{HfOC}$  formation. This SNAP neural network was not rigorously tested, and yet it was capable of matching the experimental limit

for temperature. Continuing to highlight this accomplishment, it finished optimizing in two days, and contained roughly 250,000 training data points. We propose that the inclusion of "liquid" structures play a large role in the stability at higher temperatures because of the data it provides on RDFs smaller than 1.5 Å that helps limit extrapolation errors.

An important contribution to this field is the database itself. There are no databases for these Hafnium ceramics available, and in this work three separate databases were made for HfO, HfC and HfOC. These can be used for various studies, either continuing the creation of an HfOC potential or even for the smaller HfO and HfC systems. Additionally, studies into how grouping training data would be a useful endeavor for the field of MLIAPs in general, as currently there are no best practices for separating training groups, and this database contains multiple different stoichiometries and space groups for all three Hf ceramics.

While the training data set is numerous, it is unknown how many data points, if any, are repetitive and occupying the same compositions. Going forward, a valuable addition to making new fits would be to find an efficient way to parse out which data points are extraneous data, as well as the ability to construct a t-distributed stochastic neighbor embedding plot for these neural network fits. This would allow us to see how the entire training data set affects the potential with respect to each group of training data. Sensitivity analysis of how any single training group affects the potential would also be a powerful tool for understanding which compositions, are most necessary for a stable and robust potential. Additionally, with the success of an ACE potential for HfO<sub>2</sub> stable at 3000 K, the next step forward for an HfOC MLIAPIP should use ACE descriptors with objective functions for interstitial energies and defect energies.

**5. Acknowledgements.** 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.

I would like to thank the Sustainable Horizons Institute for matching me with my mentors here at Sandia, additionally, David Littlewood and Andy Sallinger for this opportunity. Vicente Corral and Coreen Mullen for assisting in benchmarking and standardizing pseudopotentials. Natalie Weller for creating the generative monte carlo python function. The FusMatML group for their thoughts and contributions in discussion of this project. Thank you to James Goff for teaching me how to make liquid structures and automation with python. Thank you to Meg McCarthy and Drew Rohskopf for teaching me how to use OVITO, python, ASE and FitSNAP, and helping to debug many of the software problems I ran into. Lastly, I want to thank my mentor, Ember Sikorski, who facilitated my learning in all aspects of this project, and supported me throughout my time here. Thank you all for helping me rediscover my confidence in my abilities as a researcher.

## REFERENCES

- [1] C. B. BARGERON, R. C. BENSON, R. W. NEWMAN, A. N. JETTE, AND T. E. PHILLIPS, *Oxidation mechanisms of hafnium carbide and hafnium diboride in the temperature range 1400 to 2100 c*, Johns Hopkins APL Technical Digest; (United States).
- [2] R. DRAUTZ, *Atomic cluster expansion for accurate and transferable interatomic potentials*, Phys. Rev. B, 99 (2019), p. 014104.
- [3] L. DUAN, L. LUO, L. LIU, AND Y. WANG, *Ablation of c/sic-hfc composite prepared by precursor infiltration and pyrolysis in plasma wind tunnel*, Journal of Advanced Ceramics, 9 (2020).

- [4] K. HANS, S. LATHA, P. BERA, AND H. C. BARSHILIA, *Hafnium carbide based solar absorber coatings with high spectral selectivity*, Solar Energy Materials and Solar Cells, 185 (2018), pp. 1–7.
- [5] P. HIREL, *Atomsk: A tool for manipulating and converting atomic data files*, Computer Physics Communications, 197 (2015), pp. 212–219.
- [6] M. JONG, W. CHEN, T. ANGSTEN, A. JAIN, R. NOTESTINE, A. GAMST, M. SLUITER, C. KRISHNA, C. K. ANDE, S. ZWAAG, J. PLATA, C. TOHER, S. CURTAROLO, G. CEDER, K. PERSSON, AND M. ASTA, *Charting the complete elastic properties of inorganic crystalline compounds*, Scientific Data, 2 (2015).
- [7] G. KRESSE AND J. FURTHMÜLLER, *Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set*, Phys. Rev. B, 54 (1996), pp. 11169–11186.
- [8] T. L. F., N. B. A., B. G. G., K. E. A., AND M. V. I., *Preparation and study of nuclear safe porous ceramics comprised of neutron poisons (Hf, Gd) impregnated with plutonium sol-gel PuO<sub>2</sub>*, Journal of Nuclear Science and Technology, 39 (2002), pp. 850–854.
- [9] A. H. LARSEN, J. J. MORTENSEN, J. BLOMQVIST, I. E. CASTELLI, R. CHRISTENSEN, M. DUŁAK, J. FRIIS, M. N. GROVES, B. HAMMER, C. HARGUS, E. D. HERMES, P. C. JENNINGS, P. B. JENSEN, J. KERMODE, J. R. KITCHIN, E. L. KOLSBJERG, J. KUBAL, K. KAASBJERG, S. LYSGAARD, J. B. MARONSSON, T. MAXSON, T. OLSEN, L. PASTEWKA, A. PETERSON, C. ROSTGAARD, J. SCHIOTZ, O. SCHÜTT, M. STRANGE, K. S. THYGESEN, T. VEGGE, L. VILHELMSEN, M. WALTER, Z. ZENG, AND K. W. JACOBSEN, *The atomic simulation environment—a python library for working with atoms*, Journal of Physics: Condensed Matter, 29 (2017), p. 273002.
- [10] D. W. LIPKE, S. V. USHAKOV, A. NAVROTSKY, AND W. P. HOFFMAN, *Ultra-high temperature oxidation of a hafnium carbide-based solid solution ceramic composite*, Corrosion Science, 80 (2014), pp. 402–407.
- [11] X. LUAN, G. LIU, M. TIAN, Z. CHEN, AND L. CHENG, *Damage behavior of atomic oxygen on a hafnium carbide-modified c/sic composite*, Composites Part B: Engineering, 219 (2021), p. 108888.
- [12] H. LUN, Y. ZENG, X. XIONG, Z. YE, Z. ZHANG, X. LI, H. CHEN, AND Y. LIU, *Oxidation behavior of non-stoichiometric (zr,hf,ti)<sub>c</sub>x carbide solid solution powders in air*, Journal of Advanced Ceramics, 10 (2021).
- [13] L. MARTÍNEZ, R. ANDRADE, E. BIRGIN, AND J. M. MARTÍNEZ, *Packmol: A package for building initial configurations for molecular dynamics simulations*, Journal of Computational Chemistry, 30 (2009), pp. 2157 – 2164.
- [14] A. OGANOV AND C. GLASS, *Crystal structure prediction using ab initio evolutionary techniques: Principles and applications*, The Journal of chemical physics, 124 (2006), p. 244704.
- [15] J. P. PERDEW, K. BURKE, AND M. ERNZERHOF, *Generalized gradient approximation made simple*, Phys. Rev. Lett., 77 (1996), pp. 3865–3868.
- [16] C. RODENBÜCHER, E. HILDEBRANDT, K. SZOT, S. U. SHARATH, J. KURIAN, P. KOMISSINSKIY, U. BREUER, R. WASER, AND L. ALFF, *Hafnium carbide formation in oxygen deficient hafnium oxide thin films*, Applied Physics Letters, 108 (2016), p. 252903.
- [17] F. RÉJASSE, G. TROLLIARD, O. RAPAUD, A. MAÎTRE, AND J. DAVID, *Tem study of the reaction mechanisms involved in the carbothermal reduction of hafnia*, RSC Adv., 5 (2015), pp. 45341–45350.
- [18] E. L. SIKORSKI, M. A. CUSENTINO, M. J. McCARTHY, J. TRANCHIDA, M. A. WOOD, AND A. P. THOMPSON, *Machine learned interatomic potential for dispersion strengthened plasma facing components*, The Journal of Chemical Physics, 158 (2023), p. 114101.
- [19] A. STUKOWSKI, *Visualization and analysis of atomistic simulation data with OVITO—the Open Visualization Tool*, MODELLING AND SIMULATION IN MATERIALS SCIENCE AND ENGINEERING, 18 (2010).
- [20] A. P. THOMPSON, H. M. AKTULGA, R. BERGER, D. S. BOLINTINEANU, W. M. BROWN, P. S. CROZIER, P. J. IN 'T VELD, A. KOHLMAYER, 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.
- [21] S. V. USHAKOV, A. NAVROTSKY, Q.-J. HONG, AND A. VAN DE WALLE, *Carbides and nitrides of zirconium and hafnium*, Materials, 12 (2019).
- [22] S. L. VENDRA, E. KOROLEVA, A. FILIMONOV, S. VAKHRUSHEV, AND R. KUMAR, *Spark plasma sintered si(hf)c nanocomposites exhibiting thermally stable dielectric behavior processed through precursor route*, Materials Chemistry and Physics, 302 (2023), p. 127717.
- [23] V. A. ZHILYAEV, Y. G. ZAINULIN, S. I. ALYAMOVSKII, AND G. P. SHVEIKIN, *High-temperature oxidation of oxycarbides, oxynitrides, and oxycarbonitrides of zirconium and hafnium*., Porosh. Met., Akad. Nauk Ukr. SSR No. 8, 38-43(Aug 1972).

