High-order Discontinuous Galerkin Methods and Deep Reinforcement Learning with Application to Multiscale Ocean Modeling

by

Corbin Foucart

B.S. Stanford University (2015)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Mechanical Engineering
& Center for Computational Science and Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Corbin Foucart
Department of Mechanical Engineering
& Center for Computational Science and Engineering
August 2, 2023

Certified by: Pierre F. J. Lermusiaux
Professor, Department of Mechanical Engineering
Thesis Supervisor

Accepted by: Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

Accepted by: Youssef Marzouk
Professor, Department of Aeronautics and Astronautics
Co-Director, Center for Computational Science and Engineering

High-order Discontinuous Galerkin Methods and Deep Reinforcement Learning with Application to Multiscale Ocean Modeling

by

Corbin Foucart

Submitted to the Department of Mechanical Engineering
& Center for Computational Science and Engineering
on August 2, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

With the expanding availability of computational power, numerical modeling plays an increasingly pivotal role in the field of oceanography, enabling scientists to explore and understand ocean processes which are otherwise inaccessible or challenging to observe directly. It provides a crucial tool for investigating a range of phenomena from large-scale circulation patterns to small-scale turbulence, shaping our understanding of marine ecosystems, global climate, and weather patterns. However, this same wide range of spatiotemporal scales presents a distinct computational challenge in capturing physical interactions extending from the diffusive scale (millimeters, seconds) to planetary length scales spanning thousands of kilometers and time scales spanning millennia. Therefore, numerical and parameterization improvements have and will continue to define the state of the art in ocean modeling, in tandem with the integration of observational data and adaptive methods. As scientists strive to better understand multiscale ocean processes, the thirst for comprehensive simulations has proceeded apace with concomitant increases in computing power, and submesoscale resolutions where nonhydrostatic effects are important are progressively becoming approachable in ocean modeling. However, few realistic ocean circulation models presently have nonhydrostatic capability, and those that do overwhelmingly use low-order finite-difference and finite-volume methods, which are plagued by dispersive errors, and are arduous to utilize in general, especially on unstructured domains and in conjunction with adaptive numerical capabilities. High-order discontinuous Galerkin (DG) finite element methods (FEMs) allow for arbitrarily high-order solutions on unstructured meshes and often out-compete low-order models with respect to accuracy per computational cost, providing significant reduction of dispersion and dissipation errors over long-time integration horizons. These properties make DG-FEMs ideal for the next generation of ocean models, and, in this thesis, we develop a novel DG-FEM ocean model with the above longer-term vision and adaptive multiscale capabilities in mind.

Using a novel hybridizable discontinuous Galerkin (HDG) spatial discretization for both the hydrostatic and nonhydrostatic ocean equations with a free surface, we develop an accurate and efficient high-order finite element ocean model. We emphasize the stability and robustness properties of our schemes within a projection method discretization. We provide detailed benchmarking and performance comparisons for the parallelized implementation, tailored to the specifics of HDG finite element methods. We demonstrate that the model achieves optimal convergence, and is capable of accurately simulating nonhydrostatic behavior. We evaluate our simulations in diverse dynamical regimes including linear gravity waves, internal solitary waves, and the formation of Rayleigh-Taylor instabilities in the mixed layer. Motivated by investigating local nonhydrostatic submesoscale dynamics using realistic ocean simulation data, we develop schemes to initialize and nest the new DG-FEM model within a comprehensive hydrostatic ocean modeling system. Nested within such data-assimilative hydrostatic simulations in the Alboran Sea, we provide a demonstration of our new model's ability to capture both hydrostatic and nonhydrostatic dynamics that arise in the presence of wind-forced instabilities in the upper ocean layers. We show that such a model can both validate and work in tandem with larger hydrostatic modeling systems, enabling multi-dynamics simulations and enhancing the predictive fidelity of ocean forecasts.

Next, as DG-FEM methods are well-suited to adaptive refinement, we develop a method to learn new adaptive mesh refinement strategies directly from numerical simulation by formulating the adaptive mesh refinement (AMR) process as a reinforcement learning problem. Finite element discretizations of problems in computational physics can usefully rely on adaptive mesh refinement to preferentially resolve regions containing important features during simulation. However, most spatial refinement strategies are heuristic and rely on domain-specific knowledge or trial-and-error. We treat the process of adaptive mesh refinement as a local, sequential decision-making problem under incomplete information, formulating AMR as a partially observable Markov decision process. Using a deep reinforcement learning (DRL) approach, we train policy networks for AMR strategy directly from numerical simulation. The training process does not require an exact solution or a high-fidelity ground truth to the partial differential equation (PDE) at hand, nor does it require a pre-computed training dataset. The local nature of our deep reinforcement learning approach allows the policy network to be trained inexpensively on much smaller problems than those on which they are deployed, and the DRL-AMR learning process we devise is not specific to any particular PDE, problem dimension, or numerical discretization. The RL policy networks, trained on simple examples, can generalize to more complex problems and can flexibly incorporate diverse problem physics. To that end, we apply the method to a range of PDEs relevant to fluid and ocean processes, using a variety of high-order discontinuous Galerkin and hybridizable discontinuous Galerkin finite element discretizations. We show that the resultant learned policies are competitive with common AMR heuristics and strike a favorable balance between accuracy and cost such that they often lead to a higher accuracy per problem degree of freedom, and are effective across a wide class of PDEs and problems.

Thesis Supervisor: Pierre F. J. Lermusiaux
Title: Professor, Department of Mechanical Engineering

Acknowledgments

> The truth will set you free. But not until it is finished with you.
> — David Foster Wallace, *Infinite Jest*

There are a number of people who have been instrumental in my completion of this thesis, and to whom I am grateful.

I feel an enormous sense of gratitude to Pierre for his guidance throughout my time at MIT as well as his tireless work ethic. There's an apocryphal story I like about a wise and battle-hardened engineer called out of retirement as a consultant to fix a very expensive but malfunctioning machine. The engineer looks at the machine design and, after a minute or two of thought, draws with his pencil a small "x" on the part that needs to be replaced. After the replacement, the machine works perfectly. However, upon receiving a bill for half the original cost of the machine, the indignant vice president of the company demands an itemization. The engineer sends back a two-line invoice: fifty cents for the cost of the pencil and the rest for "knowing where to draw the x." Thank you, Pierre, for giving me the independence and space to work on my own ideas and mature as a researcher, but at the same time for always knowing exactly where the "x" should go.

I'd like to thank my committee for their comments and feedback. In particular, I'd like to thank Professor Peraire, Professor Patera, and Dr. Cuong Nguyen for meeting with me repeatedly over the course of my PhD to discuss research ideas and finite element methods more broadly. I'd like to thank Dr. Chris Mirabito for his near-infinite patience when I began my work on this project, not to mention his substantial contributions in proofreading the details of my work the past few years. Similarly, I owe a debt of gratitude to the legendary Dr. Pat Haley for more than living up to the title of "resident wise, old guy;" many of the ideas in these pages have directly benefited from your vast experience and bottomless well of debugging ideas. We are very grateful to the Office of Naval Research for support throughout this research and Ph.D. under grants N00014-15-1-2626 (DRI-FLEAT), N00014-18-1-2781 (DRI-CALYPSO), and N00014-20-1-2023 (MURI ML-SCOPE), the Defense Advanced Research Projects Agency (DARPA) for support under grant N66001-16-C-4003 (POSYDON), and the National Science Foundation for support under grants OCE-1061160 (ShelfIT) and EAR-1520825 (NSF-ALPHA), as well as to the Massachusetts Institute of Technology for making this research experience possible. Graduate school has taken me around the world and given me the opportunity to pursue anything and everything that I found worthwhile. I will always look back fondly on this period of my life.

Academically, I'm especially grateful to professors Boyd, Broderick, Marzouk, and Patera for their teaching—your ability to unpretentiously break down and distill complicated technical material to the point where it seems intuitive and easy has always impressed me. I'd also like to thank Dr. Alexander Linke of the Weierstrass Institute for his incredible mentorship during my time in Berlin—your zeal for applied math is infectious, and my choice to pursue research in computational physics was largely due to my time spent working with you. To Hui-Min, thank you for the years of advice, brutal honesty, friendship, and for pushing me, not only musically, but in nearly every aspect of life. You're the best mentor I've ever had and you've profoundly influenced me for the better.

To the friends I've made over the years at MSEAS, thank you for all the SGTs. Abhinav, thank you for being a continual grounding presence in my life and being the best friend I could possibly ask for. Aaron, I owe you a big "oh, danke" for the all the great vocalizations, animal noises, and memories traveling. To Arko, thank you for never failing to call me out when I believe something with too much certainty, for always knowing the shortest walking path between two points on campus, and for appreciating the nutritional importance of fried chicken. Manan, thank you for all the coding and statistical discussion over the years—on top of being an exceptional project partner, you may be the only other person I'll ever meet who appreciates Casella, Berger, and numerical software design as much as I do. To Manny, thank you for all the money you've given to me over the years in bets won; although you will always make me *so* mad, what you lack in wagering prowess you more than make up for in personality, music taste, and frisbee ability. Aditya, thank you for all your diligent work on the HDG project—it's rare to find someone who's as competent yet as easygoing as you are, and I'll miss working with you. Tony, our spontaneous coffee trips have always been a welcome break from life's other obligations; may you continue to be loved by all people and feared by all geese. To Jing, your accipitrine attention to detail will always be an inspiration to me. Thank you for all the numerical intuition you gave me as a new PhD student and for the many enjoyable swim workouts in the "p-pool." And lastly, to JVo, I'll always look back fondly on our years working together; thanks for all the late nights spent debugging, drinking seltzer, and listening to Yuja Wang and Joshua Bell rip through the Kreutzer sonata, not to mention the great times in Miami, New England, Montreal, and not Albuquerque, where things are basically post-apocalyptic and where I would never visit you. Lisa, despite the fact that everyone loves you, you're still under-rated; thank you for all the admin work, laughs, and snacks. To my fellow night-owl John, thanks for keeping our building spotless and for the engaging 4AM conversations—I hope you enjoy a restful and well-deserved retirement! To the rest of you: thank you for the countless memories and for always making MSEAS feel like home.

I feel doubly fortunate to have made many friends during my time at MIT outside of academics. To Cindy, David, Emily, Florian, Zach, Haekyung, and Diana, thank you for the rewarding musical moments with Beethoven, Franck, Schubert, and Bach. To Tim, Misha, Peter and the other kick-boxers, thank you for pushing me physically, many times to the ground (ha), and perhaps for pushing me spiritually too. To Ben and Ved, I can't tell you how much fun it's been hitting the pavement with you two—thank you for nudging me out of my comfort zone. To Emma, Julian, Vivian, Nick, Harry, and Icey, thanks for the many coffee trips, regulatory compliance pain smiles, and laughs over the strange pastries and questionably-devised drinks of Kowloon Tong, TST, and LKF. I'd like to thank Emma in particular for being such a detail-oriented and patient language exchange partner; although I still don't think you'll ever look at the pictures you insisted on taking of every dog, flower, and piece of food we encountered, I'll always cherish our friendship. To Nayun, a profound thank you for helping me overcome my crippling fear of "being nice."

To Kevin, Evan, Michael William Locke, Nicky T., Richie, and Nate: words can't describe how great it's been growing up with all of you over the past 15 years. Through all of the many "camping" trips, poker and nighttime glow-in-the-dark ultimate frisbee games, themed parties, terrible movie nights, Blue Moon calzones, Castle Island runs, beach days, games of deception, weddings... although a lot has changed over the years, one thing stays the same—you guys are the best.

And, of course, I'd like to thank my amazing biological family, to whom I owe the gift of life itself. Thank you, Mom and Dad; everything I've accomplished has been in no small part due to the love, support, and guidance you've given me over the years. Thanks to my sister Abbey for keeping me centered and for all the excellent times at the Friendly Toast diner, and thanks to Esme for selflessly defending me from all the menacing air-conditioning units, autumn leaves, and stationary shadows of Boston, MA.

Contents

Chapter 1

Introduction

Partial differential equations provide an excellent governing mathematical model of many physical phenomena and are foundational in the modern scientific understanding of our physical environment. However, despite knowing the underlying governing equations of the phenomena we seek to understand, it is typically not possible to solve them analytically. Therefore, the numerical approximation of the solutions to these partial differential equations by using computers is essential to model and understand the physical phenomena they describe. There is, consequently, a vast amount of modern mathematical and scientific research on methods in computational physics. Computational fluid dynamics (CFD) is a subset of this field which holds a central role in both scientific research and engineering due to its ability to quantitatively predict complex fluid flow phenomena. CFD provides a valuable tool for understanding the intricacies of fluid mechanics, enables weather, ocean, and climate forecasting, provides cost-effective and safe methods for testing and optimizing engineering designs, and allows the simulation and study of atmospheric and ocean processes, among countless other applications.

With the expanding availability of computational power, numerical modeling plays an increasingly pivotal role in the field of oceanography, enabling scientists to explore and understand ocean processes which are otherwise inaccessible or challenging to observe directly. It provides a crucial tool for investigating a range of phenomena from large-scale circulation patterns to small-scale turbulence, shaping our understanding of marine ecosystems, global climate, and weather patterns. However, this same wide range of spatiotemporal scales presents a distinct computational challenge in capturing physical interactions extending from the diffusive scale (millimeters, seconds) to planetary length scales spanning hundreds of thousands of kilometers and time scales spanning millennia (Figure 1-1). The direct numerical discretization of the governing equations to capture the physics at these scales for the entire ocean without further approximations and parameterizations would require computers about 10 billion times faster and bigger in storage than present supercomputers—these remain (very optimistically) about two centuries in the future [82]. Therefore, numerical and parameterization improvements have and will continue to define the state of the art in ocean modeling, in tandem with the integration and co-analysis of observational data [83, 155]. The advances in this thesis are a direct contribution towards these efforts.

1.1 Nonhydrostatic dynamics

Ocean modelers are primarily motivated by the problem of predicting the dynamic evolution from an initial state to the state at a future time. At the *synoptic* weather scale (on the order of 1,000 km in the horizontal, Figure 1-1, top right), oceanographic flows are largely two-dimensional. The depth-to-length aspect ratio of the ocean modeling domains over these these circulation length scales is very small, differing by (typically 3-4) orders of magnitude. To draw a visual comparison for clearer understanding, this is roughly equivalent to the aspect ratio corresponding to the thickness of a sheet of paper. Moreover, at these scales, the effects of Earth's rotation and density stratification impart additional vertical rigidity to the flow [56]. As a result, the vertical component of the fluid velocity

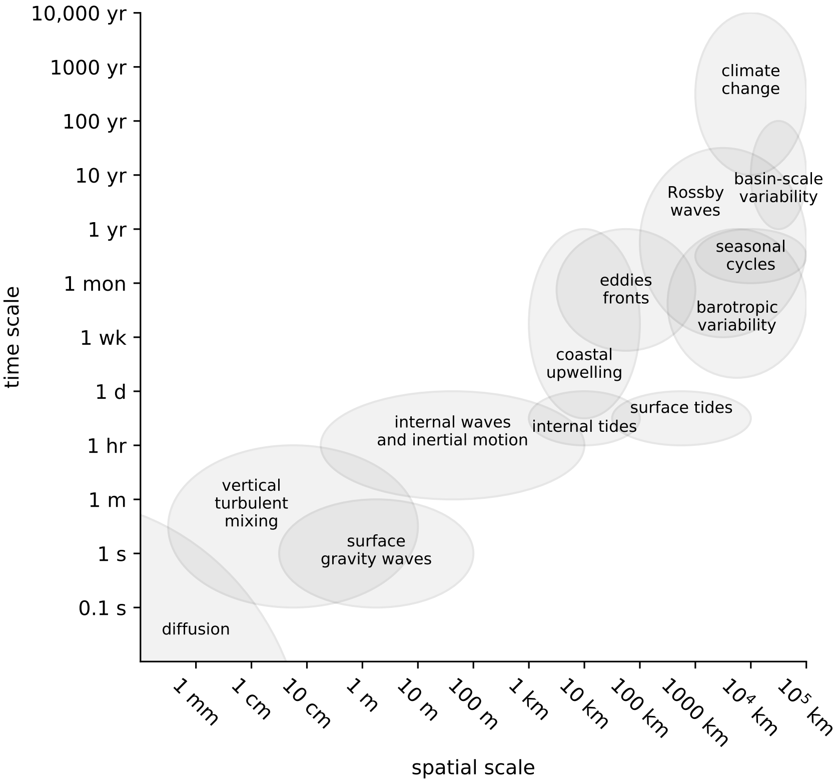


Figure 1-1: Scales of motion for ocean processes (reproduced from D. Chelton).

is typically much smaller than the horizontal components, vertical accelerations are small compared to those in the horizontal, and the fluid physics at these modeling scales is largely hydrostatic, *i.e.*, the vertical pressure profile at any point in the ocean is well-approximated as that of a static fluid. Similarly, in the horizontal, ocean dynamics at these scales are well-described by geostrophic balance, where the Coriolis forces arising due to the rotation of the Earth are balanced by the horizontal pressure gradient of the flow. Even at the smaller *mesoscale* (on the order of 10-100 km in the horizontal), oceanic flows are represented by quasi-equilibrium states, in near-hydrostatic and near-geostrophic balance [172, 56]. Therefore, traditional ocean modeling aimed at understanding the dynamics at these scales has relied on the hydrostatic approximation, which makes accurate estimation of the horizontal pressure field, and thus geostrophic flows, computationally tractable [83]; this is discussed in depth in §3. Hydrostatic models have demonstrated a remarkable ability to accurately simulate and predict the complex fluid dynamics at these scales, allowing accurate weather forecasting, climate modeling, and understanding oceanic currents. In that regard, these models represent a remarkable success story in the field of scientific modeling and prediction [209, 208].

However, as scientists strive to better understand the processes in the ocean, the unquenchable thirst for larger simulations has proceeded apace with concomitant increases in computing power, and scales where nonhydrostatic effects become important are beginning to be resolved in regional and process models. At the *submesoscale* (on the order of less than 10km in the horizontal) the hydrostatic approximation begins to lose validity, along with the hydrostatic ocean equations (§3.2, (3.9)). Nonhydrostatic effects become increasingly strong and presently are detectable at the small-scale end of the submesoscale [108, 172, 237, 240] and in deep convection [178]. At these scales, much smaller-scale nonhydrostatic and fully three-dimensional oceanic phenomena (typically on order of 100 m to 1 km), such as the breaking of internal waves, convection, subduction, and shear-induced overturning, become important [172, 180, 240]. As a result, oceanic features with relatively strong vertical velocities have been observed at these scales, particularly in the presence of wind forcing or topographic variations [119, 171]. In turn, these dynamics play a crucial role in dissipative *diapycnal mixing*. This is when transient, interior regions of small-scale turbulence mix fluids of different densities. The effects of these submesoscale motions are largely unknown [83, 173], but must be investigated. This is because these nonhydrostatic dynamics may not only affect oceanic diapycnal transport, the dissipation of energy input at the surface, the spontaneous breakdown of mesoscale structures, and vertical transport in the ocean, but may also constitute a link between non-dissipative mesoscale flows and dissipative three-dimensional motions [84, 172, 174].

Few ocean circulation models presently have nonhydrostatic capability; we provide a literature review in §3. While the lack of nonhydrostatic-capable methods is partially due to the fact that advances in computing have only relatively recently enabled the resolution of scales in which nonhydrostatic submesoscale dynamics play a role, it is also attributable to the fact that the numerical solution of the nonhydrostatic ocean equations (§3.2, (3.1 - 3.4)) is significantly more challenging to compute than its hydrostatic counterpart. The need to solve for the nonhydrostatic component of the pressure globally couples the 3D problem, often necessitating the solution of an incompressible Navier–Stokes equation with a free surface, under the Boussinesq approximation [178, 252]. The dominant technologies for solving problems of this nature are overwhelmingly low-order finite difference and finite volume methods. However, while commonly-used, modern advances in CFD suggest that high-order methods may be better suited to the problem of nonhydrostatic ocean modeling.

On one hand, high-order finite element methods allow for arbitrarily high-order solutions on unstructured meshes and often provide higher accuracy for the same computational cost [110]. General unstructured meshes can more accurately capture complex geometrical features such as coastlines and steep bathymetry. Unstructured meshes also allow open boundaries to be moved further away from the model domain of interest for a small computational cost by using much larger elements near open boundaries, reducing the impact of the open boundary condition on the simulation. However, for dispersive wave behaviors such as those that occur at the submesoscale, evidence suggests that high-order methods may have additional advantages. In [251], the authors show that for low-order finite volume methods specifically, the dominant mode of truncation error is dispersive, leading to pollution of the physically correct dispersion for internal solitary waves, imposing a strict resolution requirement on the method. The authors propose both high-order methods and adaptive mesh refinement as potential solutions. In [244], the authors indeed showed that higher-order schemes were capable of modeling physical-biogeochemical dynamics with the correct biological patches, cycles, and nonlinear behaviors while lower-order schemes could not.

1.2 Discontinuous Galerkin finite element methods

In light of the requirements of a nonhydrostatic model discussed above, high-order spectral and finite element methods are natural candidates for the next generation of multiscale ocean models [63, 62, 161]. Spectral schemes, while providing high-order accuracy, are difficult to make conservative and may spontaneously generate local extrema [83]. A more significant drawback is the difficulty applying spectral element methods to general unstructured domain geometries. Continuous Galerkin finite element methods (CG-FEM), on the other hand, have been successfully to problems with arbitrary unstructured geometries in many areas of computational physics, perhaps most notably in solid mechanics. However, classical CG-FEM schemes are not without their drawbacks. The choice of a continuous finite element polynomial space require the application of hanging node constraints in the case of adaptive refinement. Moreover, despite the natural formulation of second-order PDEs in the CG-FEM context, special treatment is required to ensure conservativity, and the methods need to rely on additional stabilization mechanisms such as streamline-upwind diffusion in the case of transport-dominated regimes and in under-resolved settings [133]. These issues, while surmountable with special care, have largely limited the applicability of CG-FEM methods to problems in CFD.

However, in recent decades, discontinuous Galerkin (DG) finite element methods have emerged as one of the most important discretization schemes for problems in computational physics. Originally developed for first-order hyperbolic conservation laws, DG-FEM methods have enjoyed a wide range of successes modeling physical systems arising in acoustics [15, 241], electrodynamics [46, 51], and the shallow water equations [257, 62, 71, 92, 258], among many others. Recent advances have applied DG-FEM to second order PDEs as well, making DG methods a competitive and attractive choice for computational fluid dynamics as well. An overview of the history and the most important application areas can be found in §2, [45, 110], and in the references therein.

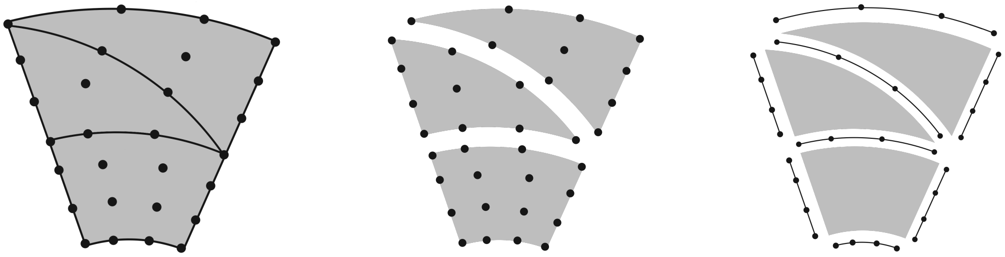


Figure 1-2: Globally coupled degrees of freedom for CG-FEM (left), DG-FEM (center), and HDG-FEM (right).