## ON COORDINATE ENCODING IN MULTI-FIDELITY NEURAL EMULATORS

CRISTIAN J. VILLATORO\*, GIANLUCA GERACI†, AND DANIELE E. SCHIAVAZZI‡

**Abstract.** Multi-fidelity emulators have versatile applications in both forward and inverse problems within computational science. In addition, thanks to recently proposed neural architectures, they offer substantial flexibility for fusing information from multiple models, maintaining a significant efficiency advantage when compared to single-fidelity approaches. In this context, existing neural multi-fidelity emulators are based on treating separately the linear and nonlinear correlation between equally parameterized high- and low-fidelity approximants. However, complex models ensembles in science and engineering applications often exhibit limited linear correlations between models, which can hinder the effectiveness of these approaches. In this work, we present a general strategy that aims at maximizing the linear correlation between two models through input encoding. We demonstrate the performance of our approach on four numerical test problems, and we show the ability of the proposed multi-fidelity emulator to accurately recover the high fidelity model response under an increasing number of quasi-random samples. We also show that input encoding produces emulators with significantly simpler non-linear correlations. Finally, input encoding enables information fusion between low- and high-fidelity models with dissimilar parametrization.

**1. Introduction.** Recent advances in numerical methods and the growing accessibility of high performance hardware resources have significantly improved the accuracy of physics-based simulations for a variety of applications, ranging from cardiovascular modeling to climate forecasting. Due to the same advances, the computational costs to run such simulations has significantly increased so that quantifying prediction uncertainty or integrating these models in inference tasks becomes often intractable. To mitigate this problem, *multi-fidelity* (MF) emulators leverage a large number of solutions from computationally inexpensive *low-fidelity* (LF) approximants, to produce accurate *high-fidelity* (HF) model predictions using a limited amount of expensive data.

Previous work has focused on networks with the ability to handle datasets with variable annotation quality [3] and transfer learning between model fidelities [2]. Additionally, recent approaches combine three dense sub-networks designed to learn a LF representation, a linear correlation and a nonlinear correlation between the LF and HF model predictors [9]. Other approaches have proposed the use of Bayesian neural networks [8] or operator networks [6]. However, in practical applications the linear correlation among models is limited by their mathematical formulation, which may not be consistent across multiple fidelities, or lack of consistency among their predictive abilities. In such cases, the introduction of a coordinate transformations can sensibly increase the linear correlation between LF and HF models (see, e.g., [10, 11]), producing more accurate predictors and surrogates with potentially simpler architectures.

We propose here an *encoded* multi-fidelity architecture where a linear encoder is applied to the HF model inputs before evaluating a LF predictor that, in turn, is fed to the HF correlation network. We also demonstrate the performance of the proposed approach using examples where the number of inputs for the LF and HF models is both equal and different.

The paper is organized as follows. In Section 2, we provide a motivation using simple exponential functions, present the proposed architecture, and discuss some details regarding the training and hyperparameter optimization process. In Section 3 we demonstrate the performance of the encoded multi-fidelity network using four numerical benchmarks. A discussion and the paper conclusions are finally provided in Section 4 and 5, respectively.

---

\*University of Notre Dame, cvillat2@nd.edu

†Sandia National Laboratory, ggeraci@sandia.gov

‡University of Notre Dame, dschiava@nd.edu

## 2. Coordinate encoding in multi-fidelity neural networks.

**2.1. Motivation.** Consider a low- and high-fidelity model pair defined as

$$\begin{cases} y_L(x) = e^x & x \in [-1, 1] \\ y_H(x) = e^{x/2} & x \in [-1, 1]. \end{cases} \quad (2.1)$$

It's clear that although  $y_L$  and  $y_H$  are very similar, they are not linearly correlated (here we mean that the two models cannot be written as a linear function of the other). However, the two functions can be related exactly by composing  $y_L$  with a simple transformation, namely  $y_H = y_L \circ T$  where  $T(x) = x/2$ . This demonstrates that a coordinate transformation can be used to increase the linear correlation among models while reducing their nonlinear correlation.

Now consider a different low- and high-fidelity model pair defined as

$$\begin{cases} y_L(x) = e^x & x \in [-1, 1], \\ y_H(x) = e^{2x} & x \in [-1, 1]. \end{cases} \quad (2.2)$$

Similarly, a reasonable guess for a simple transformation that relates  $y_L$  and  $y_H$  would be  $T(x) = 2x$ . However, this transformation causes extrapolation. Notice  $y_L(T(x))$  is only defined when  $-0.5 \leq x \leq 0.5$  if  $T(x) = 2x$ . This demonstrates that certain choices of a coordinate transformation cause extrapolation away from the LF model domain, resulting in an emulator whose accuracy might be negatively affected by the lack of accuracy in the LF response.

**2.2. Architecture.** Suppose we seek to generate a surrogate model for the response of a computationally expensive HF solver  $\mathbf{y}_H : \mathcal{X}_H \rightarrow \mathcal{Y}$ . Also suppose  $n$  LF surrogates to be available in the form  $\mathbf{y}_{L,i} : \mathcal{X}_{L,i} \rightarrow \mathcal{Y}$ ,  $i = 1, \dots, n$ , where  $\mathcal{X}_{L,i}$  may have a different dimensionality than  $\mathcal{X}_H$  and  $\mathcal{Y}$  defines a common space for the  $m$  quantities of interest. In this work we will focus on *bi-fidelity* approximations ( $n = 1$ ) of scalar quantities of interest ( $\mathcal{Y} \subset \mathbb{R}$ ).

We also define a map (referred to as an *encoder*)  $T : \mathcal{X}_H \times \Theta_T \rightarrow \mathcal{X}_L$ , a LF predictor  $\mathcal{G} : \mathcal{X}_L \times \Theta_L \rightarrow \mathcal{Y}$  and a *correlation* function  $\mathcal{F} : \mathcal{X}_H \times \mathcal{Y} \times \Theta_H \rightarrow \mathcal{Y}$  such that

$$\hat{y}_L = \mathcal{G}(x_L, \theta_L), \text{ and } \hat{y}_H = \mathcal{F}(x_H, y_L \circ T, \theta_H), \quad (2.3)$$

where

$$\mathcal{F}(x_H, y_L \circ T, \theta_H) = \mathcal{F}_l(x_H, y_L \circ T, \theta_H^l) + \mathcal{F}_{nl}(x_H, y_L \circ T, \theta_H^{nl}). \quad (2.4)$$

Here,  $\hat{y}_L, \hat{y}_H$  represent LF and HF predictors,  $\theta_L, \theta_H = \theta_H^l \cup \theta_H^{nl}$  are network parameters and (2.4) indicates a decomposition in a *linear* and *nonlinear* correlation (following [9]), respectively. For the case where  $\mathcal{F}, \mathcal{G}$  and  $T$  are expressed using dense neural networks, a schematic architecture is shown in Figure 2.1. Notice that this network is similar to the network proposed in [9] with the only difference being the injection of the coordinate map  $T$ . This change is motivated by the observation in Section 2.1 that shows the inclusion of a coordinate map may increase the linear correlation between models.

It should be noted that although we are primarily interested in learning a predictor for the HF response, our network also learns the LF predictors. This is more efficient when at least one of the LF surrogates may be computationally expensive, even if not as expensive as the HF solver, e.g., *hierarchical* multifidelity approaches where the LF are associated with coarser meshes. Alternatively, LF solver may all have different interfaces, so integrating

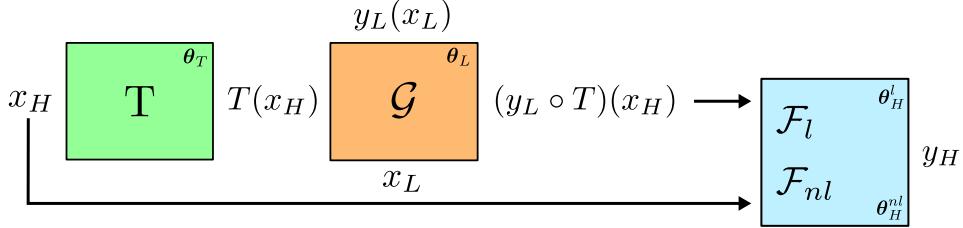


FIG. 2.1. Schematic representation of an encoded multifidelity neural network.

them under one architecture could be unnecessarily complicated. In these cases, by learning the LF predictors as well, we can reduce the online cost of constructing an HF emulator, as well as simplify the implementation working with files containing inputs and output tables for each model.

**2.3. Loss.** The triplet  $(\mathcal{G}, \mathcal{F}, T)$  is learned by solving an optimization problem of the form

$$\inf_{\theta_L, \theta_H^l, \theta_H^{nl}, \theta_T} \mathcal{L}_{\text{Err}} + \mathcal{L}_{\text{Reg}} + \mathcal{L}_{\text{Enc}}, \quad (2.5)$$

where

$$\begin{aligned} \mathcal{L}_{\text{Err}} &= \|\hat{y}_L - y_L\|_2 + \|\hat{y}_H - y_H\|_2, \\ \mathcal{L}_{\text{Reg}} &= \lambda_L \|\boldsymbol{\theta}_L\|_2 + \lambda_H^{nl} \|\boldsymbol{\theta}_H^{nl}\|_2, \\ \mathcal{L}_{\text{Enc}} &= \lambda_T \|\hat{y}_H^l - y_H\|_2 + \lambda_D \text{IS}(T(\mathbf{x}_H); \mathbf{x}_L). \end{aligned} \quad (2.6)$$