Discontinuous Galerkin methods derive their high-order accuracy from the variational principles used in finite and spectral elements, and search for optimal solutions within a particular function space. Instead of strongly enforcing continuity over the element boundaries as in conventional finite element methods, DG methods seek an approximation on each element which is not obligated to coincide with that of the neighboring elements on shared edges, leading to a discontinuity in the solution representation (Figure 1-2). This counter-intuitive formulation allows for the addition of physics-informed numerical fluxes that enjoy many of the advantages of finite volume methods, ensure consistency and conservativity, and enable high-order accuracy on complex domains with general unstructured meshes without the need to maintain hanging nodes constraints as with CG-FEM in an adaptive refinement setting. The discontinuous representation of the numerical solution also provides a natural stabilization mechanism and allows for the graceful capture of steep gradients in advection-dominated flows, discussed at length in §2 and §5. As compared to both low-order methods as well as higher order CG-FEM methods, DG-FEM discretizations significantly reduce dispersion and dissipation errors in the pre-asymptotic regime over long time integration horizons, the main source of inaccuracy in long-time simulations of flows at higher Reynolds numbers or wave propagation problems [133].

As discontinuous Galerkin elements are also well-suited to parallelism and modern computer architectures [134], it would seem that this class of methods is virtually without drawback. However, inspection of the degrees of freedom for CG-FEM as compared to DG-FEM in the schematic shown in Figure 1-2 shows that the discontinuous Galerkin methods involve a duplication of globally-coupled degrees of freedom along element edges. For discretization of the diffusion operators using a local discontinuous Galerkin (LDG) or symmetric interior penalty Galerkin (SIPG) method these additional degrees of freedom can become expensive [12]. As a result, hybridizable discontinuous Galerkin (HDG) finite element methods were first introduced for second-order elliptic problems in Cockburn [42] and later for advection-diffusion problems [191] in order to mitigate the cost of the discontinuous representation of the solution. HDG schemes involve a parameterization of the PDE onto a finite element space defined only on the mesh skeleton, such that the globally coupled unknowns have support on the element interfaces only. These methods possess the advantages of a DG discretization, as well as favorable convergence properties with local post-processing, but at the cost of solving a linear system similar to classical CG-FEM. These advantageous features make HDG finite element methods the discretization of choice for the novel nonhydrostatic model developed in §3 of this thesis. However, as hybridization alone is insufficient to overcome the expensive nature of nonhydrostatic problems, we also provide theory and schemes adapting HDG-FEM to a matrix-free and parallel context as additional contributions.

1.3 Model nesting and use of realistic ocean data

While a high-order nonhydrostatic ocean model based on DG-FEM discretizations is a substantial advance in its own right in terms of investigating submesoscale dynamics, its scientific utility extends beyond functioning as a standalone solver. Unlike experimental physics, a numerical model rarely indicates if processes are left out. Another contribution of this thesis is the development a procedure to nest the nonhydrostatic model developed in §3 within a larger, hydrostatic one. This, along with the HDG hydrostatic model also developed as part of this thesis, will allow future comparison investigations to be done as to when the resolution of nonhydrostatic submesoscale processes is crucial to regional dynamics, and when it is not. These contributions will pave the way for multi-model solvers [228], as well as allow for the discovery of parameterizations to incorporate nonhydrostatic physics into less expensive hydrostatic ocean models natively.

It is through experimentation and observation that the biases in models and processes that have been overlooked are brought to light. Additionally, systems for forecasting and state estimation are becoming increasingly valuable as they provide context and inference from observational data [83]. Data assimilation allows the control of errors in numerical modeling systems and their initial and boundary conditions, as well as inference and discoveries by the rigorous combination of information from data and models [222, 221, 223, 159] Therefore, while a crucial component of realistic ocean modeling and real-time ocean forecasting [104, 162, 184, 160, 187, 181], assimilation of observations into high-order discontinuous finite element models is in its infancy [205]. Therefore we also present a preliminary investigation into the incorporation of data assimilation techniques into DG-FEM solvers in a CFD context in §4.

1.4 Reinforcement learning for adaptive mesh refinement

Finite element discretizations of problems in computational physics often rely on adaptive mesh refinement (AMR) to preferentially resolve regions containing important features during simulation. As we've alluded to earlier in this chapter, AMR provides a potential solution to capturing localized nonhydrostatic phenomena in state of the art ocean models. However, these spatial refinement strategies are often heuristic and rely on domain-specific knowledge or trial-and-error. Furthermore, the inherent complexity of ocean models presents a significant obstacle to the straightforward implementation of conventional adaptive mesh refinement techniques. These models' distinctive dynamics necessitate solutions that are specifically designed to cater to their unique behavior.

In this context, the prospect of using machine learning approaches appears promising. Rather than deploying a one-size-fits-all technique, unsupervised methods have the potential to learn efficient error estimators, and hence, AMR strategies directly from simulation data. Approaches that enable strategies to be learned experientially from simulation directly are even more beneficial, as they eliminate the need for a high-fidelity ground truth dataset. Such a self-learning approach could furnish a more refined and accurate depiction of the physical behavior specific to the partial differential equation (PDE) under study. With these requirements in mind, deep reinforcement learning presents as a compelling solution to this challenge.

Therefore, as the final contribution of this thesis, we develop a novel method that applies deep reinforcement learning to learn adaptive mesh refinement strategies directly from simulation [81]. We treat the process of adaptive mesh refinement as a local, sequential decision-making problem under incomplete information, formulating AMR as a partially observable Markov decision process. While the ultimate aim is to provide custom-trained AMR strategies for ocean dynamics, this approach can be more generally deployed to any problem involving the numerical solution of PDEs solved using DG finite element methods. The training process does not require an exact solution or a high-fidelity ground truth to the PDE at hand, nor does it require a pre-computed training dataset. The local nature of our deep RL (DRL) allows the policy network to be trained inexpensively on much smaller problems than those on which they are deployed, and the resultant policy networks, trained on simple examples, can generalize to more complex problems, and can flexibly incorporate diverse problem physics.

1.5 Thesis organization

This thesis is structured as follows. We begin by providing the derivations for the DG-FEM and HDG-FEM methods that we will make use of for our new high-order nonhydrostatic model in §2. We provide implementation details, benchmarking, and results related to specific issues that arise in our applications: a novel matrix-free method extending the work in [80], a benchmarking and discussion addressing the solution of singular systems arising from the discretization of the Poisson equation with pure Neumann boundary conditions, and a computational examination of the effect of polynomial order and multi-threaded parallelism applied to the HDG algorithms specifically. In §3, we formulate and derive a novel HDG/DG spatial discretization of the hydrostatic and nonhydrostatic ocean equations with a surface, extending the work started in [243, 246, 247] and emphasizing stability and robustness properties of our schemes within a projection method discretization. We verify the model with analytical convergence studies. We then validate it with results from linear gravity wave theory for both surface and internal gravity waves, and from nonlinear solitary waves [79]. We provide additional verification of the model via comparison to a two-dimensional finite-volume model [245, 156] on an idealized test case examining the development of idealized Rayleigh-Taylor instabilities in the mixed layer. Lastly, we describe the procedure for nesting our new model within the larger, MSEAS primitive equation (PE) modeling system. We show results from nested model runs in both 2D and 3D from realistic ocean data in the Alboran Sea as part of the MSEAS CALYPSO sea experiment [174, 173, 181]. In §4, we study the application of uncertainty quantification techniques to HDG-FEM numerical solvers in an idealized CFD setting with implementations of the ensemble Kalman Filter for data assimilation, and Markov chain Monte Carlo methods applied to the Bayesian inversion problem of an unknown diffusivity field, extending a dimension reduction method [179] to our HDG solvers. In §5, we formulate our novel DRL-AMR method and test it over a wide variety of numerical test cases [81], showing the method to be competitive with or better than many common AMR heuristics. Lastly, in §6, we offer conclusions, discussion, and outlook on future research directions.

Chapter 2

Hybridizable Discontinuous Galerkin Methods: Theory, Schemes, and Efficiency

2.1 Introduction

This chapter provides expository information of finite element methods inspired by the material in [39, 110, 196], and is intended to give a brief exposition of the theory and ideas behind DG-FEM schemes more broadly. These ideas and concepts are readily applicable to HDG schemes and provide motivation for the all of the numerical choices made in later chapters. The schemes used for linear ODEs have direct application to the depth-integration schemes used in the ocean equations in §3. The RKDG methods have application to the discretization of the advection term in the momentum equation in §3, and lay the groundwork for the schemes and methods introduced in §5. Lastly, the HDG methods introduced in this chapter are used to discretize all the implicit terms present in the nonhydrostatic ocean model in §3.

A secondary goal of this chapter is to provide conceptual unification for all the different schemes, as well as substantive discussion of implementation and computational considerations, including novel work. First, it is often stated in the literature that the embarrassingly parallel nature of the assembly and reconstruction steps of the HDG methods are an advantage of the method. However, it is the case that this statement is dependent computationally on problem size and polynomial order, which we explore in §2.3.7. In §2.4, we provide theory and schemes for a novel matrix-free HDG method, extending and expanding the work started in [80]. Furthermore, we discuss the solution of the pure Neumann problem in §2.5, as it arises in fluid applications; while different approaches to treating the resultant singular system have been discussed in the literature, we provide a novel discussion of their computational trade-offs and provide benchmarking comparisons. To the author's knowledge, the investigations in each of these sections constitute the first of their kind.

It is the opinion of the author that one of the most rewarding aspects of applied mathematics is layering trivial steps together, until, a few innocuous steps down the line, you've suddenly created something completely non-trivial. Overall, this chapter builds up the pieces of an efficient HDG "kernel," which in-turn forms the foundation for numerical schemes used to solve fluid and ocean problems [193, 246, 247]. In doing so, this chapter serves to provide a unified look at the theory, schemes, and computational considerations that will justify the mathematical and numerical choices made in later chapters.

2.2 Discontinuous Galerkin finite element methods

The discontinuous Galerkin finite element method (DG-FEM) was first proposed as a method to solve the steady-state neutron transport equation [218], but has since developed into an entire numerical ecosystem capable of solving a wide variety of problems in computational physics. The success of discontinuous Galerkin finite element methods is largely attributable to combining the best features of finite volume and finite element schemes: DG methods provide high-order accuracy, can represent complex geometries, and admit both implicit and explicit semi-discrete forms. Additionally, the discontinuous polynomial spaces in which DG-FEM solutions are sought allow for the capture of steep gradients and wave behavior, often resulting in more stable and flexible methods than the classical continuous Galerkin finite element (CG-FEM) approaches to advection-dominated problems [110].

We begin with some notes on the classification of the different portions of the DG-FEM ecosystem to provide context to the work in this thesis. Discontinuous Galerkin methods were originally developed for hyperbolic conservation laws, and have enjoyed popularity for a multitude of applications based on hyperbolic systems: acoustics [15, 241] , Maxwell's equations [46, 51], and the shallow water equations [62, 71, 161], to name a few. DG-FEM approaches to solving elliptic problems using *primal methods* began with the introduction of interior penalty methods [11, 220], and was extended to the class of *mixed methods* by writing the second-order spatial derivatives as a system of first-order equations [21, 34]. The extension to elliptic problems has allowed for the more recent application of DG-FEM to advection-diffusion problems, and both compressible and incompressible viscous flow problems, such as the mathematical models considered in §3.1. An examination and unified analysis of the properties and differences between these methods can be found in the well-known manuscript by Arnold and Brezzi [12]. Because the theory and practice of DG-FEM has become so expansive, in what follows, we exposit the key findings and implementation details that are specific to the work in the subsequent chapters.

2.2.1 Linear ODEs

Suppose we would like to use DG-FEM to approximate the numerical solution of an ordinary differential equation (ODE). Consider the initial-value problem on the domain $\Omega = (0, L)$:

$$\frac{du(x)}{dx} = f(x)u(x), \qquad x \in \Omega, \quad u(0) = u_0, \tag{2.1}$$

for which we would like to find a numerical approximation u_h. Partition the domain into N intervals $I_n = (x_n,\, x_{n+1})$ for $n = 0, \ldots, N-1$; the intervals are the one-dimensional "elements". On each I_n we seek a polynomial of at most degree p. Let $\mathcal{P}^p(D)$ be the set of polynomials of at least p on a domain D. Then u_h is an element of the set

$$W_h^p = \left\{ w \in L^2(\Omega) \colon w\big|_{I_n} \in \mathcal{P}^p(I_n),\, n = 0, \ldots, N-1 \right\}. \tag{2.2}$$

We would like to formulate the problem in a weak sense. Namely, we multiply the entire equation by an arbitrary function $\mathsf{v} \in W_h^p$ and integrate by parts over each interval I_n. The integration by parts results in

$$\int_{I_n} \frac{d}{dx}\left(u_h(x)\right) \mathsf{v}\, dx = -\int_{I_n} u_h \frac{d\mathsf{v}}{dx}\, dx + \left[u_h \mathsf{v}\right]_{x_n}^{x_{n+1}}, \tag{2.3}$$

and we seek the function $u_h \in W_h^p$ which satisfies

$$-\int_{I_n} u_h \frac{d\mathsf{v}}{dx}\, dx + \left[u_h \mathsf{v}\right]_{x_n}^{x_{n+1}} = \int_{I_n} f(x) u_h(x) \mathsf{v}(x)\, dx \tag{2.4}$$

for all $\mathsf{v} \in W_h^p$. However, there is a problem. Since u_h consists of discontinuous polynomials on each I_n, there is not a unique value for u_h at either interface, implying that the quantity $[u_h]_{x_n}^{x_{n+1}}$ is not well-defined. We don't have the same problem with v, because we aren't solving for v; we only want

u_h to satisfy equation (2.4) for all $\mathsf{v} \in W_h^p$. This point is not pedantic; if we chose the interior values of u_h at the boundaries of I_n, that is,

$$u_h(\cdot) = \begin{cases} \lim_{\epsilon\to 0} u_h(x^n + \epsilon), & \text{at } x_n \\ \lim_{\epsilon\to 0} u_h(x_{n+1} - \epsilon) & \text{at } x_{n+1} \end{cases}, \tag{2.5}$$

there would be no coupling of information between each of the intervals I_n; we would have N completely independent equations—one for each interval—and we would be unable to solve the resultant system of equations.

Something special must be done about the value of u_h at each interface; we create a new quantity $\widehat{u}_h$, the "numerical flux" or "numerical trace", a representative value of the solution on the edge between the two elements. We re-write equation (2.6) as

$$-\int_{I_n} u_h \frac{d\mathsf{v}}{dx}\, dx + [\widehat{u}_h \mathsf{v}]_{x_n}^{x_{n+1}} = \int_{I_n} f(x) u_h(x) \mathsf{v}(x)\, dx, \tag{2.6}$$

Which we are able to sum over all elements in order to form a globally-coupled problem, once we have defined the numerical flux $\hat{u}_h$ in terms of the interior unknowns. The question of how to define $\widehat{u}_h$ is central to DG-FEM methodology. Different choices yield different schemes and different results. To start, we make the simple choice

$$\widehat{u}_h(x_n) = \begin{cases} u_0 & \text{if } x_n = 0 \\ \lim_{\epsilon\to 0} u_h(x_n - \epsilon) & \text{otherwise} \end{cases}, \tag{2.7}$$

that is, we take the left-hand side neighbor's value of u_h. This choice was arbitrary, in the sense that we chose to propagate information from left to right. If the ODE were in time, this choice would be good, since information propagates as time increases. However, by making either choice, but we have solved the issues with the definition in equation (2.5); information is coupled between the intervals and the problem is now solvable. However, we would like to do better than simply be able formulate a problem that's solvable. Namely, we would like to choose $\widehat{u}_h$ in a way such that the scheme is *consistent*, in the sense that the exact solution u should satisfy equation (2.6), and *stable*, in the sense that the total variation of the numerical solution u_h remains bounded as the discretization size $h = L/N$ goes to zero. We define the notation

$$u_h^{\pm}(x) = u_h(x^{\pm}) = \lim_{\epsilon\to 0} u_h(x \pm \epsilon) \tag{2.8}$$

$$[\![u_h]\!] = u_h^- - u_h^+ \qquad \{\{u_h\}\} = \frac{1}{2}\left(u_h^- + u_h^+\right) \tag{2.9}$$

for the left and right traces u_h^- and u_h^+, respectively, as well as the interface jump $[\![u_h]\!]$ and average $\{\{u_h\}\}$ terms.

Addressing consistency, we replace u_h with the exact solution u in equation (2.6) and we see that the scheme is consistent if $\widehat{u}_h = u$; otherwise, we could choose a v such that equation (2.6) is not satisfied. Loosely speaking, since we started from the ODE and applied legal operations, most reasonable choices of $\widehat{u}_h$ will preserve consistency, but a consistent numerical flux doesn't guarantee a good scheme, or even stability; even though equation (2.5) is a consistent choice of numerical flux, it doesn't lead to a solvable scheme. The general approach to handle this is to add terms to the formulation (by choice of numerical flux) such that we have a provably stable scheme without violating consistency. Common ways to do this are by adding combinations of jump terms and average terms at the boundaries, because if the solution u is sufficiently smooth, $[\![u]\!] = 0$ and $\{\{u\}\} = u$. Sometimes, it also is possible to incorporate problem physics into the choice of numerical flux, to give the scheme additional favorable properties. A detailed discussion of these sorts of numerical fluxes is given in [110].

It turns out that the choice of numerical flux in equation (2.7) is both consistent and stable, but proving stability can be non-trivial. The central idea used in proofs of stability is to come up with stability criterion for the ODE generally and choose $\widehat{u}_h$ such that the stability properties are guaranteed for the DG method in equation (2.6). If we multiply the ODE by u and integrate over the domain, we obtain the global inequality

$$\frac{1}{2}u^2(T) - \frac{1}{2}u_0^2 = \int_0^T f(x)u^2(x)\,dx \tag{2.10}$$

from which the L^∞ stability of the solution follows from boundedness of the final-time solution $u^2(T)$. We choose $\mathsf{v} = u_h$ in the DG formulation equation (2.6), and sum over all intervals I_n;

$$\begin{aligned}\int_0^L f(x)u_h^2(x)\,dx &= \sum_{n=0}^{N-1}\left(-\int_{I_n} u_h\frac{du_h}{dx}\,dx + [\widehat{u}_h u_h]_{x_n}^{x_{n+1}}\right)\\ &= \sum_{n=0}^{N-1}\left(-\frac{1}{2}u_h^2 + \widehat{u}_h u_h\right)\Big|_{x_n}^{x_{n+1}},\end{aligned} \tag{2.11}$$

where we have used

$$u\frac{du}{dx} = \frac{1}{2}\frac{d}{dx}\left(u^2\right).$$

We have that

$$\frac{1}{2}u_h^2(L^-) - \frac{1}{2}u_0^2 = \frac{1}{2}\frac{d}{dx}\int_0^L u_h^2\,dx = \frac{1}{2}\frac{d}{dx}\left\|u_h\right\|_2^2 \tag{2.12}$$

Then,

$$\Theta_h(L) = -\frac{1}{2}u_h^2(L^-) + \frac{1}{2}u_0^2 + \sum_{n=0}^{N-1}\left(-\frac{1}{2}u_h^2 + \widehat{u}_h u_h\right)\Big|_{x_n}^{x_{n+1}} \tag{2.13}$$

and if we can choose the numerical flux $\widehat{u}_h$ such that $\Theta_h(L) \geq 0$, it would imply existence and uniqueness of the numerical solution u_h. That's because the problem is linear, and if $\Theta_h(L) \geq 0$, then for the choice of $f = 0$, the only possible solution is $u_h = 0$; those two pieces guarantee existence and uniqueness, as well as discrete stability. We make the definition

$$u_h(x) = u_0, \qquad x < 0, \tag{2.14}$$

and rewrite $\Theta_h(L)$ as

$$\begin{aligned}\Theta_h(L) = &-\frac{1}{2}u_h^2(L^-) + \left(-\frac{1}{2}u_h^2(L^-) + \widehat{u}_h(L)u_h(L^-)\right) + \sum_{n=1}^{N-1}\left(\frac{1}{2}\left(u_h^2\right)^-\Big|_{x_n} - \widehat{u}_h(x_n)u_h^-(x_n)\right)\\ &+ \sum_{n=1}^{N-1}\left(-\frac{1}{2}\left(u_h^2\right)^+\Big|_{x_n} + \widehat{u}_h(x_n)u_h^+(x_n)\right) - \left(-\frac{1}{2}u_h^2(0^+) + \widehat{u}_h(0)u_h(0^+)\right) + \frac{1}{2}u_0^2\\ = &-\frac{1}{2}u_h^2(L^-) + \left(-\frac{1}{2}u_h^2(L^-) + \widehat{u}_h(L)u_h(L^-)\right) + \sum_{n=1}^{N-1}\left(-\frac{1}{2}[\![u_h^2]\!] + \widehat{u}_h[\![u_h]\!]\right)\Big|_{x_n}\\ &- \left(-\frac{1}{2}u_h^2(0^+) + \widehat{u}_h(0)u_h(0^+)\right) + \frac{1}{2}u_0^2.\end{aligned} \tag{2.15}$$

We use the relation in equation (7.1), $[\![u_h^2]\!] = 2\{\{u_h\}\}[\![u_h]\!]$, as well as the expansion

$$\begin{aligned}\frac{1}{2}u_h^2(0^+) - \widehat{u}_h(0)u_h(0^+) + \frac{1}{2}u_0^2 &= \frac{1}{2}u_h^2(0^+) + \frac{1}{2}u_h^2(0^+) - \frac{1}{2}u_h^2(0^+) - \widehat{u}_h(0)u_h(0^+) + \frac{1}{2}u_h^2(0^-)\\ &= -\left(\widehat{u}_h(0) - u_h(0^+)\right) + \frac{1}{2}[\![u_h^2]\!],\end{aligned} \tag{2.16}$$

yielding the massaged expression for $\Theta_h(L)$:

$$\begin{aligned}\Theta_h(L) = &\left(\widehat{u}_h(L) - u_h(L^-)\right)u_h(L^-)\\ &+ \sum_{n=1}^{N-1}\left((\widehat{u}_h - \{\{u_h\}\})[\![u_h]\!]\right)\Big|_{x_n} - \left(\widehat{u}_h(0) - u_0\right)u_h(0^+) + \frac{1}{2}[\![u_h]\!]^2(0)\end{aligned} \tag{2.17}$$

Now, if we define the numerical trace as

$$\widehat{u}_h = \begin{cases} u_0, & x_n = 0 \\ \{\{u_h\}\} + C_n[\![u_h]\!], & x_n \in (0, L) \\ u_h(L^-), & x_n = L, \end{cases} \tag{2.18}$$

where $C_n \geq 0$, then if we chose $C_0 = 1/2$,

$$\Theta_h(L) = \sum_{n=0}^{N-1} C_n[\![u_h]\!]^2 \Big|_{x_n} \geq 0 \tag{2.19}$$

we have $\Theta_h(L)$ as desired, and hence stability. We see that a sufficient choice of numerical flux can enforce the stability of the DG method. The numerical flux $\widehat{u}_h$ is indeed a combination of average and jump values and depends only on $u_h^{\pm}(x_n)$ at each interface; these properties enforce consistency, and any choice of $C_n \geq 0$ will result in both a stable and consistent scheme. If we take all $C_n = 1/2$, we recover the flux in equation (2.7), $\widehat{u}_h = u_h^-(x_n)$, see (7.2). The choice of C_n affects the properties of the method. If we use $C_n = 1/2$ and $\widehat{u}_h = u_h^-$, we can solve the ODE explicitly, element by element from left to right. If $C_n \neq 1/2$, the problem must be implicitly solved on the whole domain $(0, T)$ at once by inverting a linear system. The numerical flux also affects the overall accuracy of the method; $C_n = 0$ leads to a scheme of order $2p + 2$ [64], but the choice of $C_n = 1/2$ results in an order $2p + 1$ accurate scheme [163]—the price of the extra order of accuracy is having to solve an implicit problem. There are two important properties of the DG methods here that are also true in the multi-dimensional case and for all problems. DG-FEM solutions are *piecewise discontinuous*: there are no continuity constraints between elements. However, there is a relation between between the residuals of u_h on each element and the jumps in the numerical solution on element boundaries. Let

$$R_h(x) = \frac{d}{dt}u_h - fu_h \tag{2.20}$$

denote the residual of the ODE. If we take $\mathsf{v} = 1$, and integrate by parts once more, using $C_n = 1/2$ and equation (7.2), we can write equation (2.6) as

$$\begin{aligned} \int_{I_n} R_h \, dx &= -\left(u_h - \widehat{u}_h\right)\Big|_{x_n}^{x_{n+1}} \\ &= -u_h^-(x_{n+1}) + u_h^-(x_{n+1}) + \left[u_h^-(x_n) - u_h^+(x_n)\right] \\ &= [\![u_h]\!](x_n). \end{aligned} \tag{2.21}$$

That is, the jump of the numerical solution u_h at x_n is equal to the integral of the numerical residual R_h over the element. The implication is that if the ODE is well-approximated on I_n, the jump at x_n is small; conversely, if the ODE is not well-approximated locally, the jumps will be large. The jump terms $[\![u_h]\!]$ can loosely be thought of as dampers which can stabilize the DG method whenever the ODE cannot be well-approximated [39]. In more complicated problems solved in higher dimensions, the residual is a more complicated linear function of the jumps but the underlying idea is the same. DG-FEM solutions are also *locally conservative*. For this simple ODE, taking $\mathsf{v} = 1$ in equation (2.6) yields

$$\widehat{u}_h\Big|_{x_n}^{x_{n+1}} = \int_0^L fu \, dx, \tag{2.22}$$

which shows that the flux in minus the flux out is equal to the forcing term. We will see that in hyperbolic settings, the flux in minus the flux out will reduce to zero. Numerical schemes with this conservative property are particularly important for computational fluid dynamics.