In (2.6), we introduce the regularization penalties  $\lambda_L, \lambda_H^{nl}, \lambda_T$ , and  $\lambda_D$  and denote the linear correlation in  $\mathcal{F}$  as  $\hat{y}_H^l = \mathcal{F}_l(x_H, y_L \circ T, \theta_H^l)$ . Additionally, the *interval score* (IS [5]) of a set of  $m$  realizations  $\{\mathbf{s}^{(j)}\}_{j=1}^m \subset \mathbb{R}^n$  with respect to an hyperrectangle  $\mathcal{R} = \prod_{i=1}^n [a_i, b_i]$  is expressed as

$$\begin{aligned} \text{IS} \left( \left\{ \mathbf{s}^{(j)} \right\}_{j=1}^m; \mathcal{R} \right) &= \frac{1}{m} \sum_{j=1}^m d_1(s^{(j)}, \mathcal{R}) \\ &= \frac{1}{m} \sum_{j=1}^m \sum_{i=1}^n \text{ReLU}(a_i - s_i^{(j)}) + \text{ReLU}(s_i^{(j)} - b_i), \end{aligned} \quad (2.7)$$

where  $s_i^{(j)}$  is the  $i$ -th component of the  $j$ -th realization,  $d_1(\cdot, \cdot)$  is the  $\ell_1$  distance between a point and a set, and  $\text{ReLU}(x) = \max\{0, x\}$ . So, the interval score term calculates the average  $\ell_1$  distance of points in  $\{\mathbf{s}_j\}_{j=1}^m$  to  $\mathcal{R}$ , and it is equal to zero if every  $\mathbf{s}_j$  belongs to  $\mathcal{R}$ . This term is added to avoid extrapolation of the LF model when using encoded HF inputs.

The first term in the loss,  $\mathcal{L}_{\text{Err}}$ , combines MSE contributions from LF and HF models, respectively. The second term  $\mathcal{L}_{\text{Reg}}$  promotes regularization in the non linear correlation network using  $\lambda_H^{nl}$  and the low-fidelity surrogate network using  $\lambda_L$ . In practice, a large penalty  $\lambda_H^{nl}$  shifts the complexity to the coordinate encoder and the linear correlation sub-network. Finally, the *encoding* term  $\mathcal{L}_{\text{Enc}}$  in (2.6) tries to explain as much as possible the HF data using a linear correlation by minimizing the complexity of the nonlinear correlation, and discourages HF input locations that would lead to extrapolation in the LF model response (i.e.,  $x_H$  outside the LF training data bounding box) [5].

**2.4. Optimization Strategy.** When training encoded networks, it is critical to avoid convergence to local minima leading to sub-optimal accuracy. Thus, minimization of (2.6) is achieved with the Adam optimizer (using an exponential learning rate scheduler and an initial learning rate  $\eta_{\max}$ ) by performing a small number of *warm restarts* (in this study we use three restarts in all examples). After 5000 epochs, we restore the learning rate to  $\eta_{\max}$  and continue with the optimization process, similar to cosine annealing [7]. This strategy allows our algorithm to escape from local optima and possibly still find the global optimum. In all the examples in Section 3, both the encoded and *standard* networks (by standard network, we refer to the network proposed in [9]) are trained using 5000 epoch between three successive warm restarts.

**2.5. Hyperparameter optimization.** A systematic comparison between the proposed and standard architecture is performed by varying the number of HF samples, the ratio between LF and HF samples, and for various repetitions. At each repetition, we determine the optimal set of hyperparameters using grid search or hyperopt [1]. Optimized hyperparameters include the learning rate, the LF regularization penalty  $\lambda_L$ , the regularization penalty for the nonlinear correlation sub-network  $\lambda_H^{nl}$ , the penalty  $\lambda_T$  associated with the difference between  $\hat{y}_L$  and  $\hat{y}_L^t$ , and the penalty  $\lambda_D$  for the interval score term. From previous runs we found that a learning rate decay coefficient equal to 0.9999 led to optimal results, and therefore it is kept fixed. The ranges and values selected for the other hyperparameters are reported in Table 2.1 and Table 2.2 for grid search and hyperopt, respectively. For every combination of HF/LF samples and every repetition, the traces for the HF test loss generated by hyperopt are reported in Figure 2.2. Even after 100 iterations, the loss still oscillates with large amplitudes, rising questions about the efficacy of hyperopt that will be further investigated in future work.

Hyperparameter	Encoded MFNN	Standard MFNN
lr	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	$\{10^{-5}, 10^{-4}, 10^{-3}\}$
lr decay	{0.9999}	{0.9999}
$\lambda_L$	$\{10^{-5}, 10^{-4}, 10^{-3}\}$	$\{10^{-5}, 10^{-4}, 10^{-3}\}$
$\lambda_H^{nl}$	{0.05, 0.5, 5.0}	{0.0}
$\lambda_T$	{0.05, 0.5, 5.0}	{0.0}
$\lambda_D$	{5.0}	{0.0}

TABLE 2.1  
Hyperparameter ranges used for grid search.

Hyperparameter	Encoded MFNN		Standard MFNN	
	Type	Range	Type	Range
lr	loguniform	$[\log(10^{-5}), \log(10^{-3})]$	loguniform	$[\log(10^{-5}), \log(10^{-3})]$
lr decay	fixed	{0.9999}	fixed	{0.9999}
$\lambda_L$	loguniform	$[\log(10^{-5}), \log(10^{-3})]$	loguniform	$[\log(10^{-5}), \log(10^{-3})]$
$\lambda_H^{nl}$	choice	{0.05, 0.5, 5.0}	choice	{0.05, 0.5, 5.0}
$\lambda_T$	choice	{0.05, 0.5, 5.0}	fixed	{0.0}
$\lambda_D$	choice	{0.05, 0.5, 5.0}	fixed	{0.0}

TABLE 2.2  
Hyperparameter space definition in hyperopt.

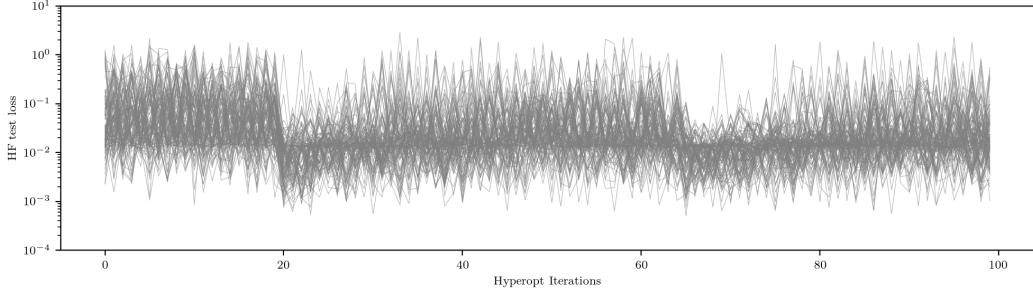


FIG. 2.2. *HF loss traces from hyperopt trials.* The figure shows 120 traces from the two-dimensional test case with equal parameterization for all HF/LF sample combinations and repetitions.