2.2.2 Notation and approximation spaces

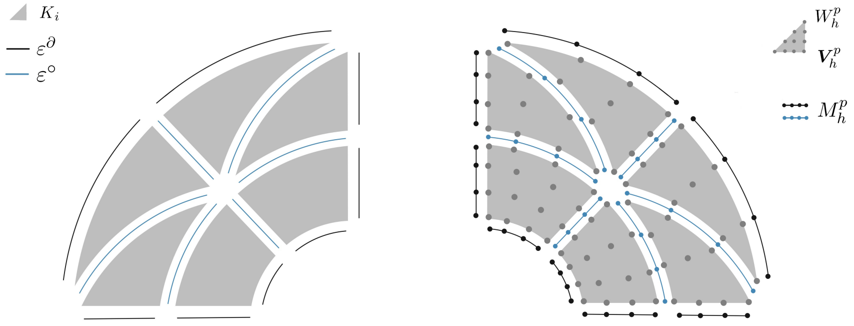
The ideas in §2.2.1 are foundational to DG-FEM, and it was sufficient to write out all intervals and integrals explicitly. However, as we develop more complicated schemes in two and three spatial dimensions, it will be beneficial to have unified notation and terminology. We now provide the notation for the computational mesh and finite-element operators in what follows. We let $\mathcal{T}_h = \cup_i K_i$ be a finite collection of non-overlapping elements K_i that discretizes the entire problem domain $\Omega \subset \mathbb{R}^d$. We refer to the boundary of the problem domain as Γ. The set $\partial\mathcal{T}_h = \{\partial K : K \in \mathcal{T}_h\}$ refers to all boundary edges and interfaces of the elements, where ∂K is the boundary of element K. For two elements K^+ and K^- sharing an edge, we define $e = \partial K^+ \cap \partial K^-$ as the edge between elements K^+ and K^-. Each edge can be classified as belonging to either ε° or ε^∂, the set of interior and boundary edges, respectively, with $\varepsilon = \varepsilon^\circ \cup \varepsilon^\partial$. These geometric relationships are summarized in Figure 2-1a.

The elements K^+ and K^- have outward pointing unit normals $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$, respectively. The quantities $a^\pm$ denote the traces of a on the edge e from the interior of $K^\pm$. When relevant for element-wise operations, we take as convention that the element K^- refers to the local element, and K^+ to the neighboring element. The jump $[\![\cdot]\!]$ operator for scalar quantities are then defined as $[\![a]\!] = a^- - a^+$ on the interior faces $e \in \varepsilon^\circ$. On the edges described by $\partial\mathcal{T}_h$, we can uniquely define the normal vector $\boldsymbol{n}$ as outward for a given cell and inward for its neighbor.

We define the inner products over a set $D \subset \mathbb{R}^d$ and its boundary $\partial D \subset \mathbb{R}^{d-1}$ using typical discontinuous Galerkin finite element notation as

$$(c, d)_D = \int_D cd \, \mathrm{d}D, \qquad \langle c, d \rangle_{\partial D} = \int_{\partial D} cd \, \mathrm{d}\partial D.$$

Let $\mathcal{P}^p(D)$ denote the set of polynomials of degree p on a domain D. For any element $K \in \mathcal{T}_h$,



(a) Domain geometry

(b) Finite element spaces

Figure 2-1: Summary schematics of spatial discretization notation in exploded view.

we denote $W^p(K) \equiv \mathcal{P}^p(K)$ and $\boldsymbol{V}^p(K) \equiv [\mathcal{P}^p(K)]^d$. We consider the discontinuous finite element spaces

$$\begin{aligned}
W_h^{p_{\text{order}}} &= \left\{ w \in L^2(\Omega) \colon w\big|_K \in \mathcal{P}^{p_{\text{order}}}(K) \, \forall K \in \mathcal{T}_h \right\}, \\
V_h^{p_{\text{order}}} &= \left\{ \boldsymbol{v} \in \left[L^2(\Omega)\right]^d : \boldsymbol{v}\big|_K \in [\mathcal{P}^{p_{\text{order}}}(K)]^d \, \forall K \in \mathcal{T}_h \right\}, \\
M_h^{p_{\text{order}}} &= \left\{ \mu \in L^2(\varepsilon_h) : \mu\big|_e \in \mathcal{P}^{p_{\text{order}}}(e) \, \forall e \in \varepsilon_h \right\},
\end{aligned}$$

We introduce the traced finite element space in anticipation of its use in the HDG sections to come. However, we remark that it typically does not play a role in classical DG finite element schemes, such as the methods we introduce in the next section.

2.2.3 Nonlinear conservation laws

The advection equation is an example of a general hyperbolic conservation law, typically written in the form

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0, \tag{2.23}$$

where $f(u)$ is a nonlinear function. Specific choices of the (possibly tensor-valued) function $f(u)$ can represent the Euler equations of gas dynamics, the maxwell equations, and the advection equation, to name a few examples [110]. The numerical approximation to the exact solution of these systems is made difficult by the potential presence of discontinuities in the exact solution. The physically relevant solution to the advection equation, for example, is called the *entropy solution* and can be obtained with so-called *monotone* schemes; however, these schemes are only first-order accurate and poorly capture moving discontinuities. High-order methods are often plagued by spurious oscillations around the discontinuities which can lead to convergence to a different solution than the entropy solution due to the nonlinearity of the underlying problem.

The *high-resolution* schemes are methods developed to provide both high-order accuracy and convergence to the entropy solution; examples include the monotone upstream-centered schemes for conservation laws (MUSCL), weighted essentially non-oscillatory (WENO) schemes, and limiter methods. These methods work by ensuring that a numerical flux, sometimes called an *approximatie Riemann solver* in this context, is chosen such that the conservation laws are enforced locally and the resulting method is a monotone scheme. If the unknown is a scalar and a piecewise-constant approximation is used, these two properties, together, ensure that the high-order accurate solution is stable and converges to the entropy solution. However, if the numerical solution is not piecewise constant, the stability of the method no longer follows from the form of the numerical fluxes and instead has to be enforced with a *flux* or *slope limiter* method [139], which is applied as post-processing after a time step to render the method stable.

Although successful, the high resolution methods are not easily able to handle complex geometries and boundary conditions while achieving high-order accuracy, as is typical of finite volume methods. Classical CG-FEM schemes for nonlinear conservation laws do not enforce conservation locally, nor do they satisfy stability properties like total variation boundedness (TVB). Furthermore, they give rise to systems of nonlinear equations that have to be solved implicitly and are very computationally expensive [30].

Runge–Kutta discontinuous Galerkin (RKDG) schemes [44, 47, 48, 49, 50] make a compromise between the high resolution schemes and CG-FEM approaches by incorporating the local conservation and slope limiting into a finite element framework.

2.2.4 RKDG methods

An overview of RKDG schemes can be found in the review paper [50], from where we summarize the main ideas applicable to the problems and numerical schemes considered in this thesis. For the spatial discretization of any conservation law in the form of (2.23), we seek $u_h \in W_h^p$ such that

$$\left(\frac{\partial u_h}{\partial t}, \mathsf{v}\right)_{\mathcal{T}_h} - \left(f(u_h), \nabla \mathsf{v}\right)_{\mathcal{T}_h} + \left\langle \widehat{f}(u_h) \cdot \boldsymbol{n}, \mathsf{v}\right\rangle_{\partial \mathcal{T}_h} = 0 \tag{2.24}$$

for all $v \in W_h^p$ on every $K \in \mathcal{T}_h$. The numerical flux $\widehat{f}(u_h)$ (approximate Riemann solver) is incorporated into the method locally. As this is first time we have written a weak form in common DG-FEM notation, we remark that the position of the test function $(\cdot, \mathsf{v})$ as compared to $(\mathsf{v}, \cdot)$ is completely immaterial, as this is simply shorthand for an integral inner product. We will make use of both notations in this thesis. The semi-discretization of the resulting system of ordinary differential equations can be written as

$$\mathcal{M} \frac{du_h}{dt} = \mathcal{F}(u_h), \tag{2.25}$$

where $\mathcal{M}$ is the mass matrix $(\cdot, v)_{\mathcal{T}_h}$ and $\mathcal{F}(u_h)$ is the (nonlinear) operator containing the remainder of the discretization. Due to the discontinuous nature of the approximation space W_h^p, the mass matrix M is block diagonal and hence easily invertible; for convenience, we multiply equation (2.25) by the inverse mass matrix $\mathcal{M}^{-1}$ to obtain a more convenient expression of equation (2.25),

$$\frac{d}{dt}u_h = \mathcal{L}(u_h), \tag{2.26}$$

where the nonlinear operator $\mathcal{L} = \mathcal{M}^{-1}\mathcal{F}(u_h)$, corresponds to the residual in §7.5. We discretize the resulting ODE system in equation (2.26) using an explicit RK method with stages $s = 1, \ldots, S$ (for details, see §7.5) to integrate u_h at time k to $k+1$:

$$u_h^{(s)} = \sum_{i=1}^{s-1} b_i k_i, \qquad k_i = u_h^{(i)} + c_i \Delta t \mathcal{L}_h\left(u_h^{(i)}\right) \tag{2.27}$$

concluding by setting $u_h^{k+1} = u_h^S$. To ensure stability, each intermediate value k_i should be such that $|u_h^{(i)}| \geq |k_i|$ in some semi-norm $|\cdot|$, because as a consequence, $|u_h^k| \geq |u_h^{k+1}|$. After each stage of time integration, RKDG methods often apply a generalized slope limiter $\Lambda\Pi_h$, a nonlinear projection operator devised so that $|u_h^{(i)} \geq |\Lambda\Pi_h k_i|$, ensuring stability. The explicit RK method modified to include the slope limiter is

$$u_h^{(s)} = \Lambda\Pi_h\left(\sum_{i=1}^{s-1} b_i k_i\right), \qquad k_i = u_h^{(i)} + c_i \Delta t \mathcal{L}_h\left(u_h^{(i)}\right) \tag{2.28}$$

When high-order polynomials are used to represent the solution, a high-order RK method that matches the accuracy of the spatial discretization should also be employed. In these cases, use of slope limiters is *crucial* to ensure the stability of the method in the presence of shocks or steep gradients, however the use of a slope limiter can be avoided in cases where the solution is known to be sufficiently smooth [50].

One of the advantages of the RKDG methods is that they are highly local—the only information required to evolve the PDE in time is information from adjacent elements, due to the spatial discretization and the explicit time marching.

2.2.5 The linear advection equation

The choice $f(u) = \boldsymbol{c}u$ leads to the linear advection equation where the known advective velocity field $\boldsymbol{c}(\boldsymbol{x})$, with $\boldsymbol{x} \in \Omega$, is presumed to be divergence-free. This equation is sometimes confusingly called the "transport equation" and is different than the original steady-state problem introduced in [218], as well as the equation considered in §2.2.9. We write the first-order linear PDE as

$$\frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{c}u) = 0, \tag{2.29}$$

along with suitable initial and boundary conditions. Direct substitution for $f(u)$ in equation (2.24) results in the weak form of the advection equation on each element, where we seek u_h such that

$$\left(\frac{\partial u_h}{\partial t}, \mathsf{v}\right)_K - (\boldsymbol{c}u_h, \nabla\mathsf{v})_K + \langle\widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v}\rangle_{\partial K} = 0 \tag{2.30}$$

for all $v \in W_h^p$ on every $K \in \mathcal{T}_h$. To complete definition of the method, we must define the numerical trace $\widehat{\boldsymbol{c}u_h}$ on ∂K. It can be proven that the method is consistent and stable if we choose a numerical flux of the form

$$\widehat{\boldsymbol{c}u_h} = \boldsymbol{c}\,\{\{u_h\}\} + C[\![u_h\boldsymbol{n}]\!]. \tag{2.31}$$

If we take C to be a constant, we can write the numerical flux equivalently as

$$\widehat{\boldsymbol{c}u_h} = \begin{cases} \boldsymbol{c}u_D, & e \in \Gamma_{in} \\ \boldsymbol{c}\left\{\!\left\{u_h\right\}\!\right\} + C[\![u_h\boldsymbol{n}]\!], & e \in \varepsilon^\circ \\ \boldsymbol{c}u_h, & e \in \Gamma_{out} \end{cases} \tag{2.32}$$

taking account of the boundaries. If we take

$$C = \frac{1}{2}|\boldsymbol{c}\cdot\boldsymbol{n}|, \tag{2.33}$$

we have the classical *upwind* flux. If instead we choose $C = \frac{1}{2}|\boldsymbol{c}|$, we obtain the *Lax-Friedrichs* numerical flux,

$$\widehat{\boldsymbol{c}u_h} = \boldsymbol{c}\left\{\!\left\{u_h\right\}\!\right\} + \frac{1}{2}|\boldsymbol{c}|[\![u_h\boldsymbol{n}]\!]. \tag{2.34}$$

The DG-FEM schemes are locally conservative, since equation (2.30) must hold for all $\mathsf{v} \in W_h^p$ and in particular for constant $\mathsf{v} = 1$, implying

$$\left(\frac{\partial u_h}{\partial t}, 1\right)_K + \langle\widehat{\boldsymbol{c}u_h}\cdot\boldsymbol{n}, 1\rangle_{\partial K} = 0 \tag{2.35}$$

and, hence, local conservativity. We can employ a suitable numerical flux and an appropriate explicit time marching scheme to obtain an RKDG scheme for linear advection.

2.2.6 Forming the discrete operators

We turn our attention to the discretization of the weak form in equation (2.30).

We can write the inner product $b(u, \mathsf{v})$ as the action of a discrete operator B on a vector of nodal coefficients u_h

$$b(u,\mathsf{v}) = (\boldsymbol{c}u_h, \nabla\mathsf{v})_K = Bu_h = B_{ij}u_j,$$

where we have expanded the numerical solution u_h in terms of the nodal the basis functions. Namely $u_h = u_j\theta_j$ where the index $j = 1, \ldots, n_b$ denotes the basis function over the mesh element K, and the integral operator can be written

$$\begin{aligned}(\boldsymbol{c}u_j\theta_j, \nabla\mathsf{v})_K &= \int_K u_j\theta_j\boldsymbol{c}\cdot\nabla\mathsf{v}\,dK \\ &= \int_K \left(c_x\frac{\partial\theta_i}{\partial x} + c_y\frac{\partial\theta_i}{\partial y}\right)\theta_j u_j\,dK \\ &= \underbrace{\int_{\widehat{K}} \left[c_x\left(\frac{\partial\widehat{\theta}_i}{\partial\xi}J_{11}^{\text{inv}} + \frac{\partial\widehat{\theta}_i}{\partial\eta}J_{12}^{\text{inv}}\right) + c_y\left(\frac{\partial\widehat{\theta}_i}{\partial\xi}J_{21}^{\text{inv}} + \frac{\partial\widehat{\theta}_i}{\partial\eta}J_{22}^{\text{inv}}\right)\right]\widehat{\theta}_j|J(\boldsymbol{x})|\,d\widehat{K}}_{B_{ij}}\ u_j\end{aligned} \tag{2.36}$$

for two dimensions specifically, where J^{inv} denotes the inverse Jacobian of the transformation from the master element to the physical element K. We discretize the equations with a quadrature rule consisting of weights w_q, and the quadrature points $\boldsymbol{\xi}_q$, $q = 1, \ldots, n_q$, over the master element $\widehat{K}$ and their corresponding locations $\boldsymbol{x}(\boldsymbol{\xi}_q)$ on the physical element K as

$$\begin{aligned}B_{ij} \approx \sum_q w_q \Bigg[c_x&\left(J_{11}^{\text{inv}}(\boldsymbol{\xi}_q)\frac{\partial\widehat{\theta}_i}{\partial\xi}(\boldsymbol{\xi}_q) + J_{12}^{\text{inv}}(\boldsymbol{\xi}_q)\frac{\partial\widehat{\theta}_i}{\partial\eta}(\boldsymbol{\xi}_q)\right) \\ &+ c_y\left(J_{21}^{\text{inv}}(\boldsymbol{\xi}_q)\frac{\partial\widehat{\theta}_i}{\partial\xi}(\boldsymbol{\xi}_q) + J_{22}^{\text{inv}}(\boldsymbol{\xi}_q)\frac{\partial\widehat{\theta}_i}{\partial\eta}(\boldsymbol{\xi}_q)\right)\Bigg]\widehat{\theta}_j(\boldsymbol{\xi}_q)|J(\boldsymbol{x}(\boldsymbol{\xi}_q))|.\end{aligned}$$

Expressing this operator as a series of matrix and vector operations is possible, but becomes awkward and unwieldy due to the many sets of indices and the need to distinguish Hadamard products from matrix products. Instead, we turn to the language of tensor contraction, which can express the products concisely.

Let the indices k, l denote the coordinate directions on the master element ξ_k and on the physical element x_l, respectively. Let the index q index the quadrature point and weight, and the indices i, j index the basis function defined on the element K. Then we may introduce the tensors Θ, S, J^{-1}, J and C, along with the first-order tensor w such that

$$\begin{aligned} &\Theta_{qj} \equiv \widehat{\theta}_j(\boldsymbol{\xi}_q), && S_{kqi} \equiv \frac{\partial \widehat{\theta}_i}{\partial \xi_k}(\boldsymbol{\xi}_q), && J^{-1}_{qlk} \equiv J^{\mathrm{inv}}_{lk}(\boldsymbol{\xi}_q), \\ &J_q \equiv |J(\boldsymbol{x}(\boldsymbol{\xi}_q))|, && C_{ql} \equiv \boldsymbol{c}_l(\boldsymbol{x}(\boldsymbol{\xi}_q)), && w_q \equiv w(\boldsymbol{\xi}_q), \end{aligned} \tag{2.37}$$

in which case, we can succinctly write

$$B_{ij} = \sum_q w_q \, C_{ql} \, J^{-1}_{qlk} \, S_{kqi} \, J_q \, \Theta_{qj}, \tag{2.38}$$

and the tensor contraction notation handles all the complexity of the sum operations and immediately generalizes to any spatial dimension, whereas equation (2.36) is specific to two dimensional problems.

Similarly, we can write the inner product $e(\widehat{u}, \mathsf{v}) = \langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{\partial K}$ as the action of a discrete operator E on the vector $\widehat{\boldsymbol{c}u_h}$.

$$e(\widehat{u}, \mathsf{v}) = \langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{\partial K} = E \, \widehat{\boldsymbol{c}u_h} = E_{lij} \, [\widehat{\boldsymbol{c}u_h}]_{lj} \, ,$$

We could have written the operator as acting on the numerical solution directly, as with the expression $E' u_h$. However, the operator E' would in general need to operate on the entire numerical solution u_h, because $\widehat{\boldsymbol{c}u_h}$ is constructed using information from neighboring elements. In this case, the operator E' would need to specify the operation for constructing the numerical flux as well as integrating it over the element boundary and could become very complicated, since there are many choices for the numerical flux $\widehat{\boldsymbol{c}u_h}$.