**2.6. Encoded predictions on normalized data.** In this section we discuss how the optimal encoding is affected by data normalization. For synthetic test cases where the true underlying encoding transformation is known, the expressions in this section are used to verify the optimal encoding resulting from training the proposed network. Given a multi-fidelity dataset in the form of a collection of high-fidelity data  $\{(\mathbf{x}_H^{(j)}, \mathbf{y}_H^{(j)})\}_{j=1}^N \subset \mathcal{X}_H \times \mathcal{Y}$  and  $n$  sets of low-fidelity data  $\{(\mathbf{x}_{L,i}^{(j)}, \mathbf{y}_{L,i}^{(j)})\}_{j=1}^{M_i} \subset \mathcal{X}_{L,i} \times \mathcal{Y}$  for some  $i = 1, \dots, n$ , suppose we wish to normalize (or standardize) our data before using it to train our surrogate models. Let this rescaled data be  $\tilde{\mathbf{x}}_{L,i}^{(j)} = W_{\mathcal{X}_{L,i}}(\mathbf{x}_{L,i}^{(j)})$ ,  $\tilde{\mathbf{x}}_H^{(j)} = W_{\mathcal{X}_H}(\mathbf{x}_H^{(j)})$ ,  $\tilde{\mathbf{y}}_L^{(j)} = W_{\mathcal{Y}_L}(\mathbf{y}_{L,i}^{(j)})$ , and  $\tilde{\mathbf{y}}_H^{(j)} = W_{\mathcal{Y}_H}(\mathbf{y}_H^{(j)})$  for some invertible functions  $W_{\mathcal{X}_{L,i}}$ ,  $W_{\mathcal{X}_H}$ ,  $W_{\mathcal{Y}_{L,i}}$ , and  $W_{\mathcal{Y}_H}$ . So, the learned encoders on the rescaled data will be defined such that  $\tilde{T}_i : W_{\mathcal{X}_H}(\mathcal{X}_H) \rightarrow W_{\mathcal{X}_{L,i}}(\mathcal{X}_{L,i})$ . It follows the transformation on the original, unscaled data will be  $T_i = W_{\mathcal{X}_{L,i}}^{-1} \circ \tilde{T}_i \circ W_{\mathcal{X}_H}$ .

For simplicity, consider a bi-fidelity case where the desired encoder is a linear transformation  $T(x) = Ax$  for some matrix  $A$ . Also suppose  $W_{\mathcal{X}_L}$  and  $W_{\mathcal{X}_H}$  are affine transformations such that  $W_{\mathcal{X}_L}(x) = \alpha_L^{-1}(x - \beta_L)$  and  $W_{\mathcal{X}_H}(x) = \alpha_H^{-1}(x - \beta_H)$ . If we calculate explicitly the encoder for the rescaled data as  $\tilde{T} = W_{\mathcal{X}_L} \circ T \circ W_{\mathcal{X}_H}^{-1}$ , we can see that

$$\tilde{T}(x) = (\alpha_L^{-1} A \alpha_H)x + \alpha_L^{-1}(A\beta_H - \beta_L). \quad (2.8)$$

This shows that even though our encoder on unscaled data did not include a bias term, our encoder on scaled data might if  $\alpha_L^{-1}(A\beta_H - \beta_L) \neq 0$ . For this reason, when working with rescaled data, it's important to always include bias term in the encoder network. For all the examples in Section 3, input and output data are standardized before network training.

### 3. Numerical Results.

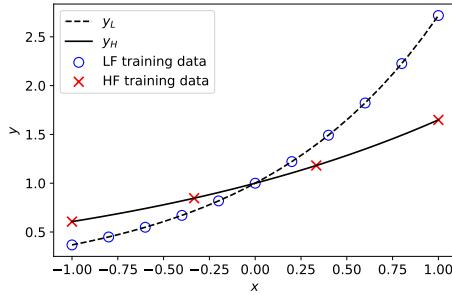
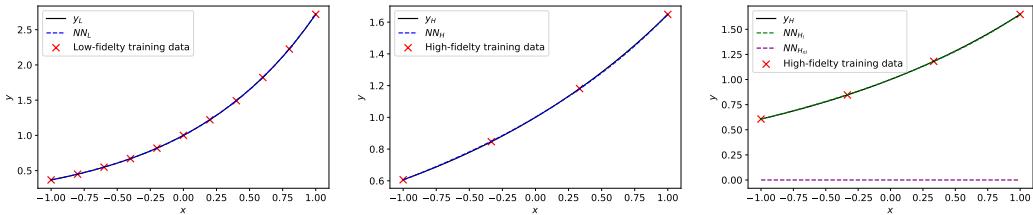
**3.1. Example 1 - Correlated Exponentials.** Recall the function in equations (2.1) shown in Figure 3.1. As discussed in Section 2.1, we can relate the two functions linearly by using an encoder. Providing the training data for the LF and HF models as  $x_L = \{-1, -0.8, \dots, 1\}$  and  $x_H = \{-1, -1/3, 1/3, 1\}$ , respectively, and training the proposed encoded network, we estimate the HF predictor as

$$\hat{y}_H(x) = (2.8966 \cdot 10^{-4})x + (9.9933 \cdot 10^{-1})(y_L \circ T)(x) + 5.3668 \cdot 10^{-5},$$

and

$$T(x) = 0.5013x + 5.6646 \cdot 10^{-4}.$$

Looking at Figure 3.2, we observe both LF and HF predictors result in high accuracy, with negligible non linear correlation.

FIG. 3.1. *LF and HF models for example 1.*FIG. 3.2. *LF (left) and HF (middle) predictors learned by the encoded multi-fidelity network for the correlated exponential example. The predictions from the linear and non linear correlation sub-networks are also shown (right).*

### 3.2. Example 2 - Linearly correlated one-dimensional example.

#### 3.2.1. Problem statement.

Consider the following example from [9],

$$\begin{cases} y_L(x) = 0.5(6x - 2)^2 \sin(12x - 4) + 10x - 10, \\ y_H(x) = (6x - 2)^2 \sin(12x - 4), \end{cases} \quad (3.1)$$

where  $y_H$  is linearly correlated with  $y_L$ . In fact,  $y_H(x) = 2y_L(x) - 20x + 20$ . In this example, we sought to replicate findings from [9] using our encoded network.