Instead, we presume that the chosen form of the numerical flux $\widehat{\boldsymbol{c}u_h}$ has been computed over the element boundary ∂K.

$$\begin{aligned} \langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{\partial K} &= \int_{\partial K} \widehat{\boldsymbol{c}u_h}_j \cdot \boldsymbol{n} \, \mathsf{v} \, d\partial K \\ &= \sum_{e \in \partial K} \int_e (\widehat{\boldsymbol{c}u}_x n_x + \widehat{\boldsymbol{c}u}_y n_y) \, \theta_i \, de \\ &= \sum_{e \in \partial K} \int_{\widehat{e}} (\widehat{\boldsymbol{c}u}_x \, n_x + \widehat{\boldsymbol{c}u}_y \, n_y) \, \widehat{\theta}_i |J_e(\boldsymbol{x}(\xi))| \, d\widehat{e} \end{aligned} \tag{2.39}$$

where, since we have written the integral specifically in the two-dimensional case, the boundary has spatial coordinates $\boldsymbol{x}(\xi)$ parametrized by the single coordinate ξ over the master edge $\widehat{e}$. Expanding $\widehat{\boldsymbol{c}u_h} = \widehat{f}_{lj}\theta_j$, where the index $j = 1, \ldots, n_e$ denotes the basis functions enumerated over a face e on the element boundary ∂K and the index l once again denotes the coordinate direction x_l on the physical element boundary. This integral is discretized with quadrature as

$$\langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{e \in \partial K} = \sum_q w_q \left(n_x \widehat{\boldsymbol{c}u_h} \Big|_{\boldsymbol{x}(\xi_q)} + n_y \widehat{\boldsymbol{c}u_h} \Big|_{\boldsymbol{x}(\xi_q)} \right) \widehat{\theta}_i(\xi_q) |J(\boldsymbol{x}(\xi_q))|$$

There are some important implementation details here. The coefficients u_j are typically stored at each nodal point j on the element faces, but the numerical flux must be evaluated at the quadrature points. Let the indices k, l denote the coordinate directions on the master element ξ_k and on the physical element x_l, respectively. Let the index q index the quadrature point and weight, and the indices i, j index the basis function defined on the element boundary ∂K. If the tensors Θ^e, N, and vectors J^e and w are defined as

$$\Theta^e_{qi} \equiv \widehat{\theta}_i(\boldsymbol{\xi}_q), \quad N_{ql} \equiv n_l(\boldsymbol{x}(\boldsymbol{\xi}_q)), \quad J^e_q \equiv |J(\boldsymbol{x}(\boldsymbol{\xi}_q))|, \quad w_q \equiv w(\boldsymbol{\xi}_q),$$

Then in order to compute the flux, we need both the velocity field $\boldsymbol{c}$ evaluated at the quadrature points, as well as the interpolation of the nodal solution to the quadrature points.

$$\widehat{\boldsymbol{c}u_h}\Big|_{\boldsymbol{x}(\xi_q)} = \widehat{f}\left(\boldsymbol{c}(\boldsymbol{x}_q)\Theta_{qi}u_i\right) \equiv \widehat{F}_{lq}$$

where, as in equation (2.30), $\widehat{f}(\cdot)$ denotes the flux evaluation function. The complete edge integral is

$$\langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{e \in \partial K} = \sum_q w_q\, J^e_q\, N_{ql}\, \Theta^e_{qi}\, \widehat{F}_{lq} \tag{2.40}$$

Finally, to include the entire integral over each edge $e \in \partial K$ and making use of the lifting operator L^e defined on each edge to re-index the edge integral contributions to the interior of the element K, the discrete operator E_{lij} is

$$E_{qli} = \sum_{e \in \partial K} \mathsf{L}^e \left(w_q\, J^e_q\, N_{ql}\, \Theta^e_{qi}\right) \tag{2.41}$$

such that

$$[\langle \widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}, \mathsf{v} \rangle_{\partial K}]_i = E_{qli}\widehat{F}_{lq}. \tag{2.42}$$

2.2.7 Temporal discretization

For the temporal discretization, we begin with a simple forward Euler scheme

$$\left(\frac{\partial u_h}{\partial t}, \mathsf{v}\right)_K \approx \left(\frac{u_h^{k+1} - u_h^k}{\Delta t}, \mathsf{v}\right)_K \tag{2.43}$$

yielding the explicit scheme

$$\left(\frac{u_h^{k+1}}{\Delta t}, \mathsf{v}\right)_K = \left(\boldsymbol{c}u_h^k, \mathsf{v}\right)_K - \left\langle \widehat{\boldsymbol{c}u_h}^k \cdot \boldsymbol{n}, \mathsf{v} \right\rangle_{\partial K} + \left(\frac{u_h^k}{\Delta t}, \mathsf{v}\right)_K \tag{2.44}$$

where, again making use of the expansion $u_h = u_j\theta_j$,

$$\begin{aligned} \left(\frac{u_h}{\Delta t}, \mathsf{v}\right)_K &= \int_K \frac{1}{\Delta t} u_j \theta_j \theta_i \, dK \\ &= \underbrace{\int_{\widehat{K}} \widehat{\theta}_i |J(\boldsymbol{x}(\boldsymbol{\xi}))| \widehat{\theta}_j \, d\widehat{K}}_{M_{ij}} \; \frac{u_j}{\Delta t} \end{aligned} \tag{2.45}$$

$$M_{ij} \approx \sum_q w_q \widehat{\theta}_i(\boldsymbol{\xi}_q) |J(\boldsymbol{x}(\boldsymbol{\xi}_q))| \widehat{\theta}_j(\boldsymbol{\xi}_q) \tag{2.46}$$

Let the tensors and indices be defined as in equation (2.37), then

$$M_{ij} = \sum_q w_q J_q \Theta_{qi} \Theta_{qj}. \tag{2.47}$$

$$M \frac{u_h^{k+1}}{\Delta t} = B u_h^k - E\left[\widehat{\boldsymbol{c}u_h}^k\right] + M \frac{u_h^k}{\Delta t} \tag{2.48}$$

which can be painlessly implemented in the form

$$u_h^{k+1} = u_h^k + \Delta t M^{-1}\left(Bu_h^k - E\left[\widehat{\boldsymbol{c}u_h}^k\right]\right). \tag{2.49}$$

Remark. The M^{-1} operator is specific to each element K due to the Jacobian determinant $|J(\boldsymbol{x})|$. However, if the physical elements are affine transformations of the master element $\widehat{K}$, the matrix M^{-1} need not be computed nor stored, since J and hence $|J|$ is constant over the physical element K, allowing

$$M_{ij} = |J|\int_{\widehat{K}} \widehat{\theta}_j\widehat{\theta}_j \, d\widehat{K}, = |J|\widehat{M}_{ij} \tag{2.50}$$

and we can write the matrix M as a function of the template mass matrix $\widehat{M}$, computed once over the master element and stored along with its inverse $\widehat{M}^{\text{inv}}$. This form admits the inverse

$$M^{-1} = \frac{1}{|J|}\widehat{M}^{\text{inv}}, \tag{2.51}$$

which can be inexpensively applied, since $\widehat{M}^{\text{inv}}$ is the same for all elements. The only storage requirement is $|J|$. This idea is the basis of all the quadrature-free integration schemes we will encounter in §2.4.

Extensions to higher order temporal schemes completes the description of the RKDG method for linear advection. Instead of the specific temporal discretization in equation (2.43), we can instead write the semi-discretization

$$\frac{du_h}{dt} = \mathcal{L}_h(u_h)$$

where $\mathcal{L}_h(u_h) = M^{-1}\left(Bu_h - E\left[\widehat{\boldsymbol{c}u_h}\right]\right)$. Just as in equation (2.26), this system of ODEs can be discretized using an explicit RK scheme of choice (see §7.5).

2.2.8 Numerical experiments

2.2.8.1 Discretization of divergence term

Equation (2.30) demonstrates that we represent the integral of the divergence over each element as

$$\left(\nabla\cdot\boldsymbol{C},\, \mathsf{v}\right)_K = -\left(\boldsymbol{C},\, \nabla\mathsf{v}\right)_K + \left\langle \widehat{\boldsymbol{C}}\cdot\boldsymbol{n},\, \mathsf{v}\right\rangle_{\partial K}. \tag{2.52}$$

Therefore, as a numerical experiment, we can monitor how quickly the right-hand side of equation (2.52) converges to zero, for a choice of divergence-free velocity field $\boldsymbol{C}$ for which the scalar field $\nabla\cdot\boldsymbol{C}\notin W_h^p$, that is, a scalar field which can not be exactly represented in the polynomial approximation space.

We can use the approach in Appendix section 7.1.0.3 to construct the non-trivial, divergence-free vector field

$$\boldsymbol{C} = \left[-e^{-y}\sin(\pi x),\; -\pi\left(x\sin(\pi x) + e^{-y}\cos(\pi x)\right)\right] \tag{2.53}$$

shown in Figure 2-2. This velocity field is a good test case because of its nontrivial inflow boundary Γ_{in} on the top and bottom domain boundaries. The left and right sides of the domain have $\boldsymbol{C}\cdot\boldsymbol{n} = 0$ and are therefore part of the outflow boundary Γ_{out}. Numerically, the top and bottom inflow boundaries are (approximately)

$$\begin{aligned}\Gamma_{\text{in}}^{\text{top}} &= [-0.873, 0.873],\\ \Gamma_{\text{in}}^{\text{bottom}} &= [-1,1]\setminus(-0.565, 0.565),\end{aligned}$$

respectively. Measuring convergence numerically corresponds to forming the discrete operators as

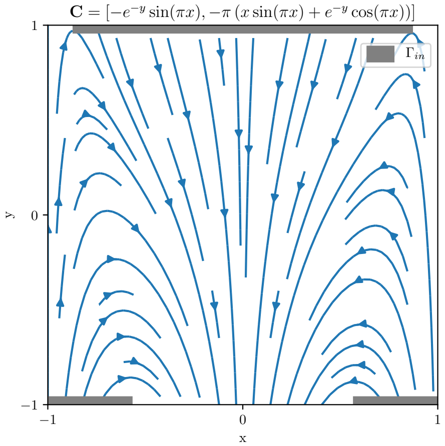


Figure 2-2: The divergence-free vector field in equation (2.53).

described in section 2.2.6 and evaluating

$$(\nabla \cdot \boldsymbol{C},\, \mathsf{v})_K \approx -BC^K + E\widehat{C}^K$$

on each element $K \in \mathcal{T}_h$ and computing the error over the domain at different resolutions. Doing so recovers optimal order $p+1$ convergence rates (not shown), and verifies the implementation of the discrete operators.

2.2.8.2 Rotating Gaussian pulse

We consider a velocity field $\boldsymbol{c} = [2\pi y,\, -2\pi x]$, which corresponds to a rigid body rotation about the origin in the clockwise direction. We solve the linear advection equation (2.29) on the domain $\Omega = [-1,\, 1]^2$. The initial condition is a Gaussian pulse given by

$$u_0 = \exp\left(-120\left(x^2 + (y-0.5)^2\right)\right),$$

shown in Figure 2-3. The exact solution is such that at time $t = 1$, the initial distribution will have returned to its initial position, and any error in the numerical solution can be measured against the initial condition u_0. As boundary conditions, we specify that $u_h = 0$ on Γ_{in}.

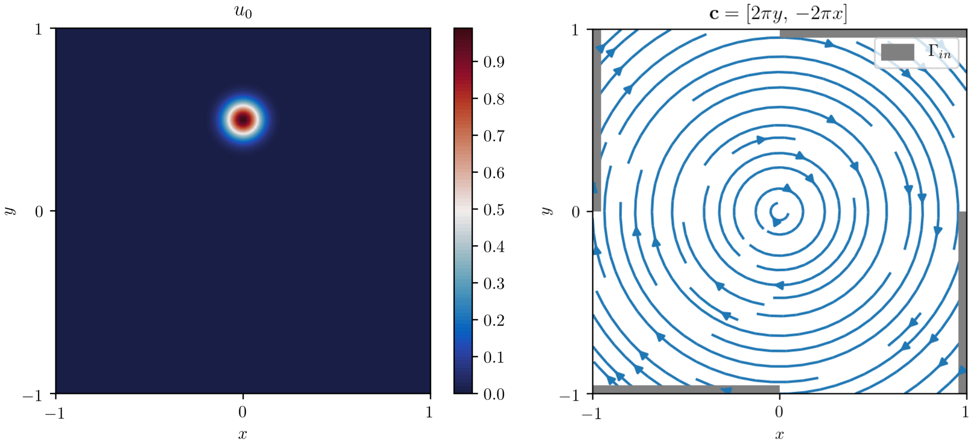


Figure 2-3: RKDG advection of Gaussian pulse on a coarse grid with $p = 5$ quad elements.

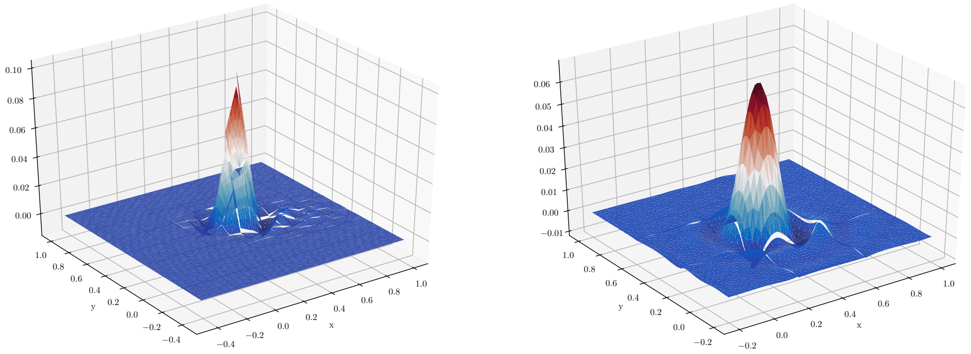


Figure 2-4: RKDG advection of Gaussian pulse on a coarse grid with $p = 5$ quad elements.

We consider quadrilateral meshes obtained by splitting a regular $n \times n$ Cartesian grid into a total of n^2 quadrilateral elements, giving uniform element sizes of $h = 2/N$. We present the convergence behavior in the L^2-norm in Figure 2-5. An explicit fourth-order Runge-Kutta scheme is used for time integration, with the time-step chosen small enough such that temporal error is small compared to the spatial error ($\Delta t = 1e - 3$ for $p = 4$). We see that the approximate solution u_h converges optimally at rate $p + 1$ or better for $p = 1, 2, 3, 4$.

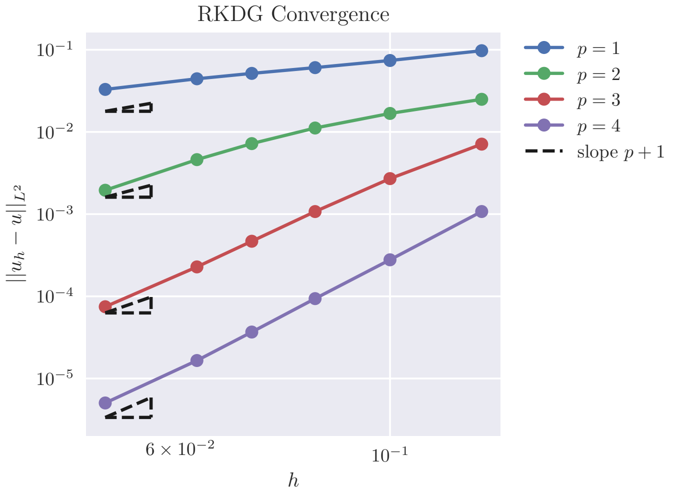


Figure 2-5: Convergence behavior of the RKDG scheme for the rotating pulse test case.

2.2.9 Implicit treatment of first-order DG systems

The transport equation is

$$\begin{aligned} \sigma u + \nabla \cdot (\boldsymbol{c} u) &= f, && \text{in } \Omega \\ u &= u_D && \text{on } \Gamma_{\text{in}} \end{aligned} \tag{2.54}$$

where the inflow boundary Γ_{in} is defined as $\Gamma_{\text{in}} = \{\boldsymbol{x} \in \partial\Omega \colon \boldsymbol{c} \cdot \boldsymbol{n} < 0\}$, and the outflow boundary is defined as $\Gamma_{\text{out}} \equiv \partial\Omega \setminus \Gamma_{\text{in}}$. The parameter $\sigma > 0$ is a reaction rate [196]. Multiplying the PDE by a test function v and integrating by parts, we have the weak formulation

$$\int_\Omega (\sigma u v - \boldsymbol{c} u \cdot \nabla v) + \int_{\partial\Omega} \boldsymbol{c} u \cdot \boldsymbol{n} v = \int_\Omega f v \tag{2.55}$$

The DG discretization seeks $u_h \in W_h^p$ such that

$$(\sigma u_h,\, \mathsf{w})_K - (\boldsymbol{c} u_h,\, \nabla \mathsf{w})_K + \langle \widehat{\boldsymbol{c} u_h} \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial K} = (f,\, \mathsf{w})_K \tag{2.56}$$

for all $K \in \mathcal{T}_h$. To complete the definition of the method, we need to define the numerical flux $\widehat{\boldsymbol{c} u_h}$ and sum the local problems over all the elements of the computational domain. It turns out that the numerical flux in equation (2.32) with $C > 0$ is sufficient for stability of the method. However, unlike the RKDG schemes for linear advection, this DG scheme is implicit. Substituting the numerical flux, the element-local equations become

$$(\sigma u_h,\, \mathsf{w})_K - (\boldsymbol{c} u_h,\, \nabla \mathsf{w})_K + \langle \widehat{\boldsymbol{c} u_h} \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial K \cap \varepsilon^\circ} + \langle \boldsymbol{c} u_h \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial K \cap \Gamma_{\text{out}}} = (f,\, \mathsf{w})_K - \langle \boldsymbol{c} u_D \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial K \cap \Gamma_{\text{in}}}\,,$$

which has a discrete solution that, summed over all elements $K \in \mathcal{T}_h$, yields the following weak problem: find $u_h \in W_h^p$ such that

$$(\sigma u_h,\, \mathsf{w})_{\mathcal{T}_h} - (\boldsymbol{c} u_h,\, \nabla \mathsf{w})_{\mathcal{T}_h} + \langle \widehat{\boldsymbol{c} u_h} \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial \mathcal{T}_h \setminus \Gamma_{\text{in}}} = (f,\, \mathsf{w})_{\mathcal{T}_h} - \langle \boldsymbol{c} u_D \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\Gamma_{\text{in}}}$$

for all $\mathsf{w} \in W_h^p$. This weak formulation can be expressed with the discrete bilinear form $a_h(u_h,\, \mathsf{w}) = \ell(\mathsf{w})$. In principle, this completes the description of the DG method.

It is illustrative to consider the formation of the discrete operators on the element faces, since this is the first implicit DG method we've considered. Formation of the numerical flux terms involves assembling the jumps and averages in terms of the knowns of the linear system.

It is convenient to perform the integrals over each face rather than over each element boundary. In the case of the interior edges,

$$\langle \widehat{\boldsymbol{c} u_h} \cdot \boldsymbol{n},\, \mathsf{w} \rangle_{\partial \mathcal{T}_h \setminus \Gamma} = \sum_{e \in \varepsilon^\circ} \langle \widehat{\boldsymbol{c} u_h},\, [\![\mathsf{w} \boldsymbol{n}]\!] \rangle_e\,.$$

Using the definition of the numerical flux, as well as expanding the mean and jump operators for a single interior edge $e \in \varepsilon^\circ$,

$$\begin{aligned}
\langle \widehat{\boldsymbol{c} u_h},\, [\![\mathsf{w} \boldsymbol{n}]\!] \rangle_e &= \langle \boldsymbol{c} \{\!\{ u_h \}\!\} + \tau [\![u_h \boldsymbol{n}]\!],\, [\![\mathsf{w} \boldsymbol{n}]\!] \rangle_e \\
&= \left\langle \frac{\boldsymbol{c}}{2} \left(u_h^+ + u_h^- \right) + \tau \left(u_h^+ \boldsymbol{n}^+ + u_h^- \boldsymbol{n}^- \right),\, \mathsf{w}^+ \boldsymbol{n}^+ + \mathsf{w}^- \boldsymbol{n}^- \right\rangle_e \\
&= \left\langle \left(\frac{\boldsymbol{c}}{2} u_h^+ + \tau u_h^+ \boldsymbol{n}^+ \right) + \left(\frac{\boldsymbol{c}}{2} u_h^- + \tau u_h^- \boldsymbol{n}^- \right),\, \mathsf{w}^+ \boldsymbol{n}^+ + \mathsf{w}^- \boldsymbol{n}^- \right\rangle_e\,,
\end{aligned}$$

resulting in four contributions:

$$\begin{aligned}
E^{++} &= \left\langle \frac{\boldsymbol{c}}{2} u_h^+ + \tau u_h^+ \boldsymbol{n}^+,\, \mathsf{w}^+ \boldsymbol{n}^+ \right\rangle_e\,, & E^{+-} &= \left\langle \frac{\boldsymbol{c}}{2} u_h^+ + \tau u_h^+ \boldsymbol{n}^+,\, \mathsf{w}^- \boldsymbol{n}^- \right\rangle_e\,, \\
E^{-+} &= \left\langle \frac{\boldsymbol{c}}{2} u_h^- + \tau u_h^- \boldsymbol{n}^-,\, \mathsf{w}^+ \boldsymbol{n}^+ \right\rangle_e\,, & E^{--} &= \left\langle \frac{\boldsymbol{c}}{2} u_h^- + \tau u_h^- \boldsymbol{n}^-,\, \mathsf{w}^- \boldsymbol{n}^- \right\rangle_e\,,
\end{aligned}$$

where, for example, E_{ij}^{+-} is formed with the usual numerical integration over the master edge using the finite element expansion of the solution $u_h^+ = u_j^+ \phi_j^+(\boldsymbol{x})$

$$E_{ij}^{+-} \approx \sum_q w_q \left(\frac{1}{2} \boldsymbol{c}_l(\boldsymbol{x}(\boldsymbol{\xi}_q)) + \tau \boldsymbol{n}_l^+(\boldsymbol{x}(\boldsymbol{\xi}_q)) \right) \widehat{\phi}_j^+(\boldsymbol{\xi}_q) \left(\widehat{\phi}_i^-(\boldsymbol{\xi}_q) \boldsymbol{n}_l^-(\boldsymbol{x}(\boldsymbol{\xi}_q)) \right) |J_e| \Big|_{\boldsymbol{x}(\boldsymbol{\xi}_q)}$$

where there is a sum over the physical coordinate l by Einstein convention to handle the inner product. The remaining operators are computed similarly. Now we have laid the groundwork for implicit (H)DG-FEM schemes arising in the solution of second-order problems.

2.3 HDG methods for second-order, elliptic problems

The benefits of DG-FEM come at a price; namely, the increase in total degrees of freedom as a result of the discontinuous approximation spaces and element-wise decoupling of the solution. In particular, for implicit elliptic and parabolic problems, the increase in the size of the linear system can outweigh the benefits of a DG-FEM approach. Hybridizable discontinuous Galerkin (HDG) methods were first introduced for second-order elliptic problems in Cockburn [42] in order to mitigate the cost of the discontinuous representation of the solution. In HDG schemes, the globally coupled unknowns have support on the element interfaces only, resulting in a DG approximation and convergence properties at a cost similar to classical FEM [43, 124, 191]. The differences in degree of freedom distribution for CG-FEM, DG-FEM, and HDG are schematically illustrated in Figure 2-6. In light of these benefits, recent research has extended HDG methods to a wide variety of implicit problems [193, 194, 195, 197, 207, 243, 244, 246]. However, for large-scale computations, hybridization alone is often insufficient to overcome memory and time-to-solution limitations, resulting in ongoing research [73, 79, 134, 225, 224] and partially motivating the work completed in this thesis.

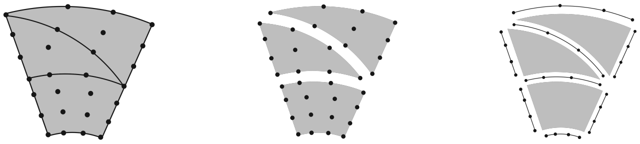


Figure 2-6: Globally coupled degrees of freedom for CG-FEM (left), DG-FEM (center), and HDG-FEM (right). Reproduced from previous chapter.

The purpose of this section is to provide an overview of the HDG methodology applied to a simple model problem in the interest of exposition. The techniques and ideas employed in the model problem will generalize well to the more complicated problems considered in the remainder of the text.

The HDG methods we will consider in this context are mixed methods. Mixed methods refer to finite element methods where more than one approximation space is used to approximate the quantities of interest in the problem [34]. For example, in the case of the numerical solution of the Stokes equations with classical CG-FEM techniques, different approximation spaces can be used to treat the velocity and pressure [196]. In other mixed methods, an auxiliary variable is introduced to the differential form of the governing equations and is solved as an additional problem unknown. This approach is common in DG-like schemes with higher than first-order spatial derivatives (e.g., diffusion problems) in order to preserve problem consistency and stability [110, 264]. Such is the case in the following model problem [243, 246].