**3.2.2. Performance for a fixed number of equally spaced samples.** We first train a network based on the sampling point location specified in [9], obtaining a correlation between LF and HF predictors of the form

$$y_H = 1.9918y_L - 19.8568x + 19.9062, \text{ and } T(x) = 1.0000x - 9.5310 \cdot 10^{-5}.$$

The network learns an identity transformation which gives the correct linear correlation. Looking at Figure 3.3, we see, similar to the last example, the LF and HF predictors are fairly accurate, with a negligible nonlinear correlation between HF and LF approximants. This shows that the proposed architecture can recover the same correlation as those found in [9].

**3.2.3. Training campaign.** To further test the encoded network's ability to replicate the findings from [9], we also trained the encoded network with different sizes of training samples. We considered a HF training set consisting of  $\{4, 8, 12, 16\}$  Sobol' samples, with LF samples added following oversampling ratios equal to 2 and 4 (therefore resulting in 8 combinations of HF/LF samples). A total of 10 repeats is then performed for each combination of HF/LF samples. The test set has an equal number of HF and LF model evaluations,

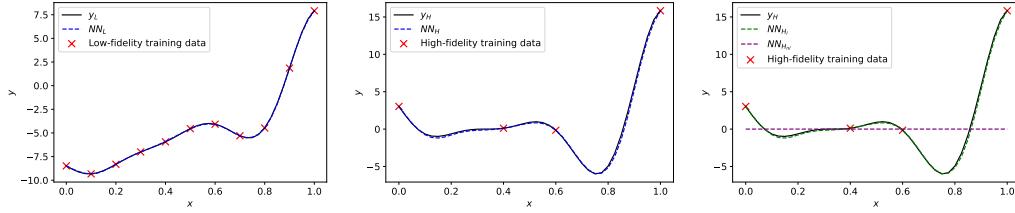


FIG. 3.3. *LF* (left) and *HF* (middle) predictors learned by the encoded multi-fidelity network for example 2. The linear and non linear correlation network predictions are also shown (right).

i.e., uses a training-to-test sample ratio of 1.0. This relates to the small number of samples considered for this one-dimensional benchmark.

The results in Figure 3.4 compare the trends for four different MSE. The test MSE for the LF and for the HF model (this latter used as the objective in hyperparameter optimization) is plotted together with the HF MSE evaluated on a validation grid of 31 equally spaced locations. Additionally, the two-norm of the weights for the nonlinear correlation network is reported.

The proposed network achieves higher accuracy and less variability on a regular validation grid (green triangles), as shown in Figure 3.4. In addition, the two-norm of the weights for the nonlinear correlation network is significantly reduced when using an encoded network, indicating a sensible reduction in its complexity for this sub-network.

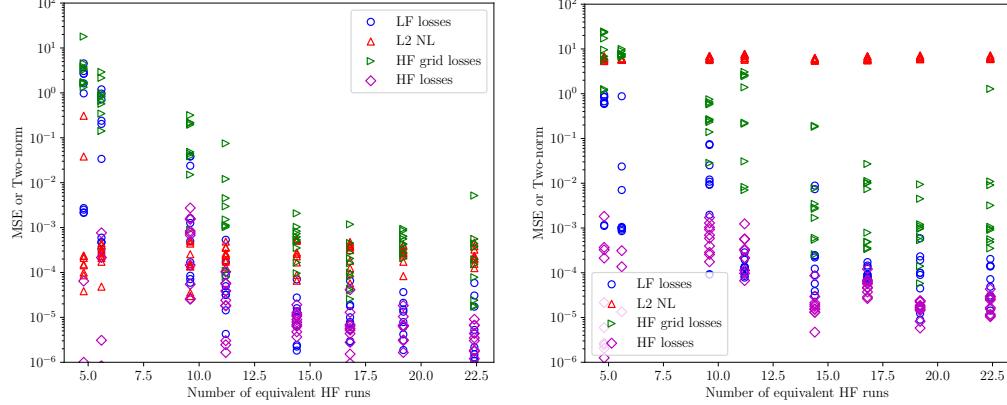


FIG. 3.4. *Simulation campaign for exercise 2.* Losses obtained from the proposed encoded network (left) are compared with those from the standard architecture (right). The equivalent number of HF models on the x axis assumes a 10/1 HF/LF cost ratio.

**3.2.4. Approximation results for HF and LF predictors.** Figure 3.5 shows the quality of the approximation resulting from an increasing number of quasi-randomly selected training points. Stable approximations can be obtained with only 4/16 HF/LF evaluations, consistent with the performance discussed in Section 3.2.2.

### 3.3. Example 3 - Correlated 2D functions with equal parametrization.

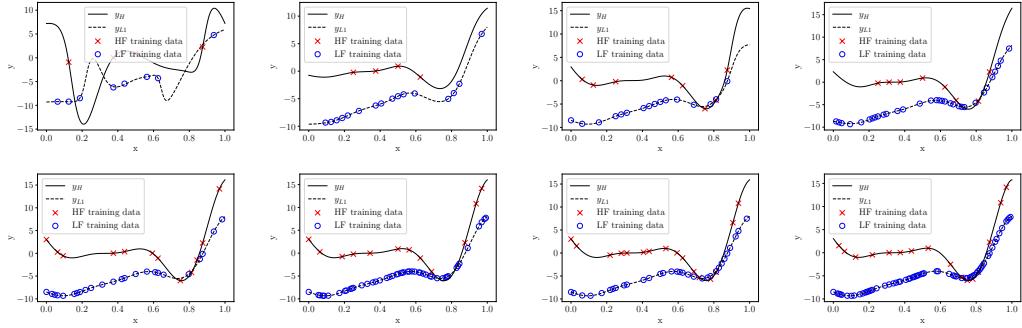


FIG. 3.5. Results of curve fitting for normalized outputs. The number of HF/LF samples (left to right and top to bottom) are 4/8, 4/16, 8/16, 8/32, 12/24, 12/48, 16/32 and 16/64.

### 3.3.1. Problem statement.

Consider the a HF/LF model pair of the form

$$\begin{aligned} y_H(x, y) &= \exp(0.7x + 0.3y) + 0.15 \sin(2\pi x), \\ y_L(x, y) &= \exp(0.01x + 0.99y) + 0.15 \sin(3\pi y). \end{aligned} \quad (3.2)$$

These functions are not linearly correlated yet they can be related exactly using a linear transformation, i.e.  $y_L = y_H \circ T^*$  where

$$T^*(x, y) = \begin{pmatrix} 0 & 1.5 \\ 1/30 & -0.2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \quad (3.3)$$

Note that the HF function is increasing in a direction close to  $x$ , while  $y_L$  varies mainly along  $y$ . In addition, both functions include an oscillatory response on orthogonal directions, with frequencies that differ by 50%. While in their current configuration the HF to LF linear correlation is relatively small, as stated before, the models can be related exactly with the use of the coordinate transformation  $T^*$ . This coordinate transformation is what the encoded network is designed to learn, so we expect it to uncover a more significant linear correlation when compared to the standard network.

**3.3.2. Training campaign.** The number of HF samples is set to  $\{25, 50, 100, 200\}$  with LF oversampling ratios equal to  $\{5, 10, 20\}$ , for a total of 12 HF/LF sample combinations. For each combination, 10 networks are trained and the optimal hyperparameters are determined, for each of these networks, based on a quasi-random testing set with the same number of samples as the training set.

Figure 3.6 reports the same MSEs as discussed for the one-dimensional example in Section 3.2. The resulting MSEs are higher then for the one-dimensional test case, with comparable accuracy resulting from the encoded and standard multifidelity emulators. Finally, the two-norm of the non linear correlation between HF and LF is significantly smaller for the encoded network, which successfully promoted a higher linear correlation.

**3.3.3. Approximation results for HF and LF predictors.** Surface fitting results are illustrated in Figure 3.7. While the HF and LF functions are approximated equally well by the two networks, the linear and nonlinear HF predictors are completely different. The encoded network results in a linearly correlated HF predictor that is almost identical to the true HF function, and a nonlinear HF predictor that is practically zero everywhere. Conversely, a standard network explains the HF predictor using a large nonlinear correlation and a relatively simple linear HF predictor.

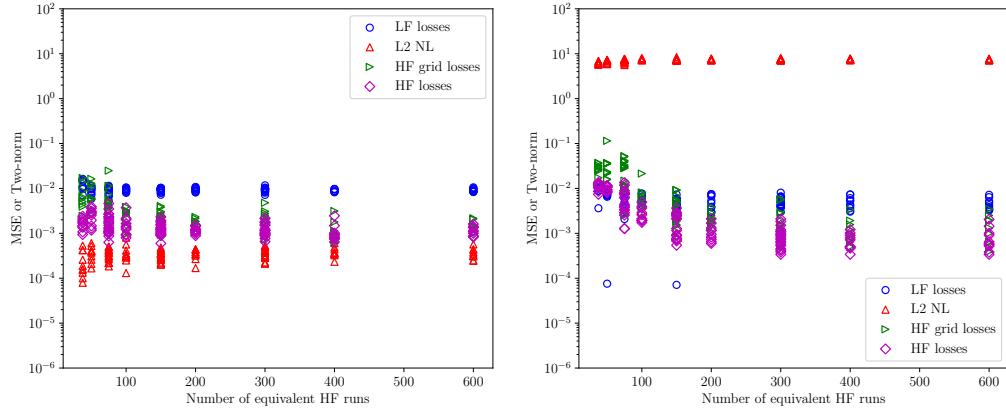


FIG. 3.6. *Simulation campaign for the two-dimensional example with equal number of parameters. The losses resulting from the proposed encoded network (left) are compared with those of the standard architecture (right). The equivalent number of HF models on the x axis assumes a 10/1 HF/LF cost ratio.*

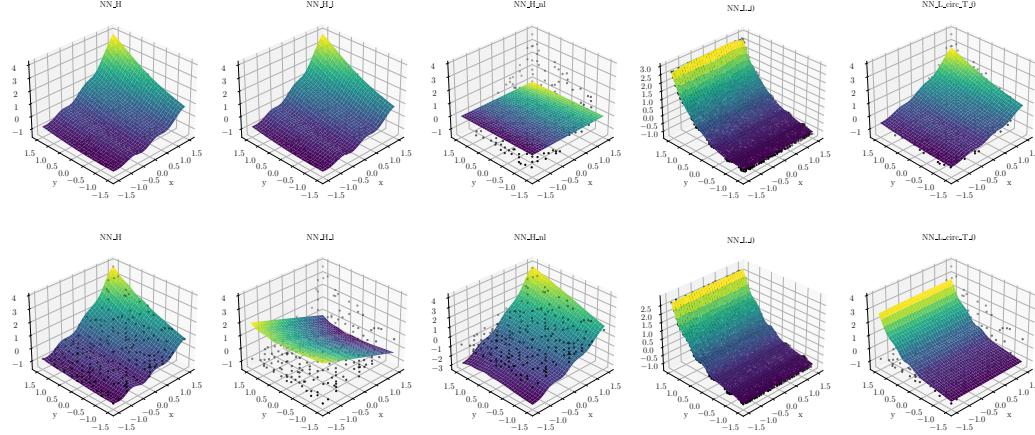


FIG. 3.7. *Results of surface fitting for normalized outputs. All these results are generated with 200 and 4000 HF and LF samples, respectively for the proposed encoded network (top) and standard network (bottom). Starting from left to right on each row, we show the HF approximant, the linearly correlated HF approximant, the non linearly correlated HF approximant, the LF approximant and the LF approximant evaluated at the encoded HF inputs.*

### 3.4. Example 4 - Correlated Functions with dissimilar parametrization.

#### 3.4.1. Problem statement.

Define a HF/LF function pair as

$$\begin{aligned} y_H(x, y, z) &= \exp(0.7z + 0.3y) + 0.15 \sin(2\pi z) + 0.5x^3, \\ y_L(x, y) &= \exp(0.01x + 0.99y) + 0.15 \sin(3\pi x). \end{aligned} \quad (3.4)$$

In this case, the HF inputs are three-dimensional and the LF inputs two-dimensional. A standard architecture cannot be used for this example, since the LF predictor is evaluated at the HF model inputs. An encoder that closes this dimensionality gap is hence necessary.