2.3.1 The model problem

Consider the boundary value problem (BVP) consisting of the Poisson equation, subject to the Dirichlet and Neumann boundary conditions:

$$\begin{aligned} -\nabla \cdot (\kappa \nabla u) &= f && \text{in } \Omega, \\ u &= g_D && \text{on } \Gamma_D, \\ \kappa \nabla u \cdot \boldsymbol{n} &= g_N && \text{on } \Gamma_N. \end{aligned} \tag{2.57}$$

We consider the mixed formulation of the boundary value problem by introducing an auxiliary variable $\boldsymbol{q} = \kappa \nabla u$:

$$\begin{aligned} \boldsymbol{q} - \kappa \nabla u &= 0 && \text{in } \Omega, \\ -\nabla \cdot \boldsymbol{q} &= f && \text{in } \Omega, \\ u &= g_D && \text{on } \Gamma_D, \\ \boldsymbol{q} \cdot \boldsymbol{n} &= g_N && \text{on } \Gamma_N. \end{aligned} \tag{2.58}$$

In addition to the approximation spaces introduced in §2.2.2, we also consider the space $M_h^p(g_D) \equiv \{\mu \in M_h^p \colon \mu = \mathsf{P}g_D \text{ on } \Gamma_D\}$ where P denotes the L^2-projection into the space $\{\mu\big|_{\partial\Omega} \, \forall \mu \in M_h^p\}$. Figure 2-1b visually illustrates the spaces defined in equation (3.4.1) using an exploded view of the curvilinear mixed mesh depicted earlier in Figure 2-1a. Boundary edges are colored differently to emphasize that the degrees of freedom on boundary edges may or may not be a problem unknown, if the boundary edge lies on Γ_N or Γ_D, respectively.

2.3.2 Weak formulation

To formulate the HDG methodology, we solve a local DG problem on each element $K \in \mathcal{T}_h$, and solve a global problem on an "element edge space", which enforces transmission conditions and domain boundary conditions. The global solution is then supplied as boundary data to each element locally as boundary conditions (problem data), from which the interior solution can be reconstructed by solving each of the element-local systems.

We consider the model problem (2.58). On each $K \in \mathcal{T}_h$, for the given data $f\big|_K$ and $\widehat{u}_h\big|_{\partial K}$, we seek $(\boldsymbol{q}_h, u_h) \in \boldsymbol{V}^p(K) \times W^p(K)$ as the solution of the element-local problem

$$\begin{aligned} \left(\kappa^{-1}\boldsymbol{q}_h,\, \boldsymbol{v}\right)_K + (u_h,\, \nabla\cdot\boldsymbol{v})_K - \langle \widehat{u}_h,\, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\partial K} &= 0 & \forall \boldsymbol{v} \in \boldsymbol{V}^p(K), \\ (\boldsymbol{q}_h,\, \nabla w)_K - \langle \widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n},\, w\rangle_{\partial K} &= (f,\, w)_K & \forall w \in W^p(K), \end{aligned} \tag{2.59}$$

which represents a weak variational form of the original problem in equation (2.58) multiplied by test functions $\boldsymbol{v}$ and w and integrated by parts over the element K. The choice of weak variational form is not unique, and different options are discussed in [110], but we proceed using equation (2.59) without loss of generality. We define

$$\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n} \equiv \boldsymbol{q}_h\cdot\boldsymbol{n} + \tau\left(\widehat{u}_h - u_h\right) \text{ on } \partial K \tag{2.60}$$

where τ is assumed to be known. Since $\widehat{\boldsymbol{q}}_h$ is a function of $\widehat{u}_h$, if $\widehat{u}_h$ is known on ∂K, the element-local problem is solvable. Further, if $\widehat{u}_h$ is known on the element interfaces ε, then $\widehat{u}_h$ is also known on every element boundary ∂K and *every* element-local problem can be solved independently. To determine $\widehat{u}_h$ globally, we take $\widehat{u}_h \in M_h^p$ and impose the transmission condition that the normal component of the numerical flux $\widehat{\boldsymbol{q}}_h$ be single-valued on the element edge space:

$$\langle [\![\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n}]\!],\, \mu\rangle_{\partial\mathcal{T}_h} = \langle \mathsf{P}g_N,\, \mu\rangle_{\Gamma_N}, \qquad \forall \mu \in M_h^p\left(g_D\right). \tag{2.61}$$

For PDEs with diffusion, $\boldsymbol{q} \propto \nabla u$ can be interpreted as a diffusive flux. We expect that, physically, the normal component of the diffusive flux can not jump across interfaces [66]; this is a physical interpretation of the transmission condition in equation (2.61). By summing over all mesh elements and imposing the continuity of the normal component of the numerical flux [191], the complete weak problem can be written as follows: find $(\boldsymbol{q}_h, u_h, \widehat{u}_h) \in (\boldsymbol{V}_h^p,\, W_h^p,\, M_h^p)$ such that

$$\begin{aligned} \left(\kappa^{-1}\boldsymbol{q}_h,\, \boldsymbol{v}\right)_{\mathcal{T}_h} + (u_h,\, \nabla\cdot\boldsymbol{v})_{\mathcal{T}_h} - \langle \widehat{u}_h,\, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} &= 0 & \forall \boldsymbol{v} \in \boldsymbol{V}_h^p \\ (\boldsymbol{q}_h,\, \nabla w)_{\mathcal{T}_h} - \langle \widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n},\, w\rangle_{\partial\mathcal{T}_h} &= (f,\, w)_{\mathcal{T}_h} & \forall w \in W_h^p \\ \langle \widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n},\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} + \langle \widehat{u}_h - \mathsf{P}g_D,\, \mu\rangle_{\Gamma_D} &= \langle \mathsf{P}g_N,\, \mu\rangle_{\Gamma_N} & \forall \mu \in M_h^p \end{aligned} \tag{2.62}$$

In principle, this completes the formulation of the HDG methodology for linear diffusion problems. However, we have not made precise the procedure for expressing $\widehat{\boldsymbol{q}}_h$ and $\widehat{u}_h$ in terms of the element-local data. Moreover, by more closely examining the formulation, we can gain insight into the properties and features of the method.

2.3.2.1 The global problem for $\widehat{u}_h$: weak form

We now discuss the discrete equivalent of the continuous approach above whereby we eliminate the variables $\boldsymbol{q}_h$ and u_h in equation (2.62) and form a single equation for $\widehat{u}_h$, a procedure we will refer to as "static condensation". For the purposes of handling the Dirichlet boundary conditions, we introduce a new unknown $\lambda_h \in M_h^p(0)$ that vanishes on Γ_D such that

$$\widehat{u}_h = \begin{cases} \lambda_h, & \text{on } \partial\mathcal{T}_h \setminus \Gamma_D, \\ \mathsf{P} g_D, & \text{on } \Gamma_D. \end{cases} \tag{2.63}$$

Substituting the flux definitions (2.60) and (2.63) into (2.62), we can rewrite the problem as: find $(\boldsymbol{q}_h,\, u_h,\, \lambda_h) \in (\boldsymbol{V}_h^p,\, W_h^p,\, M_h^p(0))$ such that

$$\begin{aligned} \left(\kappa^{-1}\boldsymbol{q}_h,\, \boldsymbol{v}\right)_{\mathcal{T}_h} + (u_h,\, \nabla\cdot\boldsymbol{v})_{\mathcal{T}_h} - \langle \lambda_h,\, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} &= \langle \mathsf{P} g_D,\, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\Gamma_D} \\ -\,(\nabla\cdot\boldsymbol{q}_h,\, w)_{\mathcal{T}_h} + \langle \tau u_h,\, w\rangle_{\partial\mathcal{T}_h} - \langle \tau\lambda_h,\, w\rangle_{\partial\mathcal{T}_h} &= (f,\, w)_{\mathcal{T}_h} + \langle \tau\mathsf{P} g_D,\, w\rangle_{\Gamma_D} \\ \langle \boldsymbol{q}_h\cdot\boldsymbol{n},\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} - \langle \tau u_h,\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} + \langle \tau\lambda_h,\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} &= \langle \mathsf{P} g_N,\, \mu\rangle_{\Gamma_N} \end{aligned} \tag{2.64}$$

for all $(\boldsymbol{v},\, w, \mu) \in (\boldsymbol{V}_h^p,\, W_h^p,\, M_h^p(0))$. Note that $\mu \in M_h^p(0)$ removes all global basis functions on Γ_D from the unknowns of the system. The Dirichlet boundary conditions are enforced directly by the definition of $\widehat{u}_h$ in (2.63).

Following the approach in [191], we define the following linear operators and bilinear forms:

$$\begin{aligned} a(\boldsymbol{q},\, \boldsymbol{\mathsf{v}}) &= \left(\kappa^{-1}\boldsymbol{q},\, \boldsymbol{\mathsf{v}}\right)_{\mathcal{T}_h} & & \\ b(u, \boldsymbol{\mathsf{v}}) &= (u,\, \nabla\cdot\boldsymbol{\mathsf{v}})_{\mathcal{T}_h} & & \\ c(\lambda, \boldsymbol{v}) &= \langle \lambda,\, \boldsymbol{\mathsf{v}}\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} & r(\boldsymbol{\mathsf{v}}) &= \langle \mathsf{P} g_D,\, \boldsymbol{\mathsf{v}}\cdot\boldsymbol{n}\rangle_{\mathcal{T}_h} \\ d(u, \mathsf{w}) &= \langle \tau u,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} & f(\mathsf{w}) &= (f,\, \mathsf{w})_{\mathcal{T}_h} + \langle \tau\mathsf{P} g_D,\, \mathsf{w}\rangle_{\Gamma_D} \\ e(\lambda, \mathsf{w}) &= \langle \tau\lambda,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} & \ell(\mu) &= \langle \mathsf{P} g_N,\, \mu\rangle_{\Gamma_N} \\ g(u, \mu) &= \langle \tau u,\, \mu\rangle_{\partial\mathcal{T}_h} & & \\ h(\lambda, \mu) &= \langle \tau\lambda,\, \mu\rangle_{\partial\mathcal{T}_h} & & \end{aligned} \tag{2.65}$$

and we can write equation (2.64) compactly as

$$\begin{aligned} a(\boldsymbol{q}_h,\, \boldsymbol{\mathsf{v}}) + b(u_h,\, \boldsymbol{\mathsf{v}}) - c(\lambda_h,\, \boldsymbol{\mathsf{v}}) &= r(\boldsymbol{\mathsf{v}}), \\ b(\mathsf{w},\, \boldsymbol{q}_h) - d(u_h,\, \mathsf{w}) + e(\lambda_h,\, \mathsf{w}) &= -\,f(\mathsf{w}), \\ c(\boldsymbol{q}_h, \mu) - g(u_h,\, \mu) + h(\lambda_h,\, \mu) &= \ell(\mu), \end{aligned} \tag{2.66}$$

which represents the weak form of the problem. We could perform the discretization of (2.64) directly, which would lead to the linear system

$$\begin{bmatrix} A & B & -C \\ B^T & -D & E \\ C^T & -G & H \end{bmatrix} \begin{bmatrix} Q \\ U \\ \Lambda \end{bmatrix} = \begin{bmatrix} R \\ -F \\ L \end{bmatrix}. \tag{2.67}$$

If we were to assemble the discretized linear system (2.67) with the unknowns arranged in the order $[(q_h,\, u_h)_K\ \forall K \in \mathcal{T}_h,\, \lambda_h]$, the linear system would have features as shown in Figure 2-8, where Q, U, Λ are the vectors of degrees of freedom for $\boldsymbol{q}_h$, u_h, λ_h, respectively. The important feature is that elemental unknowns $\boldsymbol{q}_h$ and u_h correspond to a block-diagonal structure due to the discontinuous nature of the approximation spaces. To see this, take the inner product $\left(\kappa^{-1}\boldsymbol{q}_h,\, \boldsymbol{v}\right)_{\mathcal{T}_h}$, without loss of generality. Choose a nodal basis of $\boldsymbol{v}$ such that $\boldsymbol{q}_h = \sum_i q_i\theta_i$ for global basis functions θ_i. Then the quantity $\left(\kappa^{-1}q_i\theta_i,\, \theta_j\right)_{\mathcal{T}_h}$ can *only* be nonzero if θ_i and θ_j are on the same element K, due to the definition of V_h^p. In particular,

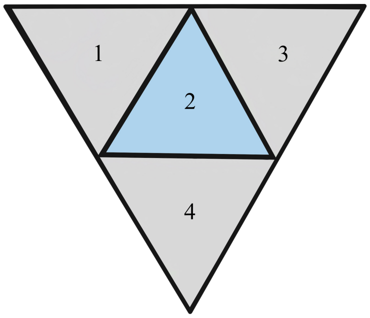


Figure 2-7: Mesh corresponding to the discretization in Figure 2-8

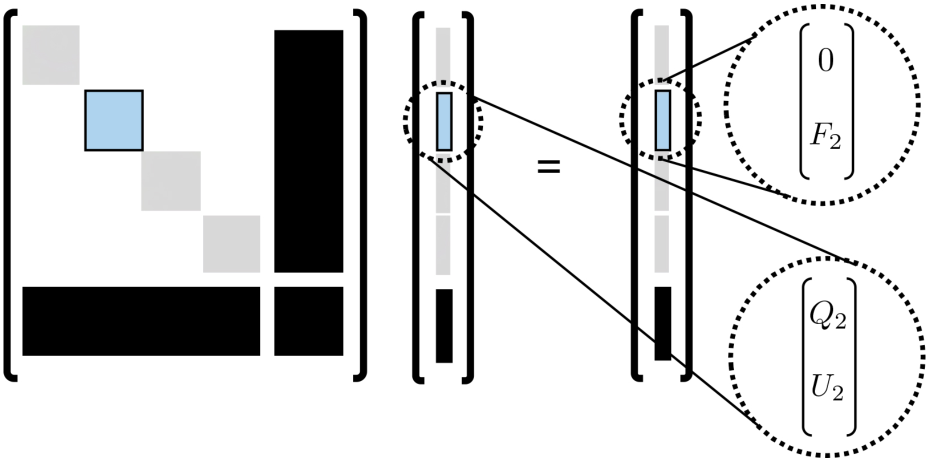


Figure 2-8: Direct discretization of (2.67) (bottom) with zero Dirichlet boundary conditions ($g_D = 0$) for a four-element mesh (top). The unknown vector of the system is ordered $(\boldsymbol{q}_h, u_h)_K \ \forall K$ (green, blue), followed by λ_h (black).

$$\begin{bmatrix} A & B \\ B^T & -D \end{bmatrix} \tag{2.68}$$

is block diagonal, which implies that Q and U can be eliminated on each element, resulting in a linear system in terms of Λ alone. To perform this elimination, we write the system (2.67) as:

$$\begin{bmatrix} A & B \\ B^T & -D \end{bmatrix} \begin{bmatrix} Q \\ U \end{bmatrix} - \begin{bmatrix} C \\ -E \end{bmatrix} \Lambda = \begin{bmatrix} R \\ -F \end{bmatrix},$$
$$\begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} Q \\ U \end{bmatrix} + H\Lambda = L.$$

Since, by eliminating the block system for Q and U,

$$\begin{bmatrix} Q \\ U \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & -D \end{bmatrix}^{-1} \left(\begin{bmatrix} R \\ -F \end{bmatrix} + \begin{bmatrix} C \\ -E \end{bmatrix} \Lambda \right), \tag{2.69}$$

substitution yields a single equation in terms of Λ:

$$\begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} A & B \\ B^T & -D \end{bmatrix}^{-1} \left(\begin{bmatrix} R \\ -F \end{bmatrix} + \begin{bmatrix} C \\ -E \end{bmatrix} \Lambda \right) + H\Lambda = L,$$

forming the linear system $\mathbb{K}\Lambda = \mathbb{F}$, where

$$\begin{aligned}\mathbb{K} &= H + \begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} A & B \\ B^T & -D \end{bmatrix}^{-1} \begin{bmatrix} C \\ -E \end{bmatrix}, \\ \mathbb{F} &= L - \begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} A & B \\ B^T & -D \end{bmatrix}^{-1} \begin{bmatrix} R \\ -F \end{bmatrix}.\end{aligned} \tag{2.70}$$

As is typical in finite element implementations, these contributions are computed for each element (independently of the other elements) and are assembled to form the global linear system $\mathbb{K}\Lambda = \mathbb{F}$.

Once the linear system has been solved, Q and U can be computed directly from Λ by a modification of equation (2.69). We refer to this step as the "reconstruction" of the unknowns $\boldsymbol{q}_h$ and u_h. The traced solution variable $\widehat{u}_h$ contains both the computed unknown fluxes λ_h and the boundary conditions $\mathsf{P}g_D$. Alternatively, if g_D is used in $\hat{u}$ rather than the L^2 projection $\mathsf{P}g_D$, the reconstructed solution will obey the boundary conditions exactly, but this can pollute the convergence order if g_D lies outside the polynomial space. Hence, we can write

$$\begin{bmatrix} Q \\ U \end{bmatrix} = \begin{bmatrix} A & B \\ B^T & -D \end{bmatrix}^{-1} \left(\begin{bmatrix} 0 \\ -F \end{bmatrix} + \begin{bmatrix} C \\ -E \end{bmatrix} \widehat{u}_h \right), \tag{2.71}$$

where the operator F is defined as $F = (f, \mathsf{w})_{\mathcal{T}_h}$ in equation (2.71) rather than as defined in equation (2.65)—a slight abuse of notation. [1] We take a moment to remark upon the form of the elemental contributions $\mathbb{K}$ and right-hand side contributions $\mathbb{F}$, since their construction and application will have important implications for iterative solution techniques. The matrix inverse present in both $\mathbb{K}$ and $\mathbb{F}$ represents the parametrization of the element-local degrees of freedom to the global edge space, and is fundamental to the HDG methodology—such an inverse is not required for continuous Galerkin schemes. Assuming a static mesh in time, this inverse ought to be computed once and stored (ideally, in a memory-friendly way), and applied when forming either $\mathbb{K}$ or $\mathbb{F}$.

2.3.3 Boundary condition treatment

We take a moment to address treatment of Dirichlet boundary conditions in the formation of the linear system. Due to definition (2.63), all boundary data appears only on the right hand side of equation (2.64). This choice is important to the structure of the discretized linear system for λ_h. Deferring the details of how to discretize the continuous operators for the moment, if definition

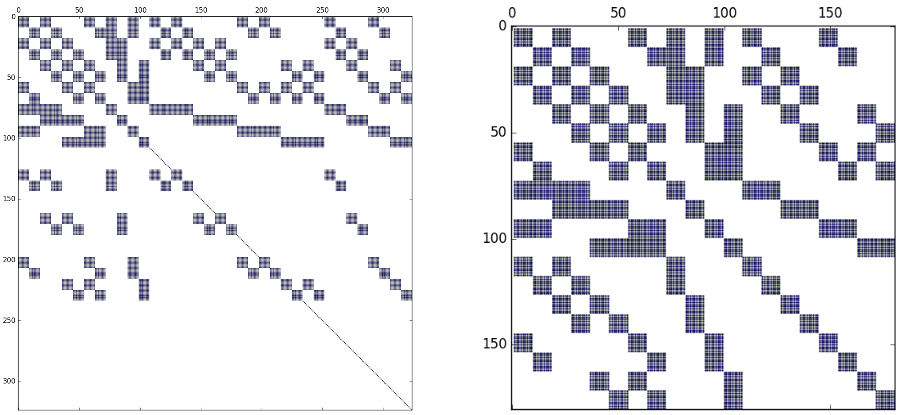


Figure 2-9: Sparsity patterns of $\mathbb{K}$ with implicit (left) and explicit (right) treatment of Dirichlet boundary conditions.

(2.63) is not made, and λ_h is instead sought in the space M_h^p, the Dirichlet boundary conditions will

[1] This is because the Dirichlet information g_D is encapsulated in $\widehat{u}_h$.

be treated implicitly, resulting in a nonsymmetric linear system $\mathbb{K}\Lambda = \mathbb{F}$, as shown in Figure 2-9. Therefore, this definition is not merely a frivolous detail, but has computational implications and results in a more smaller, symmetric linear system.

2.3.4 The HDG Algorithm

The steps to compute the solution $\boldsymbol{q}_h$ and u_h using the HDG methodology are summarized in Figure 2-10. We will consider various modifications to this algorithm in the interest of computational efficiency in this work. It bears repeating that computation of the element-local contributions $\mathbb{K}^K$, $\mathbb{F}^K$ as well as the elemental reconstruction of $\boldsymbol{q}_h^K$ and u_h^K can be done independently of every other element. Both steps are embarrassingly parallelizable. The solution of the global linear system $\mathbb{K}\Lambda = \mathbb{F}$ represents the portion of the algorithm where elemental information is coupled through the edge space unknowns in the approximation space M_h^p.

```
HDG Algorithm($\mathcal{E}$):
for $K \in \mathcal{T}_h$
    compute elemental contributions $\mathbb{K}^K$, $\mathbb{F}^K$ (2.70)
    scatter($\mathbb{K}$, $\mathbb{F}$, $\mathcal{E}$, $\mathbb{K}^K$, $\mathbb{F}^K$)
$\lambda_h \leftarrow$ solve $\mathbb{K}\Lambda = \mathbb{F}$
$\hat{u}_h \leftarrow \lambda_h \cup g_D$
for $K \in \mathcal{T}_h$
    reconstruct $\boldsymbol{q}_h^K, u_h^K \leftarrow \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix}^{-1} \left( \begin{bmatrix} 0 \\ F \end{bmatrix} + \begin{bmatrix} C \\ E \end{bmatrix} \widehat{u}_h^K \right)$
    $\boldsymbol{q}_h, u_h \leftarrow \boldsymbol{q}_h^K, u_h^K$
return $\boldsymbol{q}_h$, $u_h$
```

```
scatter($\mathbb{K}$, $\mathbb{F}$, $\mathcal{E}$, $\mathbb{K}^K$, $\mathbb{F}^K$):
for $K \in \mathcal{T}_h$
    $\alpha \leftarrow \mathcal{E}(K) \setminus \Gamma_D$
    $J, I \leftarrow$ meshgrid$(\alpha, \alpha)$
    $\mathbb{K}[I[:], J[:]] \leftarrow \mathbb{K}[I[:], J[:]] + \mathbb{K}^K$
    $\mathbb{F}[I[:]] \leftarrow \mathbb{F}[I[:]] + \mathbb{F}^K$
return
```

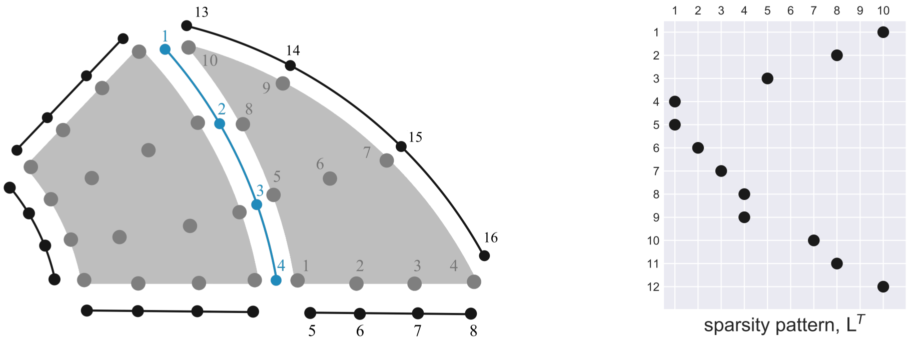
Figure 2-10: HDG algorithm

Indeed, both algorithmic features are desirable and demonstrate the advantages of the HDG approach. While a standard discontinuous Galerkin scheme involves the spatially duplicated interior degrees of freedom in the spaces W_h^p and $\boldsymbol{V}_h^p$, the only globally-coupled unknowns in the HDG approach are the edge space degrees of freedom, drastically reducing the linear system size.

2.3.5 Assembly of the global linear system

Assembly of the global linear system from the elemental contributions $\mathbb{K}^K$ and $\mathbb{F}^K$ requires a mapping between the volume nodes and global unknowns in $M_h^p(g_D)$.

We define the "lifting" operator L, a permutation matrix that maps the nodes on the element boundary ∂K to the ordering of volume nodes. Similarly, L^T maps the volume nodes to nodes on the element boundary ∂K. The lifting operator L operator is computed once on the master element $\widehat{K}$. For details, see the implementation in Hesthaven and Warburton [110]. This operator allows for transfer of information from the elemental volume nodes to boundary nodes and vice-versa. We also define the local-to-global index map $\mathcal{E}(K)$ that maps the nodes on ∂K to the edge space nodes in M_h^p for each element K. The procedure for assembly of $\mathbb{K}$ is written in Figure 2-10, where the `meshgrid` function returns the matrices resulting from the first and second elements of the Cartesian product $\alpha \times \alpha$ and $I[:]$ refers to the vector formed by iterating over an array in a row-wise manner. To make the index map procedure explicit, we refer to the example in Figure 2-11, which depicts the node numbering around the triangular element K_{tri} for a two element mixed mesh. The boundary nodes on ∂K_{tri} are labeled from 1 to 12 counterclockwise, starting from edge node 1. The edge space nodes have a global numbering, but only the nodes on the blue interior edge are unknowns $\lambda_h \in M_h^p(g_D)$.



$$\mathcal{E}(K_{tri}) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 16 & 15 & 14 & 13 \end{bmatrix}$$

Figure 2-11: Illustration of the node numbering (left), lifting operator L^T (right), and index map $\mathcal{E}$ (bottom) for the triangular element K_{tri} in the two element mesh. Nodes on the interior edge ε° are blue, nodes on Dirichlet boundary edges $\varepsilon^\partial \in \Gamma_D$ are black, and volume nodes are grey.

2.3.6 Extension to time-dependent problems

Consider the time-dependent extension [196, 191] of the model problem in equation (2.58)

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \cdot (\kappa \nabla u) &= f && \text{in } \Omega \times (0, T], \\ u &= g_D && \text{on } \Gamma_D \times (0, T], \\ \kappa \nabla u \cdot \boldsymbol{n} &= g_N && \text{on } \Gamma_N \times (0, T], \\ u &= u_0 && \text{in } \Omega \text{ at } t = 0, \end{aligned} \tag{2.72}$$

written as a system of first-order equations:

$$\begin{aligned} \boldsymbol{q} - \kappa \nabla u &= 0 && \text{in } \Omega \times (0, T], \\ \frac{\partial u}{\partial t} - \nabla \cdot \boldsymbol{q} &= f && \text{in } \Omega \times (0, T], \\ u &= g_D && \text{on } \Gamma_D \times (0, T], \\ \boldsymbol{q} \cdot \boldsymbol{n} &= g_N && \text{on } \Gamma_N \times (0, T], \\ u &= u_0 && \text{in } \Omega \text{ at } t = 0. \end{aligned} \tag{2.73}$$

We modify the element-local problem with a "method of lines" approach; on each $K \in \mathcal{T}_h$, for the given data $f\big|_K$ and $\widehat{u}_h\big|_{\partial K}$, we seek $(\boldsymbol{q}_h, u_h) \in \boldsymbol{V}^p(K) \times W^p(K)$ as the solution of the problem

$$\begin{aligned} \left(\kappa^{-1} \boldsymbol{q}_h,\, \boldsymbol{v}\right)_K + (u_h,\, \nabla \cdot \boldsymbol{v})_K - \langle \widehat{u}_h,\, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\partial K} &= 0 && \forall \boldsymbol{v} \in \boldsymbol{V}^p(K), \\ \left(\frac{\partial u_h}{\partial t},\, w \right)_K + (\boldsymbol{q}_h,\, \nabla w)_K - \langle \widehat{\boldsymbol{q}}_h \cdot \boldsymbol{n},\, w \rangle_{\partial K} &= (f,\, w)_K && \forall w \in W^p(K). \end{aligned} \tag{2.74}$$

The above equation can be discretized using a suitable time-marching scheme (for details, see §7.5); we use a to denote a generic time-stepping coefficient, from an IMEX [242, 246], DIRK, or backward Euler scheme ($a = 1$, shown here). Making the same substitutions as above and using the operators and bilinear forms introduced in (2.65), we have the following weak problem at time-level k, without loss of generality: find $(\boldsymbol{q}_h,\, u_h,\, \lambda_h) \in (\boldsymbol{V}_h^p,\, W_h^p,\, M_h^p(0))$ such that

$$\begin{aligned} a(\boldsymbol{q}_h^k,\, \mathsf{v}) + b(u_h^k,\, \mathsf{v}) - c(\lambda_h^k,\, \mathsf{v}) \quad &= r(\mathsf{v}), \\ \frac{1}{a\Delta t^k} m(u_h^k,\, \mathsf{w}) - b(\mathsf{w},\, \boldsymbol{q}_h^k) + d(u_h^k,\, \mathsf{w}) - e(\lambda_h^k,\, \mathsf{w}) \quad &= f(\mathsf{w}) + \frac{1}{a\Delta t^k} m(u_h^{k-1},\, \mathsf{w}), \\ c(\boldsymbol{q}_h^k, \mu) - g(u_h^k,\, \mu) + h(\lambda_h^k,\, \mu) \quad &= \ell(\mu), \end{aligned} \tag{2.75}$$

for all $(\mathsf{v},\, \mathsf{w},\, \mu) \in (\boldsymbol{V}_h^p,\, W_h^p,\, M_h^p(0))$, where $m(u,\, \mathsf{w}) = (u,\, \mathsf{w})_{\mathcal{T}_h}$, and where boundary conditions and forcing are evaluated at time t^k or t^{k+1}.

2.3.7 Parallelization

HDG methods offer several well-known advantages over standard DG-FEM methods. In addition to favorable convergence properties, hybridization of the system by introducing the space M_h^p can lead to a substantial reduction in globally-coupled degrees of freedom as compared to classical DG methods. However, the complexities associated with the static condensation procedure introduce a set of non-typical computational performance considerations as compared to other high-order methods. In this section, we provide a brief discussion of practical computational considerations specific to HDG schemes in the context of parallelization and matrix-free solvers. These topics are often overlooked or unaddressed in the literature, with some discussion given in [124].

A commonly cited performance advantage of HDG is the embarrassingly parallel nature of the assembly as well as the local reconstruction of the local numerical solution $(\boldsymbol{q}_h, u_h)$ from $\hat{u}_h$ on the element edges. Means that people brush this step off computationally, as a parallel implementation should make a perfect speedup trivial. However, in practice, there are additional computational considerations.

Though it may seem counter-intuitive, multi-threaded programs can sometimes run slower than their single-threaded or serial counterparts. This is attributable to several factors. Overhead: Threads require resources for creation, termination, and synchronization. The overhead of managing multiple threads can slow down the program, especially if the computation performed by each thread is not significant compared to this overhead. Contention: When multiple threads try to access a shared resource simultaneously, contention can occur. This can lead to a situation called a lock, where one thread has to wait for another to release a resource. In extreme cases, this can lead to a slowdown known as lock contention, where the threads spend more time waiting to access the resource than doing useful work. In this context, contention occurs in the assembly of the element-local data into the global linear system. While the computation of the element local contributions can be done independently, a lock must be placed on the degrees of freedom being scattered to in the global linear system to ensure that no race condition occurs. Furthermore, the order in which elemental data is scattered to the global linear system matters and must be ensured to be consistent. Otherwise, as floating point arithmetic is not commutative, and the different threads are not guaranteed to receive work in the same order, each execution of the same code could produce numerically different results. A large amount of computer science work has been done in devising algorithms that ensure that these two conditions are met, while maintaining efficient implementations [10].

False sharing: Even if different threads are working on different data, if that data is close enough in memory (i.e., in the same cache line), then the hardware can treat it as if it were shared. When one thread updates its data, the hardware will think the entire cache line has been modified and will have to update it across all the cores, leading to a significant slowdown. This is known as false sharing. Load imbalance: If the work is not evenly distributed across threads, some threads may finish their work early and sit idle, while others are still working. This is known as load imbalance. It can result in the execution time of the program being dominated by the slowest thread. Non-parallelizable tasks: Some tasks simply can't be parallelized effectively due to their nature, they are inherently serial (or "sequential"). This principle is encapsulated by Amdahl's Law, which states that the maximum improvement to a system's performance is limited by the portion of the system that can't be parallelized. Memory limits: Every thread created uses some memory for its stack. If the number of threads is too high, it can cause significant memory usage, potentially leading to swapping if the system runs out of physical memory, which can drastically decrease performance.

Developing an efficient multi-threaded program requires careful design to minimize these issues. This can involve techniques such as fine-tuning the number of threads, avoiding contention and false sharing, balancing the load properly, and choosing appropriate data structures and synchronization primitives. But even with all these considerations, there's no guarantee that a multi-threaded solution will always outperform a well-optimized single-threaded one. It highly depends on the nature of the problem and the specific hardware the software is running on.

In our case, the synchronization work necessary to ensure no race condition while copying elemental contributions into the global linear system is highly sensitive to problem dimension and polynomial order. This is due to a trade-off between the time required to compute the elemental work, the overhead of starting and joining threads, and the synchronization requirements to ensure the correct scattering of local data into the global linear system.

In the following, we discretize the model HDG problem at different polynomial orders such that the global number of primal degrees of freedom are held constant. We evaluate the assembly times using both serial and parallelized assembly. It turns out that for one-dimensional problems using any polynomial order less than or equal to $p = 10$, it is far more efficient to perform assembly in serial as opposed to in parallel using multi-threading. This is because the problem dimension leads to element-local systems small enough that the synchronization work between the threads to prevent a race condition copying into the global linear system is much larger than the actual work on the cell. From the 2D and 3D results shown in Figure 2-12 and Figure 2-13, there is a clear dependence on polynomial order and problem size dictating whether a serial or parallel implementation should be used.

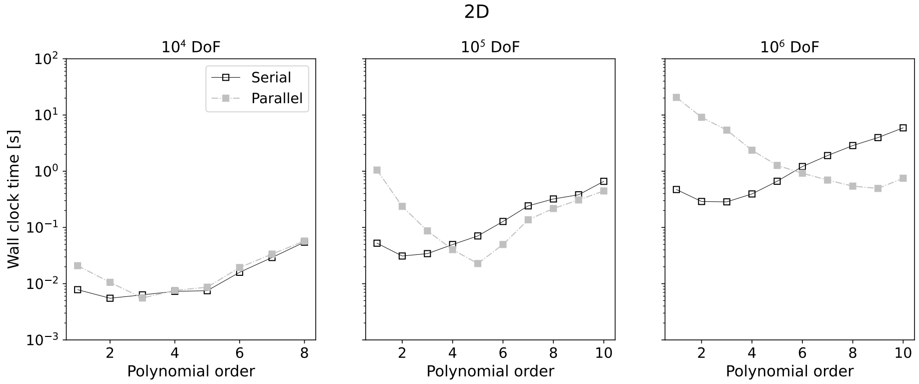


Figure 2-12: Wall clock times for 2D HDG assembly at constant problem size, serial vs. parallel

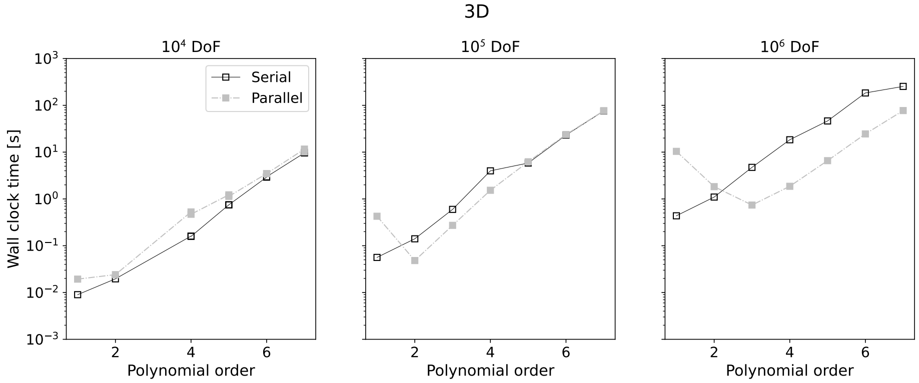


Figure 2-13: Wall clock times for 3D HDG assembly at constant problem sizes, serial vs. parallel

Therefore, when it comes to software implementing HDG methods, parallelization must be accompanied by thorough profiling to ensure that the parallelization is actually beneficial, as that is not always the case. For specific examples related to the applications considered in this thesis: the large 3D model-nested cases considered in §3 hugely benefit from a multi-threaded parallel implementation, but the Markov chain Monte Carlo samplers, which solve very many small HDG problems during sampling, can be slowed down by more than an order of magnitude if multi-threading is enabled.

2.4 Matrix-free schemes for HDG-FEM

In this section, we present quadrature-free and matrix-free implementations of the hybridizable discontinuous Galerkin (HDG) method for second-order elliptic problems and their time-dependent generalizations in two and three spatial dimensions.

Finite element codes typically employ Gaussian quadrature to evaluate discrete integral operators, as they provide an accurate and efficient means of computing spatial integrals. However, discontinuous Galerkin (DG) schemes require the evaluation of both volume and face integrals and no single quadrature rule can be used to compute both types of integrals to the same accuracy [15, 16]. Furthermore, nodal discontinuous Galerkin schemes typically formulate the problem unknowns as the coefficients of a nodal polynomial basis on each element [110]. As a consequence, the solution is expressed naturally through data at the nodal points on each element, and evaluation of spatial integrals involves the interpolation of data to quadrature points.

The original quadrature-free schemes for discontinuous Galerkin finite elements were introduced in [16], in the context of explicit schemes for hyperbolic systems. Quadrature-free discontinuous Galerkin schemes are particularly efficient in an explicit setting because they eliminate the need to interpolate nodal data to the quadrature points, significantly reducing the storage requirements and operations necessary to form the boundary and interior integral operators for evolving the numerical solution on each element. Subsequent developments have remained largely focused on the explicit hyperbolic setting, including the shallow water equations [74, 189] and the level-set equations [177].

DG methods are often paired with explicit time discretization because the discontinuous nature of the polynomial approximation spaces results in a mass matrix is block-diagonal by element and hence cheaply inverted. However, a major drawback of DG schemes is the increase in degrees of freedom associated with discontinuous approximation spaces. For implicit temporal discretizations, HDG methods parametrize the relevant problem into one for the trace of the numerical solution on the mesh skeleton, yielding a substantial reduction in globally-coupled degrees of freedom. Furthermore, HDG solutions allow for element-wise post-processing which provides more accurate numerical solutions than classical DG methods [43].

In [132], the authors make inroads into quadrature-based, matrix-free evaluation of implicit HDG operators—however, the schemes therein rely on fast sum factorization techniques specialized to tensor-product quadrilateral and hexahedral elements, whereas the current work focuses on techniques for more general finite elements. In particular, choosing a nodal basis collocated with a quadrature scheme may require a higher-order nodal basis for tetrahedral and wedge finite elements. Secondly, the schemes in [132] involve an HDG formulation in terms of both the primal variable and its trace, whereas the present work focuses on the classical HDG formulation in terms of the trace alone.

Matrix-free HDG schemes pose a numerical challenge because forming the operators for the globally-coupled linear system requires a matrix inversion, rendering straightforward implementations inefficient in both memory and time. The primary contribution of this work is the construction of a numerically efficient matrix-free evaluation for implicit HDG schemes, leveraging the templated form of quadrature-free integral operators. However, the advantages of quadrature-free integration come at the cost of accuracy. An important question considered in this work is the trade-off between the computational advantages and the loss of accuracy due to the abandonment of quadrature. To the best of our knowledge, there is no existing work comparing the accuracy and cost trade-offs for quadrature-free and quadrature-based discretizations for implicit systems such as those considered in [191, 192, 193, 194, 197].

This section represents the continuation of work done in [244], which compared quadrature-based and quadrature-free discretizations of source terms in an HDG setting.

We focus primarily on schemes for the diffusion equation, because the HDG solvers for such problems are readily generalized to a host of more complicated problems, including the linear and nonlinear convection-diffusion equations [191, 197], the equations of biogeochemical ocean models [244], the acoustic wave equation [193], and finite element projection methods for the incompressible Navier–Stokes equations [246]. The matrix-free schemes contained herein are immediately applicable to all of the aforementioned problems.

In section 2.4.1, we extend the quadrature-free approaches described in [15, 16, 110] to implicit HDG methods. In section 2.4.2, we use the quadrature-free methodology to derive a novel set of matrix-free schemes for implicit and explicit HDG discretizations. In section 2.4.4, we perform an investigation of the trade-offs between accuracy and efficiency for quadrature-based and quadrature-free HDG schemes.

2.4.1 Quadrature-free integration

We use $\boldsymbol{\xi}$ to denote the coordinates on the reference finite element $\widehat{K}$. The physical coordinates $\boldsymbol{x}$ on every element $K \in \mathcal{T}_h$ can be expressed as a vector-valued transformation $X : \widehat{K} \to K$ from the reference element. Isoparametric finite element formulations approximate the transformation X as a linear combination of the nodal basis functions on the reference element, $\boldsymbol{x} = \boldsymbol{X}(\boldsymbol{\xi}) \approx \sum_i \boldsymbol{x}_i \psi_i(\boldsymbol{\xi})$. We denote the Jacobian of the isoparametric transformation and its inverse as

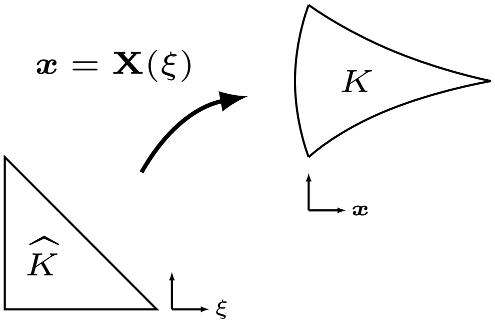


Figure 2-14: Transformation from the reference coordinate system to the physical coordinate system.

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial \boldsymbol{x}_j}{\partial \boldsymbol{\xi}_i}, \qquad \left[J^{-1}(\boldsymbol{x})\right]_{ij} = \frac{\partial \boldsymbol{\xi}_j}{\partial \boldsymbol{x}_i},$$

along with its determinant $|J| = \det(J(\boldsymbol{x}))$. DG schemes involve both volume and surface integrals, each of which requires isoparametric transformation data from the reference element and the reference edge [110]. We use ζ to denote the coordinates on the reference edge $\widehat{\varepsilon}$; the corresponding transformation and its Jacobian are $\boldsymbol{x} = \boldsymbol{X}_\varepsilon(\boldsymbol{\zeta})$ and J_e, respectively.

We define the following template operators on the reference element and edges, which can be computed using only the nodal basis functions $\psi(\boldsymbol{\xi})$ and $\psi^\varepsilon(\boldsymbol{\zeta})$ defined on the reference element $\widehat{K}$ and edge $\widehat{\varepsilon}$, respectively:

$$\mathsf{M}_{ij} = (\psi_i, \psi_j)_{\widehat{K}}, \quad \mathsf{S}_{k,ij} = \left(\psi_i, \frac{\partial \psi_j}{\partial \xi_k}\right)_{\widehat{K}}, \quad [\mathsf{M}_\varepsilon]_{ij} = \left(\psi_i^\varepsilon, \psi_j^\varepsilon\right)_{\widehat{\varepsilon}}, \quad \mathsf{M}_{\partial \widehat{K}} = \boldsymbol{I} \otimes \mathsf{M}_\varepsilon \ \ \forall \varepsilon \in \partial \widehat{K}, \tag{2.76}$$

where $\mathsf{M}_{\partial \widehat{K}}$ is the mass matrix on the element boundary, defined as the block-diagonal Kronecker product of the identity matrix with the reference edge mass matrix for each element face $F \in \partial K$. Additionally, the lifting matrix L is the permutation matrix which re-indexes degrees of freedom on the element boundary ∂K to the element interior K.

2.4.1.1 Formation of the discrete operators

The matrices in (2.64) are readily written as

$$\begin{aligned} A_{ji} &= \left(\kappa^{-1}\boldsymbol{\theta}_i,\, \boldsymbol{\theta}_j\right)_{\mathcal{T}_h}, \quad & B_{ji} &= \left(\theta_i,\, \nabla\cdot\boldsymbol{\theta}_j\right)_{\mathcal{T}_h}, \quad & C_{ji} &= \left\langle \theta_{\varepsilon,i},\, \boldsymbol{\theta}_j\cdot\boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h} \\ D_{ji} &= \left\langle \tau\theta_i,\, \theta_j\right\rangle_{\mathcal{T}_h}, & E_{ji} &= \left\langle \tau\theta_{\varepsilon,i},\, \theta_j\right\rangle_{\partial\mathcal{T}_h}, & G_{ji} &= \left\langle \tau\theta_i,\, \theta_{\varepsilon,j}\right\rangle_{\partial\mathcal{T}_h}, \end{aligned} \tag{2.77}$$

where $\boldsymbol{\theta}_i(\boldsymbol{x})$ denote the physical space basis functions spanning the space $\boldsymbol{V}_h^p$, $\theta_i(\boldsymbol{x})$ are the basis functions spanning the space W_h^p, and $\theta_\varepsilon(\boldsymbol{x})$ are the basis functions which span the space $M_h^p(0)$. The operator H is formed similarly. The right-hand side vectors are analogously

$$R_j = \left\langle \mathsf{P}g_D,\, \boldsymbol{\theta}_j\cdot\boldsymbol{n}\right\rangle_{\Gamma_D}, \qquad F_j = \left(f,\, \boldsymbol{\theta}_j\right)_{\mathcal{T}_h} + \left\langle \tau\mathsf{P}g_D,\, \theta_j\right\rangle_{\Gamma_D}, \qquad L_j = \left\langle \mathsf{P}g_N,\, \theta_{\varepsilon,j}\right\rangle_{\Gamma_N}.$$

All integrals are mapped to the reference element through the isoparametric transformation (2.4.1). For example, the volume integral B_{ji} could be computed over the element K as

$$B_{ji} = \left(\theta_i,\, \frac{\partial\boldsymbol{\theta}_j}{\partial x_j}\right)_K = \int_{\widehat{K}} \psi_i \left[\sum_{\ell=1}^{d} \frac{\partial\psi_j}{\partial\xi_k} J_{k\ell}^{-1}(\boldsymbol{x})\right] |J(\boldsymbol{x})|\, d\widehat{K}, \tag{2.78}$$

where ψ_i refers to the nodal basis functions over the reference element as in §2.4.1 and we have made use of Einstein summation notation. Most finite element schemes approximate the integrals in equation (2.78) with quadrature; that is,

$$B_{ji} \approx \sum_{q=1}^{n_q} w_q \psi_i(\boldsymbol{\xi}_q) \left[\sum_{\ell=1}^{d} \frac{\partial\psi_j(\boldsymbol{\xi}_q)}{\partial\xi_k} J_{k\ell}^{-1}(\boldsymbol{X}(\boldsymbol{\xi}_q))\right] |J(\boldsymbol{X}(\boldsymbol{\xi}_q))|,$$

where the indices $q = 1, \ldots, n_q$ and the points $\boldsymbol{\xi}_q$ denote a set of elemental quadrature points with weights w_q.