**3.4.2. Training campaign.** The number of HF, LF samples and repetitions is the same as for the two-dimensional example in Section 3.3. Since the standard network cannot

be used for this example, Figure 3.6 compares predictions with optimal hyperparameters determined through grid search and hyperopt, respectively. The accuracy produced by grid search and hyperopt appear comparable. Hyperopt results in greater variability for the validation losses (computed on a  $31 \times 31 \times 31$  regular grid) due to the larger search space.

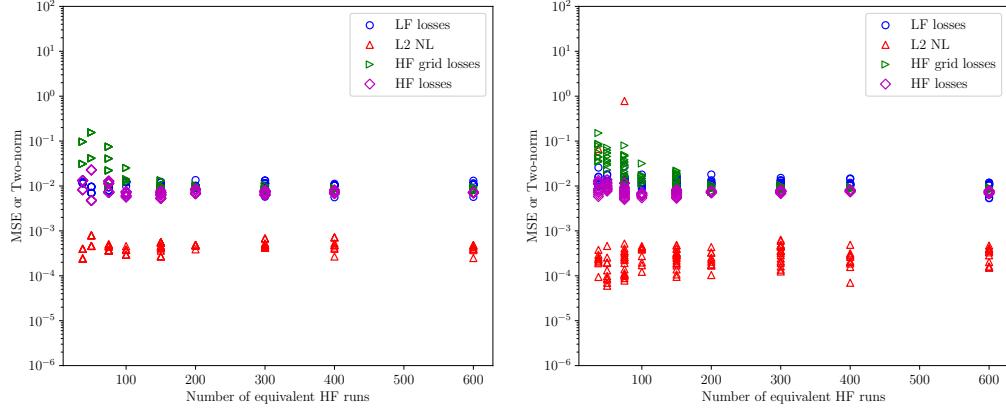


FIG. 3.8. *Simulation campaign for the two-dimensional example with dissimilar parametrization. We compare minimum test losses obtained through grid search (left) and hyperopt (right). The equivalent cost of HF model solutions on the x axis is computed by assuming a 10/1 HF/LF cost ratio.*

**4. Discussion.** We present an encoded multifidelity neural network that achieves better or comparable accuracy with respect to previously proposed architectures, but significantly reduces the complexity of the nonlinear correlations between LF and HF predictor, by leveraging a linear coordinate transformation. This leads to accurate multifidelity data-driven emulators characterized by simpler architecture (i.e., with fewer weights).

Training encoded multifidelity architectures is, in general, more complex than with standard architectures, due to an increased presence of local minima. We mitigate this problem by restarting training three times, each time resetting the learning rate to its original value. Another manifestation of this issue is that optimal results are almost invariably determined for learning rate decay coefficients close to 1.0. Moreover, encoding of HF inputs may lead to extrapolation for the LF predictor and a consequent drop in accuracy. To prevent this possibility, an interval score term is added to the loss function penalizing HF inputs that lie outside the bounding box of the LF training samples.

**5. Conclusion and future work.** The study investigates the effect of introducing a coordinate transformation to maximize the correlation between HF and LF predictors in multifidelity neural emulators. Accurate emulators are obtained for one-, two- and three-dimensional test cases with equal and dissimilar parameterization. Additionally, the proposed approach leads to a sensible reduction in complexity for the non linear correlations between HF and LF predictors. The accuracy we have achieved in the two-dimensional examples is compatible with that obtained from the standard architecture proposed in [9]. This is because the complexity of the non linear correlation between HF and LF predictors can still be captured with a modest network depth and width. However, this may not be the case for simpler networks, more complex nonlinear correlations, and in high dimensions. Future work will focus on such cases. Also, the architecture in [9] has been extended to accommodate multiple types of network configurations, including, recurrent, convolutional, graph and operator networks [6]. In future work, we will investigate the use of encoded networks for other types of architectures. In the present study we have focused on bi-fidelity

examples and linear encoders, but the code that we have developed is designed to handle an arbitrary number of LF sub-networks and arbitrary encoders. We will test this capability in future studies. Finally, hyperparameter optimization is important to train accurate encoded emulators. In our experiments (see, e.g., Figure 2.2) we have found that hyperopt generally provides improved results with respect to combinatorial grid search. However, a significant increment in the number of iterations was found necessary for noticeable reductions in the optimal HF losses. Thus, further comparison between hyperopt and Bayesian optimization [4] is needed for more effective hyperparameter search.

## REFERENCES

- [1] J. BERGSTRA, D. YAMINS, AND D. COX, *Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures*, in International conference on machine learning, PMLR, 2013, pp. 115–123.
- [2] S. DE, J. BRITTON, M. REYNOLDS, R. SKINNER, K. JANSEN, AND A. DOOSTAN, *On transfer learning of neural networks using bi-fidelity data for uncertainty propagation*, International Journal for Uncertainty Quantification, 10 (2020).
- [3] M. DEHGHANI, A. MEHRJOU, S. GOUWS, J. KAMPS, AND B. SCHÖLKOPF, *Fidelity-weighted learning*, in International Conference on Learning Representations, 2018.
- [4] P. I. FRAZIER, *Bayesian optimization*, in Recent advances in optimization and modeling of contemporary problems, Informs, 2018, pp. 255–278.
- [5] T. GNEITING AND A. E. RAFTERY, *Strictly proper scoring rules, prediction, and estimation*, Journal of the American statistical Association, 102 (2007), pp. 359–378.
- [6] A. A. HOWARD, M. PEREGO, G. E. KARNIADAKIS, AND P. STINIS, *Multifidelity deep operator networks*, arXiv preprint arXiv:2204.09157, (2022).
- [7] I. LOSHCHILOV AND F. HUTTER, *Sgdr: Stochastic gradient descent with warm restarts*. arxiv 2016, arXiv preprint arXiv:1608.03983, (2019).
- [8] X. MENG, H. BABAEE, AND G. KARNIADAKIS, *Multi-fidelity bayesian neural networks: Algorithms and applications*, Journal of Computational Physics, 438 (2021), p. 110361.
- [9] 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.
- [10] X. ZENG, G. GERACI, A. GORODETSKY, M. ELDRED, J. JAKEMAN, AND R. GHANEM, *Improving bayesian networks multifidelity surrogate construction with basis adaptation*, AIAA SciTech 2023, (2023), pp. 2023–0917.
- [11] ———, *Multifidelity uncertainty quantification with models based on dissimilar parameters*, Computer Methods in Applied Mechanics and Engineering, 415 (2023), p. 116205.