Surface integrals on the mesh interfaces are computed element-wise on the boundary ∂K with a different quadrature set for which we denote the index q_ε. For example, the surface integral C_{ji} can be mapped to the reference edge and approximated with quadrature as

$$\begin{aligned} C_{ji} = \left\langle \theta_{\varepsilon,i},\, \boldsymbol{\theta}_j\cdot\boldsymbol{n}\right\rangle_\varepsilon &= \int_{\widehat{\varepsilon}} \psi_i^\varepsilon \left[\sum_{\ell=1}^{d} \psi_j n_\ell(\boldsymbol{x})\right] |J_e(\boldsymbol{x})|\, d\widehat{\varepsilon} \\ &\approx \sum_{q_\varepsilon=1}^{n_{q_\varepsilon}} w_{q_\varepsilon} \psi_i^\varepsilon(\boldsymbol{\zeta}_{q_\varepsilon}) \left[\sum_{\ell=1}^{d} \psi_j(\boldsymbol{\zeta}_{q_\varepsilon})\, n_\ell(\boldsymbol{X}(\boldsymbol{\zeta}_{q_\varepsilon}))\right] |J_e(\boldsymbol{X}(\boldsymbol{\zeta}_{q_\varepsilon}))| \end{aligned}$$

While numerical quadrature enjoys favorable accuracy when integrating polynomial functions, DG schemes require the evaluation of both volume and face integrals and no single quadrature rule can be used to compute both types of integrals to the same accuracy [16]. Furthermore, while the isoparametric transformation data J^{-1}, $|J|$, and $|J_e|$ and normal vectors $\boldsymbol{n}$ can be precomputed at the elemental quadrature points and stored, any nodal data to be integrated must be interpolated to the interior or edge quadrature points. In extensions of the elliptic problems considered in this work to convection-diffusion and Navier–Stokes problems, this interpolation can become expensive [246]. Finally, and most importantly in the context of matrix-free methods, assembly of the left-hand side operators (2.77) require tensor contraction operations cast as matrix-vector products over the reference element data and the transformation data.

Quadrature-free schemes, by contrast, compute spatial integrals using only nodal data. For a finite element expansion $a_i\theta_i$,

$$\begin{aligned}(a_i\theta_i,\, \theta_j)_K &= \int_{\widehat{K}} a_i\psi_i(\boldsymbol{\xi})\psi_j(\boldsymbol{\xi})|J(\boldsymbol{X}(\boldsymbol{\xi}))|\, d\widehat{K} \\ &\approx a_i|J(\boldsymbol{X}(\boldsymbol{\xi}_i))|\,(\psi_i,\, \psi_j)_{\widehat{K}} = a_i|J|_i\mathsf{M}_{ij}.\end{aligned} \tag{2.79}$$

It is evident that if the Jacobian is constant over the element K that this approximation is exact. If the transformation $\boldsymbol{X}(\boldsymbol{\xi})$ is non-affine, J will vary over space, introducing a transformation error to the approximation of the integral [110]. However, despite the error induced with the approximation, the form of integral operator expressed in equation (2.79) confers many computational advantages. This manner of integration obviates the need to interpolate to a differing set of quadrature points. The matrix-vector operations required with quadrature are reduced to much faster broadcasting operations. Perhaps more importantly, quadrature-free approaches allow for transformational data from the reference to physical element to be used directly with templated operators (2.76) defined on the reference element and edges. The importance this decoupling plays in the construction of matrix-free schemes specific to HDG methods is described in detail in §2.4.2.

We now present the quadrature-free forms of the operators in equation (2.64). It will be convenient to define the basis $\theta_i^{\partial K}(\boldsymbol{x}) \in \mathcal{P}^p\left(\{e\colon e \in \partial K\}\right)$, $i = 1,\ldots,n_r$, where the integer $n_r = \sum_{e\in\partial K} \dim\mathcal{P}^p(e)$ is the number of nodal basis functions on the element boundary ∂K; we denote the corresponding basis functions on the boundary of the master element as $\psi^{\partial K}(\boldsymbol{\zeta})$. Since the basis functions $\psi^{\partial K}$ have support only on their respective faces and not on the entire boundary of the reference element, it is apparent that $\left\langle \psi_i^{\partial K}, \psi_j^{\partial K}\right\rangle_{\partial\widehat{K}} = [\mathsf{M}_{\partial K}]_{ij}$ as specified in (2.76).

Quadrature-free HDG operators will make use of the nodal transformation data computed on each physical element K and its boundary ∂K:

$$\begin{aligned} &|J|_i = |J(\boldsymbol{x}_i)|, && && i = 1,\ldots,n_b, \\ &|J_e|_i = |J(\boldsymbol{x}_i)|, && \boldsymbol{n}_{\ell i} = \widehat{\boldsymbol{n}}_\ell(\boldsymbol{x_i}), && i = 1,\ldots,n_r, \end{aligned} \tag{2.80}$$

where the index i specifies basis function, ℓ indexes the physical space coordinate, and where the integer n_b denotes the number of nodal basis functions $\theta_i(\boldsymbol{x})$ on the interior of each physical element K.

To evaluate the quadrature-free analogue of (2.78), we make use of the fact that the vector-valued test function $\boldsymbol{\theta}_i$ is spanned by θ_i in each component. To that end, for each spatial dimension $k = 1,\ldots,d$, we form the discrete blocks

$$B^k = \sum_{\xi=1}^{d} \operatorname{diag}\left(|J| \circ J_{k\xi}^{-1}\right)[\mathsf{S}_\xi]^T\,, \tag{2.81}$$

where the $\circ$ operator denotes the point-wise or Hadamard product. By substituting in the finite element expansions and testing against each basis function on element K, we form the following elemental quadrature-free operators for the linear system in equation (2.64):

$$\begin{aligned} A &= \left(\kappa^{-1}(\cdot),\, \mathbf{v}\right)_K = I \otimes \operatorname{diag}\left(|J| \circ 1/\kappa_k\right)\mathsf{M}, \\ B &= \left(\,\cdot\,,\, \nabla\cdot\mathbf{v}_k\right)_K = \textstyle\sum_{\xi=1}^{d} \operatorname{diag}(|J| \circ J_{k\xi}^{-1})\mathsf{S}_\xi^T, \\ C &= \langle\,\cdot\,,\, \mathbf{v}\cdot\boldsymbol{n}_k\rangle_{\partial K} = \mathsf{L}\operatorname{diag}\left(|J_e| \circ \boldsymbol{n}_k\right)\mathsf{M}_{\partial\widehat{K}}, \\ D &= \langle\tau\left(\cdot\right),\, \mathsf{w}\rangle_{\partial K} = \mathsf{L}\operatorname{diag}\left(|J_e| \circ \tau\right)\mathsf{M}_{\partial\widehat{K}}\mathsf{L}^T, \\ E &= \langle\tau\left(\cdot\right),\, \mathsf{w}\rangle_{\partial K} = \mathsf{L}\operatorname{diag}\left(|J_e| \circ \tau\right)\mathsf{M}_{\partial\widehat{K}}, \\ G &= \langle\tau\left(\cdot\right),\, \mu\rangle_{\partial K} = \operatorname{diag}\left(|J_e| \circ \tau\right)\mathsf{M}_{\partial\widehat{K}}\mathsf{L}^T, \\ H &= \langle\tau\left(\cdot\right),\, \mu\rangle_{\partial K} = \operatorname{diag}\left(|J_e| \circ \tau\right)\mathsf{M}_{\partial\widehat{K}}, \end{aligned} \tag{2.82}$$

where I is the identity operator, and where the free index k specifies the block component of the operators B and C.

Similarly, we form the right-hand side vectors:

$$\begin{aligned} R_k &= \langle \mathsf{P} g_D,\, \mathbf{v}\cdot \boldsymbol{n}_k \rangle_{\partial K} = \mathsf{LM}_{\partial \widehat{K}}\left(|J_e| \circ \boldsymbol{n}_k \circ g_D\right), \\ F &= (f,\, \mathsf{w})_K + \langle \tau \mathsf{P} g_D,\, \mathsf{w}\rangle_{\partial K \cap \Gamma_D} = \mathsf{M}\left(|J| \circ f\right) + \mathsf{LM}_{\partial \widehat{K}}\left(|J_e| \circ \tau \circ g_D\right), \\ L &= \langle \mathsf{P} g_N,\, \mu \rangle_{\partial K \cap \Gamma_N} = \mathsf{M}_{\partial \widehat{K}}\left(|J_e| \circ g_N\right), \end{aligned} \tag{2.83}$$

where the integer $k = 1, \ldots, d$ represents coordinate direction. Let us stress once again that the transformation and parameter data has been separated from the templated data on the reference element, since the diag operator is applied through a broadcast rather than a matrix-vector product. Furthermore, the right-hand side data can be integrated directly from the nodal values without interpolation.

2.4.2 Methodology

In this section we derive a new set of matrix-free HDG schemes which make use of the quadrature-free implementation described in §2.4.1.

2.4.2.1 Memory costs

Iterative methods scale well as compared to direct methods, but it is important to identify and address the computational limiting factors both in memory and in time. In continuous Galerkin or standard implicit discontinuous Galerkin finite element schemes, these limitations are immediately apparent: storage and solution of the linear system are the memory and time-to-solution bottlenecks. However, for HDG schemes, the static condensation procedure in which $\boldsymbol{q}_h$ and u_h are eliminated to form a linear system for λ_h complicates scalability.

Equation (2.70) implies that that even if the linear system is assembled and stored in memory, computing the right–hand side $\mathbb{F}$ involves the forming the matrix inverse

$$\mathcal{A}_{loc} = \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix} \tag{2.84}$$

or its factorization for every element. The inverse A_{loc}^{-1} represents the HDG parametrization of the element-local PDE onto the edge space, and is distinct from the element-local contributions to a standard CG or DG finite element scheme. Somewhat surprisingly, the application of this inverse is an important computational consideration for large HDG problems.

Since the support of $\mu \in M_h^p(0)$ is on the inter-element edges $\varepsilon = \partial K^+ \cap \partial K^-$, the contributions to each basis function on the edge ε are from basis functions on the boundaries $\partial K^+ \cup \partial K^-$. Therefore the linear system $\mathbb{K}$ consists of a block structure with blocks of square matrices of size dim$\mathcal{P}^p$ [191]. Further, due to the contributions from $\partial K^+ \cup \partial K^-$, in two-dimensional problems, each block-row contains at most five non-zero blocks for triangular elements and seven non-zero blocks for quadrilateral elements. In three-dimensional problems, there are at most seven non-zero blocks for tetrahedral elements, nine blocks for wedge elements, and eleven blocks for hexahedral elements.

For a standard nodal basis on tensor-product elements in spatial dimension d and at polynomial order p, the number of degrees of freedom on each element interior n_b is $(p+1)^d$. The number of basis functions on each edge n_ε is $(p+1)^{d-1}$. The dimension of each $\mathcal{A}_{loc}$ matrix is $\left((d+1)(p+1)^d\right)^2$, and the dimension of each edge contribution matrix $\mathbb{K}_\varepsilon$ is $(p+1)^{d-1}$. Therefore for a mesh with N_K elements and N_{ed} interior edges, the number of non-zero entries required to store the $\mathcal{A}_{loc}^{-1}$ matrices scales as $N_K p^{2d} d^2$, whereas the number of non-zero entries in the global linear system $\mathbb{K}$ scales as $N_{ed} b_R\, p^{2d-2}$, with b_R being the number of non-zero blocks per block-row of $\mathbb{K}$.

For a three-dimensional domain of N hexahedral elements per side, the number of elements N_K as well as the number of edges N_{ed} scales as N^3. The number of non-zero entries for $\mathcal{A}_{loc}^{-1}$ scales as $N^3 p^6$, whereas the number of non-zero entries for $\mathbb{K}$ scales as $N^3 p^4$. At even modest polynomial orders, the memory required to store the matrix $\mathcal{A}_{loc}^{-1}$ on each element dominates the memory requirements.

In summary, the primary concern with regards to memory limitations is the storage or computation of the A_{loc}^{-1} arrays. If these are not stored, then storage of the sparse linear system $\mathbb{K}$ becomes the bottleneck.

Matrix-free methods are aimed at removing the memory bottleneck of $\mathbb{K}$, but applying $\mathbb{K}$ implicitly requires applying A_{loc}^{-1} — if the inverse is not available, applying $\mathbb{K}$ would require inversion of each A_{loc} matrix at every iteration of every iterative solve at every time step. If we were able to efficiently apply A_{loc}^{-1} without needing to store the array, we would have only the master-to-physical element mapping data J^{-1}, $|J|$, $|J_e|$ and the normal vectors as the comparatively modest memory requirements, and we could devise an efficient matrix-free implementation for the HDG schemes addressed earlier in this work.

We introduce the operator Ξ, defined as the Kronecker product $I \otimes \Xi_k$, where

$$\Xi_k = \mathsf{M}^{-1} \operatorname{diag}\left(\frac{\kappa_k}{|J|}\right).$$

Application of Ξ to the first equation in (2.70), assembled with the quadrature-free operators specified in (2.82), results in the modified linear system

$$\begin{bmatrix} I & \tilde{B} & -\tilde{C} \\ B^T & -D & E \\ C^T & -G & H \end{bmatrix} \begin{bmatrix} Q \\ U \\ \Lambda \end{bmatrix} = \begin{bmatrix} \tilde{R} \\ -F \\ L \end{bmatrix},$$

where $\tilde{B}$, $\tilde{C}$, and $\tilde{R}$ represent the matrix-product of Ξ with the original operators B, C, and R, respectively. Crucially, the quadrature-free representation of the operator A in (2.82) allows manipulation into the identity operator using only the templated mass matrix M on the reference element and local transformation data $|J|$. Performing the same element-by-element static condensation procedure as in §2.3, resulting in the equivalent global linear system $\tilde{\mathbb{K}}\Lambda = \tilde{\mathbb{F}}$, with

$$\begin{aligned} \tilde{\mathbb{K}} &= H + \begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} I & \tilde{B} \\ B^T & -D \end{bmatrix}^{-1} \begin{bmatrix} \tilde{C} \\ -E \end{bmatrix}, \\ \tilde{\mathbb{F}} &= L - \begin{bmatrix} C^T & -G \end{bmatrix} \begin{bmatrix} I & \tilde{B} \\ B^T & -D \end{bmatrix}^{-1} \begin{bmatrix} \tilde{R} \\ -F \end{bmatrix}. \end{aligned} \tag{2.85}$$

The elemental contributions to the linear system $\tilde{K}$ and right-hand side $\tilde{F}$ make use of the modified element-local system

$$\tilde{\mathcal{A}}_{loc} = \begin{bmatrix} I & \tilde{B} \\ B^T & -D \end{bmatrix}.$$

Since the HDG scheme is a mixed formulation, the identity operator I has size $\dim\mathcal{P}^p d$ where d is the spatial problem dimension. On the other hand, the block operator D has only size $\dim\mathcal{P}^p$. Therefore, the resultant matrix $\tilde{\mathcal{A}}_{loc}$ has a favorable sparsity pattern due to the identity operator, and admits the efficiently-applied inverse

$$\tilde{\mathcal{A}}_{loc}^{-1} = \begin{bmatrix} I + \tilde{B}S^{-1}B^T & -\tilde{B}S^{-1} \\ -S^{-1}B^T & S^{-1} \end{bmatrix},$$

where $S = -D - B^T\tilde{B}$ is the Schur complement with respect to the upper-left block of $\tilde{A}_{loc}$.

Since only the lower-left block must be stored, use of this formulation implies an immediate factor of 9 reduction of required memory for 2D problems and a factor of 16 reduction of required memory for 3D problems. Similarly, if the memory reduction is such that S^{-1} can be computed and stored, this approach removes the need for explicit matrix inversion, and the contribution operators can be applied successively to the solution vector in an efficient manner.

2.4.3 Extensions

The methodology of employing quadrature-free operators (§2.4.1) to derive efficient matrix-free schemes (§2.4.2) for the model problem (§2.3) can be readily extended to a rich set of additional problems.

2.4.3.1 Time-dependent problems

The weak problem (2.64) can be readily extended to the time-dependent problem with the Galerkin method of lines [191], where the weak form of the problem is modified as

$$\begin{aligned}
\left(\kappa^{-1}\boldsymbol{q}_h,\, \mathbf{v}\right)_{\mathcal{T}_h} + (u_h,\, \nabla\cdot\mathbf{v})_{\mathcal{T}_h} - \langle\lambda_h,\, \mathbf{v}\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} &= \langle \mathsf{P}g_D,\, \mathbf{v}\cdot\boldsymbol{n}\rangle_{\Gamma_D} \\
\left(\frac{\partial u_h}{\partial t},\, \mathsf{w}\right)_{\mathcal{T}_h} - (\nabla\cdot\boldsymbol{q}_h,\, \mathsf{w})_{\mathcal{T}_h} + \langle\tau u_h,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} - \langle\tau\lambda_h,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} &= (f,\, \mathsf{w})_{\mathcal{T}_h} + \langle\tau\mathsf{P}g_D,\, \mathsf{w}\rangle_{\Gamma_D} \\
\langle\boldsymbol{q}_h\cdot\boldsymbol{n},\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} - \langle\tau u_h,\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} + \langle\tau\lambda_h,\, \mu\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} &= \langle\mathsf{P}g_N,\, \mu\rangle_{\Gamma_N}
\end{aligned} \tag{2.86}$$

where the time derivative term can be discretized using a time marching scheme of choice. For example, using an s-stage diagonally-implicit Runge Kutta (DIRK) scheme leads to the relation

$$u_h^{n+1} = u_h^n + \Delta t \sum_{i=i}^{s} b_i u_h^{(i)} \tag{2.87}$$

where the second equation in (2.86) is temporally discretized as

$$\begin{aligned}
\left(\frac{u_h^{(i)}}{a_{ii}\Delta t},\, \mathsf{w}\right)_{\mathcal{T}_h} &- (\nabla\cdot\boldsymbol{q}_h,\, \mathsf{w})_{\mathcal{T}_h} + \langle\tau u_h,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} - \langle\tau\lambda_h,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h} \\
&= (f,\, \mathsf{w})_{\mathcal{T}_h} + \langle\tau\mathsf{P}g_D,\, \mathsf{w}\rangle_{\Gamma_D} + \frac{1}{a_{ii}\Delta t} m(x_h^{(i)},\, \mathsf{w})_{\mathcal{T}_h}
\end{aligned} \tag{2.88}$$

where $x_h^{(i)} = u^n + \sum_{j=1}^{i-1} a_{ij} u_h^{(j)}$, using the RK coefficients a_{ij}, b_i, and c_i, to advance from u^n to u^{n+1}. Comparing the above to the operators introduced thus far, the only required modifications are

$$\begin{aligned}
d(u, \mathsf{w}) &= \left(\frac{u_h^{(i)}}{a_{ii}\Delta t},\, \mathsf{w}\right)_{\mathcal{T}_h} + \langle\tau u_h,\, \mathsf{w}\rangle_{\partial\mathcal{T}_h}, \\
f(\mathsf{w}) &= (f,\, \mathsf{w})_{\mathcal{T}_h} + \langle\tau\mathsf{P}g_D,\, \mathsf{w}\rangle_{\Gamma_D} + \frac{1}{a_{ii}\Delta t} m(x_h^{(i)},\, \mathsf{w})_{\mathcal{T}_h},
\end{aligned} \tag{2.89}$$

which can be appropriately discretized into the corresponding discrete D and F operators and used directly in the methodology specified in §2.4.2 without further modification.

2.4.3.2 Convection-diffusion equation

Extension to the time-dependent convection-diffusion equation

$$\begin{aligned}
\boldsymbol{q} - k\nabla u &= 0, \\
\frac{\partial u}{\partial t} + \nabla\cdot(\boldsymbol{c}u) - \nabla\cdot\boldsymbol{q} &= f, \qquad \text{in } \Omega, \\
u &= g_D \quad \text{on } \Gamma_D, \\
\kappa\nabla u\cdot\boldsymbol{n} &= g_N \quad \text{on } \Gamma_N.
\end{aligned} \tag{2.90}$$

written here as a first order system, is likewise straightforward. Similar to the time-dependent schemes in §2.4.3.1, the addition of the convective term $\nabla\cdot(\boldsymbol{c}u)$ does not affect the equation for the gradient $\boldsymbol{q}$; therefore the manipulations in §2.4.2 carry through as well, with appropriate modifications to the $d(u,\,\mathsf{w})$, $e(\lambda,\,\boldsymbol{w})$, $f(\mathsf{w})$ and $h(\mu,\,\lambda)$ operators. Using the modified HDG flux definition $\widehat{\boldsymbol{c}u_h} + \widehat{\boldsymbol{q}}_h = \boldsymbol{c}\widehat{u}_h + \boldsymbol{q}_h + \tau(u_h - \widehat{u}_h)\boldsymbol{n}$ and the same time stepping scheme as the previous section, we have

$$\begin{aligned} d(u,\mathsf{w}) &= \left(\frac{u_h^{(i)}}{a_{ii}\Delta t},\,\mathsf{w}\right)_{\mathcal{T}_h} + \langle \tau u_h,\,\mathsf{w}\rangle_{\partial\mathcal{T}_h} - (\boldsymbol{c}u_h,\,\nabla\mathsf{w})_{\mathcal{T}_h}\,, \\ e(\lambda,\mathsf{w}) &= \langle(\tau - \boldsymbol{c}\cdot\boldsymbol{n})\lambda_h,\,\mathsf{w}\rangle_{\partial\mathcal{T}_h}\,, \\ f(\mathsf{w}) &= (f,\,\mathsf{w})_{\mathcal{T}_h} + \langle(\tau - \boldsymbol{c}\cdot\boldsymbol{n})\mathsf{P}g_D,\,\mathsf{w}\rangle_{\Gamma_D} + \frac{1}{a_{ii}\Delta t}m(x_h^{(i)},\,\mathsf{w})_{\mathcal{T}_h}, \\ h(\lambda,\mu) &= \langle(\tau - \boldsymbol{c}\cdot\boldsymbol{n})\lambda_h,\,\mathsf{w}\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D}\,, \end{aligned} \tag{2.91}$$

(see [191] for details), representing the implicit treatment of the convective and diffusive fluxes. The analogous discrete quadrature-free operators can be formed in a straightforward manner from the definitions given in §2.4.1.1.

Alternatively, the convective term can be treated explicitly and the problem (2.90) can be integrated in time with an HDG-IMEX scheme. HDG projection methods for the incompressible Navier–Stokes equations [246] can also be formulated as a series of implicit diffusion problems with explicit advection and can be advanced in time with IMEX schemes as well. We remark that the matrix-free discretizations and methods presented in this thesis §2.4.2 are agnostic to both these alterations, and can be applied to any of the aforementioned problems.

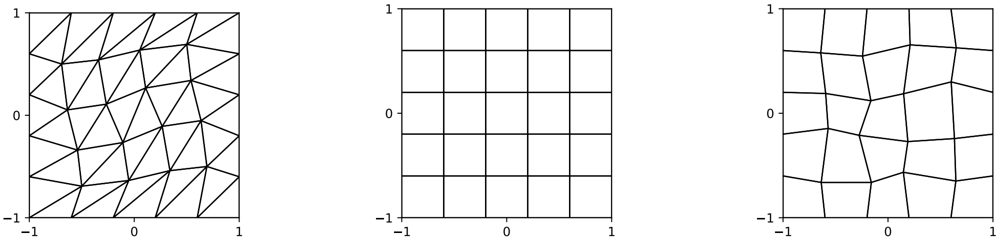
2.4.4 Numerical experiments and discussion

2.4.4.1 Accuracy of quadrature-free and quadrature-based integration

In the first example, we study the performance of quadrature-based and quadrature-free numerics for the diffusion-dominated case. We consider a steady diffusion problem (2.58) in which $\Omega = (-1,1)\times(-1,1)$, and $\kappa = 1$. We take the boundary Γ_N to be the left-side of the domain and $\Gamma_D = \Gamma\setminus\Gamma_N$. The source term and inhomogeneous boundary conditions g_D and g_N are chosen such that the exact solution is

$$u = \exp(x+y)\sin(\pi x)\sin(\pi y).$$

We consider a nodal basis [110] of polynomial order p defined on both triangular and quadrilateral meshes obtained by splitting the domain into a regular $n\times n$ Cartesian grid for a total of $2n^2$ or n^2 elements, respectively. Additionally, we consider meshes where the vertices have been perturbed as shown in Figure 2-15. We present the convergence data in L^2-norm for both quadrature-free (QF) and quadrature-based (QB) schemes in Figure 2-16. All quadrature rules can be found in [53, 54]. The HDG method results approximations which converge at optimal order $p+1$ for the scalar variable u_h. For the perturbed triangular mesh and the Cartesian quadrilateral mesh, we observe optimal convergence for both the QF and QB schemes. All triangular elements and affinely



(a) Cartesian triangular mesh with $h/10$ clockwise vertex perturbation.

(b) Cartesian quadrilateral mesh.

(c) Cartesian quadrilateral mesh with $h/10$ random vertex perturbation.

Figure 2-15: Mesh configurations for numerical experiment 2.4.4.1.

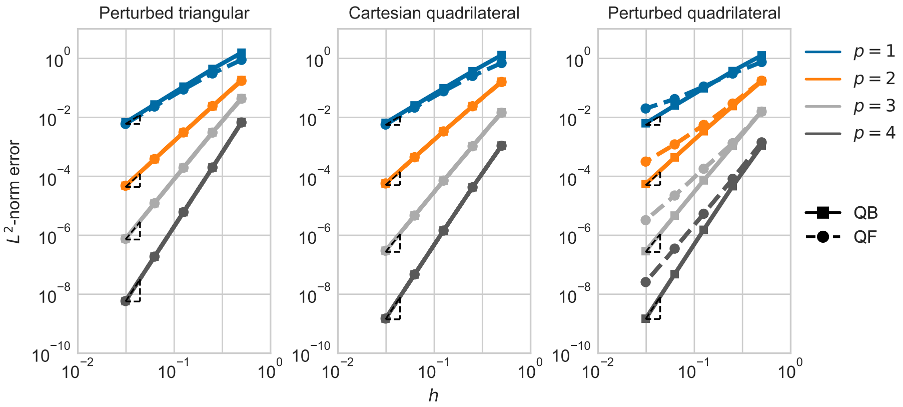


Figure 2-16: History of convergence in the L^2-norm for the numerical solution u_h on the classes of mesh in Figure 2-15. Dashed triangles indicate optimal convergence rates of order $p + 1$.

transformed quadrilateral elements can be represented exactly by a linear transformation from the master element, and therefore have no transformation error due to the isoparametric transformation.

On the other hand, the perturbed quadrilateral elements are non-affinely transformed from the master element, inducing integration errors. We observe a pollution in convergence order and a non-trivial loss in the accuracy of the QF approximation. Therefore, we heretofore assume straight-sided elements with spatially constant Jacobians over each elements. We remark that for unique elements where the transformation $\boldsymbol{X}(\boldsymbol{\xi})$ is non-linear—for example, with curvilinear finite elements representing a domain boundary—quadrature-based operators can be evaluated or stored for these elements in particular to avoid transformation or aliasing errors, while the rest of the mesh elements can be treated in a quadrature-free manner [110].

2.4.4.2 A 3D steady anisotropic diffusion problem

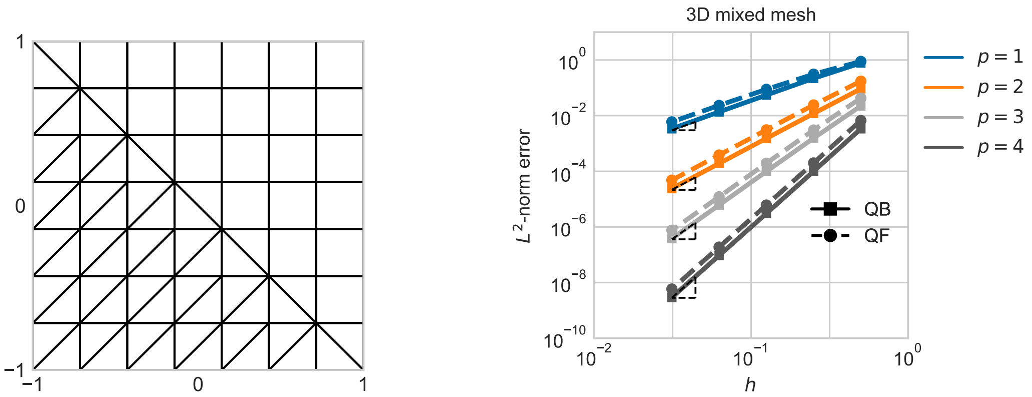
We consider a steady convection-diffusion problem a three-dimensional domain consisting of a six layer extrusion of the horizontal domain $\Omega_H = (-1, 1) \times (-1, 1)$, represented with a mesh consisting of wedge and hexahedral elements Figure 2-17a. We consider an anisotropic, spatially variable diffusion coefficient $\kappa = (1, 1, z^2)$. The source term f and boundary conditions are chosen such that we have the exact solution

$$u(x, y, z) = xyz\frac{(1 - e^{(x-1)})(1 - e^{(y-1)})}{(1 - e^{-x})(1 - e^{-y})}. \tag{2.92}$$

The convergence results are presented in Figure 2-17. We see that there is a negligible loss in accuracy for the quadrature-free scheme, and that both the QB and QF approaches converge optimally.

2.4.4.3 Matrix-free performance comparison

We consider the performance of our quadrature-free, matrix-free (QFMF) implementation in §2.4.2 compared to a classical quadrature-based, matrix-free implementation. The quadrature-based, matrix-free (QBMF) implementation applies the action of the left hand side operator $\mathbb{K}$ in (2.70), requiring the elemental assembly of the $\mathcal{A}_{\text{loc}}$ matrix and application of its inverse via a dense linear solve. We investigate the performance both for the 2D problem in §2.4.4.1 and the 3D problem in §2.4.4.2. All experiments are run using a 24-core dual-socket Intel Xeon E5-2680 v3 system running at 2.50 GHz. Figure 2-18 displays the wall clock time required to evaluate a matrix-free application of the operators $\mathbb{K}$ (QBMF) and $\tilde{\mathbb{K}}$ (QFMF) for a problem with around 5 million primal unknowns u_h. Even in 2D, the modified operator application $\tilde{\mathbb{K}}$ is over an order of magnitude faster than a standard quadrature-based matrix-free application of $\mathbb{K}$. In 3D, the performance gap is more



(a) Top-down view of a representative 3D wedge/hexahedral mixed mesh used in the anisotropic diffusion convergence study.

(b) History of convergence in the L^2-norm of the numerical solution u_h to the exact solution (2.92). Dashed lines indicate optimal convergence rates of order $p + 1$.

Figure 2-17: Representative computational mesh and convergence history for the anisotropic diffusion test case, §2.4.4.2

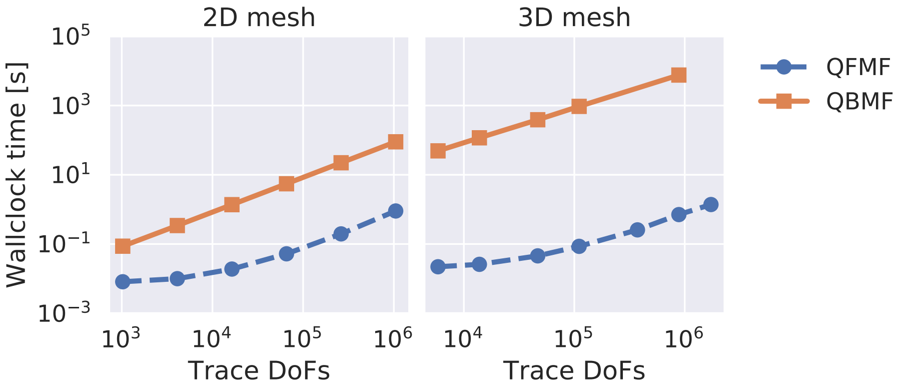


Figure 2-18: Comparison of wall-clock times for a single matrix-vector product as a function of problem size, measured in trace $\widehat{u}_h$ degrees of freedom, for 2D (left) and 3D (right) mixed meshes, at polynomial order $p = 3$ and $p = 5$, respectively.

dramatic, as the memory and computational requirements to form and apply the quadrature-based $\mathcal{A}_{\text{loc}}$ matrix become more expensive relative to S^{-1}, due to the problem dimension.

Figure 2-19 compares the computational throughput of the matrix-free application approaches as a function of polynomial order p. We observe a concomitant performance difference in throughput between the two algorithms, as measured in primal degrees of freedom u_h per second. The QFMF approach achieves more than an order of magnitude higher throughput at all polynomial orders, in both 2D and 3D. While the differences in performance by polynomial order is best explained by the particular computer architecture and cache sizes used for this experiment, the difference between the two algorithms can be explained by the higher arithmetic intensity resulting from repeatedly applying the templated operators associated with quadrature-free schemes, rather than the memory-bound formation of the quadrature-based operators. Namely, the broadcasting operations required to apply the transformational data $|J|$, $|J_e|$, and $\boldsymbol{n}$ to the templated operators M, $\mathsf{M}_{\partial\widehat{K}}$, and S are cheaper than computing the products and sums over the quadrature points necessary to form the quadrature-based operators. For 3D elements in particular, the QFMF approach scales much better than the QBMF approach at high polynomial order, due to the increase in size of the element

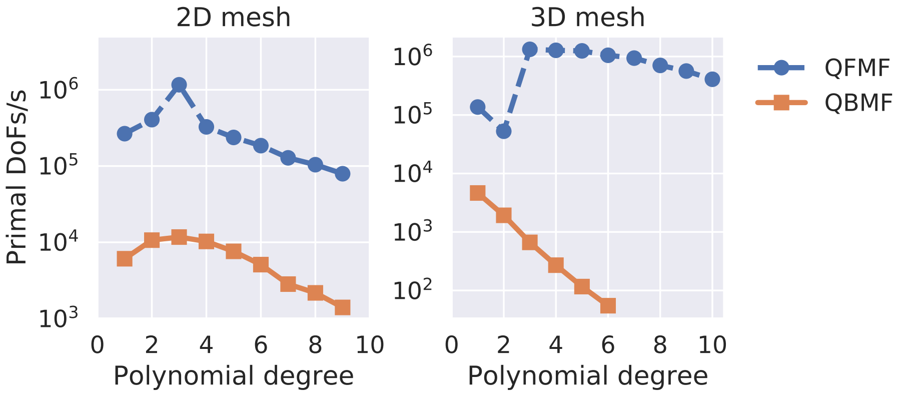


Figure 2-19: Comparison of throughput measured by DoF/s as a function of polynomial order for 2D and 3D matrix-free matrix-vector products.

local linear system $\mathcal{A}_{\text{loc}}$ for 2D and 3D problems as a function of polynomial order: $9(p+1)^2$ and $16(p+1)^3$, respectively.

2.4.5 Discussion

Classical matrix-free approaches for finite element schemes are straightforward because they require only an evaluation of the global basis functions and their gradients over the computational domain [131]. Matrix-free HDG schemes are more complicated because they involve a matrix inversion operation as well, due to the parametrization of the PDE to the trace unknown space. These inversions can become intractable when the problem size is sufficiently large that the matrix factorizations representing the inverses can no longer be stored. This is because a matrix inverse operation is required for each element at every application of the left hand side operator.

We have presented an efficient matrix-free scheme which takes advantage of quadrature-free operator formation to avoid these matrix inversions, requiring inversion of a much smaller operator. This reduction in memory is substantial and may indeed allow the storage of the smaller inverses, while storage of the original inverses would be infeasible. On the other hand, if neither can be stored, we demonstrated that this method results in significantly better performance over a wide range of problem sizes and polynomial orders with several numerical examples. Furthermore, we demonstrated that loss of accuracy due to the quadrature-free integration was negligible, provided that any irregular elements are handled separately, with quadrature. The performance gains associated with the QFMF method could also be augmented with standard parallelization approaches, further increasing throughput and decreasing time-to-solution.

2.5 On the solution of the pure Neumann problem

Projection methods are often used to solve the incompressible Navier–Stokes equations, and will form the backbone of the ocean equation model in §3. For these types of pressure-correction projection schemes [95], the second step involves computing a correction δp to the pressure by solving

$$-\nabla^2 \delta p = f(\bar{u}) \tag{2.93}$$

subject to the boundary conditions

$$\nabla \delta p \cdot \boldsymbol{n} = 0 \quad \text{ on } \Gamma_D, \tag{2.94}$$

where the right hand side data $f(\bar{u})$ is dependent on a predictor velocity $\bar{u}$ and Γ_D refers to the boundary condition on the velocity [77, 110, 246]. A common occurrence is the case of all Dirichlet boundaries on the velocity predictor, resulting in pure Neumann boundary conditions on the pressure correction δp. This so-called “pure Neumann problem” results in a pressure level defined only up to a constant. To see this, note that if the scalar field δp_h is a solution of (2.93), with $\Gamma_N = \partial\Omega$, then $\delta p_h + c$ for any $c \in \mathbf{R}$ is also a solution. Discretely, the null space of the linear system arising due to the discretization of (2.93) is spanned by the set of constant vectors, implying a singular linear system. Although crucial to the performance of pressure correction schemes, the issue of finding a solution to the rank-one deficient singular system in such cases receives little attention in the literature, outside of specialized literature for iterative solvers [18, 114, 7, 168, 167].

In [26], the authors rigorously describe strategies for addressing the singular system in a continuous finite element context. Here, we extend that discussion to the DG-FEM setting and illustrate computational trade-offs associated with different candidate approaches.

2.5.1 Proposed solutions

Subspace projection. As a first approach, we apply a subspace projection using a Krylov-solver [253], making use of the fact that iterative solvers solve singular systems provided that the right-hand side is in the orthogonal complement of the null-space. In summary, to perform this singularity treatment, we project out the null-space of the linear system by subtracting the mean from the right-hand side vector at each iteration of the iterative linear solve algorithm, in this case, conjugate gradient. We implemented this method by overloading the function for matrix-multiplication within the iterative solver in order to perform the mean removal at each iteration.

Penalty method. As a second approach, we apply a penalty-based method which can be interpreted as a regularization. This method, while simple and independent of linear solver type, involves the specification of a hyperparameter γ. Suppose that A is the discretized Laplacian operator in the Poisson system (2.93), such that we are solving $A\boldsymbol{x} = \boldsymbol{b}$. This problem is equivalent to the problem of minimizing the functional

$$J_1(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T A\boldsymbol{u} - \boldsymbol{b}^T\boldsymbol{u} \tag{2.95}$$

with optimal solution $u = \delta p$. Suppose that additionally, we would like to enforce that $\boldsymbol{e}^T\boldsymbol{x} = 0$, where $\boldsymbol{e}$ denotes some basis vector of $\mathbb{R}^n$. This is equivalent to fixing some pressure correction in the domain to 0. Therefore instead we seek to minimize the modified functional

$$J(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{u}^T A\boldsymbol{u} + \frac{\gamma}{2}\left(\boldsymbol{e}^T\boldsymbol{u}\right)^2 - \boldsymbol{b}^T\boldsymbol{u} \tag{2.96}$$

where γ is some penalization parameter which penalizes the deviation of the pressure-correction at that degree of freedom departing from zero. In that sense, this penalty approach can be thought of as a regularization, since γ must be chosen (as well as the degree of freedom on which to apply the penalty). Lastly, we assumed that $\boldsymbol{b} \in \operatorname{Im} A$, which is not true in general for a numerical discretization. Therefore, there is an additional condition to impose. Given that $\nabla^2(\alpha\boldsymbol{v}) = 0$, where $\boldsymbol{v} = [1, 1, \ldots, 1]^T$, we have that $\boldsymbol{v} \in \ker A$. Since the finite element discretization is symmetric, we have that $A = A^T$ and hence $\operatorname{Im} A \perp \ker A$. Further, since A is rank-1 deficient, we have that $\ker A$ is spanned by $\boldsymbol{v}$. If $\boldsymbol{w}^T\boldsymbol{v} = 0$ then $\boldsymbol{w} \in \operatorname{Im} A$. Therefore the space of zero-average vectors is orthogonal to $\ker A$, and a sufficient condition is that $\boldsymbol{v}^T\boldsymbol{b} = 0$ for the penalty method to work. Therefore subtracting the average of $\boldsymbol{b}$ projects the vector out of $\ker A$. Overall, the implementation is quite simple; to the row corresponding to the degree of freedom chosen add $\gamma/2$ to the diagonal, and subtract the mean of b from the right-hand side before beginning the linear solve. Roundoff error can produce low-quality solutions if γ is too large, so we choose γ to be on the same order of magnitude as the entries of A.

We remark that a popular alternative approach is to discretely impose the mean-value constraint

$$\int_\Omega \delta p \, d\Omega = 0 \tag{2.97}$$

in the linear system directly, avoiding the saddle point system that arises as a result of applying the constraint as a Lagrange multiplier [26]. In light of the discussion above, this would amount to replacing the unit basis vector $\boldsymbol{e}^T$ with $\mathbf{1}^T$, the vector of ones. However, this additional penalization constraint is not sparse on the row that it is applied to, and makes the underlying linear system non-symmetric. Therefore we do not consider this approach as a candidate.

Point constraint. An approach favored by many practitioners is to manually specify the value of the candidate solution at a single point by removing an equation from the discrete system and applying a Dirichlet constraint fixing the value of the solution at that point to an arbitrary constant, eliminating the null space and allowing solution of the linear system using a conventional direct solver. However, in a finite-element context, the function δ_p is often represented in the Sobolev space $H^1(\Omega)$ or $L^2(\Omega)$, spaces in which point evaluations do not make sense. In these cases, imposing such a constraint can render the variational problem ill-posed. A typical manner of fixing this is to specify the value of the solution along a measurable subset of the domain; typically an element face; while this constraint is well-posed, it's also undesirable, because specifying that the value of the numerical solution along the face will change the character of the numerical solution and thus the physics.

2.5.2 Computational benchmarks

In order to investigate the performance of each proposed method to solve the pure Neumann problem, we asses the convergence, accuracy, and efficiency of each method over a wide range of problem sizes and polynomial degrees p. We consider the analytical solution

$$\delta_p^* = \frac{1}{3} \sin\left(-\frac{\pi}{2}x\right) \cos\left(2\pi y\right), \tag{2.98}$$

(Figure 2-20, left) with boundary conditions and forcing function deduced from the exact solution over the computational domain $\Omega = [-1, 1]^2$. By symmetry, we have that the exact solution δ_p^* is analytically zero-mean in the sense of equation (2.97). Additionally, the value of the boundary condition $g_N = \nabla \delta_p^* \cdot \boldsymbol{n}$ vanishes algebraically over the boundary $\partial\Omega$, imitating the boundary conditions imposed in the pressure corrector step. We consider a uniform Cartesian grid consisting of quadrilateral elements of length $L/2^\ell$ in each spatial dimension, where the integer $\ell = 0, 1, \ldots$ denotes the level of refinement. To assess the convergence of the primal unknown δ_p, the integral mean level of the numerical solution is subtracted as post-processing to agree with the analytical solution. All iterative solves are performed using conjugate gradient (CG) iterations to solve for the trace unknowns $\hat{\delta}_{p,h}$, from which the primal solution $\delta_{p,h}$ and its gradient $\boldsymbol{q}_{\delta_p,h}$ are reconstructed.

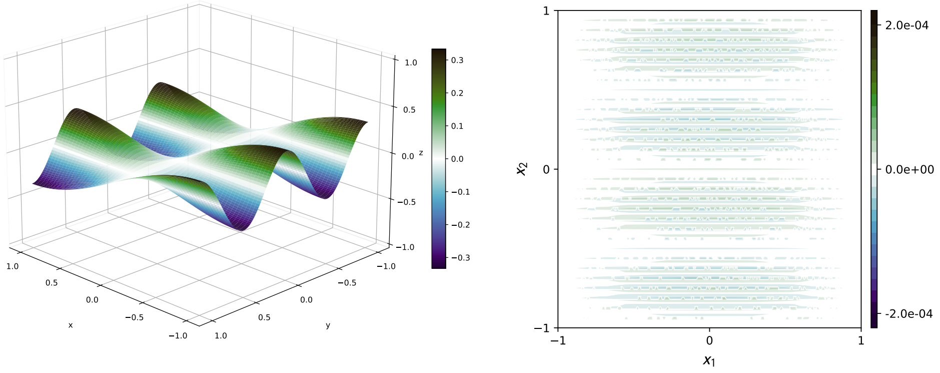


Figure 2-20: (Left) Analytical solution of the pure Neumann problem. (Right) Distribution of numerical errors in the gradient $\nabla \delta_p^* - \boldsymbol{q}_{\delta_p,h\delta_p,h}$ over the computational domain for a mesh of 32×32 elements at polynomial order $p = 3$.

All three methods achieve optimal $p+1$ order convergence to the exact solution in both the primal $\delta_{p,h}$ and gradient $\boldsymbol{q}_{\delta_p,h}$ unknowns, respectively, before the iterative solver tolerance and finite-precision effects begin to pollute the convergence order. Figure 2-21 illustrates the convergence history of the primal unknown $\delta_{p,h}$ for the projection approach, both from the perspective of mesh resolution and in globally-coupled trace unknowns. The errors and orders of convergence in the L^2-norm are presented in tabular form in 7.6. In what follows, we demonstrate the subspace projection approach to be the computationally superior method; therefore, we present the convergence results for the projection method and omit the others for brevity.

An examination of the spatial distribution of the errors in the numerical gradient[2] over the computational domain (Figure 2-20, right) illustrates that they are neither disproportionately large on the domain boundary, nor are they localized around a single point, problems that have been known to plague the penalization and point constraint methods, which depend on a specific choice of degree of freedom for their implementation [26, 95]. Given the comparable order of convergence to the analytical solution for all three approaches, we compare their relative computational advantages. Reporting a condition number for the resultant linear system solved with each method is not feasible. On one hand, the problem sizes reported in this work are large enough to make direct computation of the eigenvalue spectrum intractable. Moreover, as the subspace projection transformation is applied at each conjugate gradient iteration, there is not a single resultant linear system for which to compute a condition number. Therefore, as a practical indication of the conditioning of each algorithm, we examine the efficiency of the iterative solver. Figure 2-22a shows the number of conjugate gradient iterations required to achieve convergence to the specified solver tolerance (without preconditioning) over a wide range of problem sizes. We observe that the point constraint and penalization methods require many more iterations to converge than the subspace projection approach across polynomial orders, typically between a factor of two and a factor of ten. Since the subspace projection requires removal of the mean at every conjugate gradient iteration and the other two methods do not, we also benchmark the performance of each algorithm in terms of the overall L^2-error in the primal solution variable as a function of the wall clock time to solution of the linear system for the trace unknown $\hat{\delta}_{p_h}$ in Figure 2-22b. We observe that the subspace projection approach equals or outperforms the

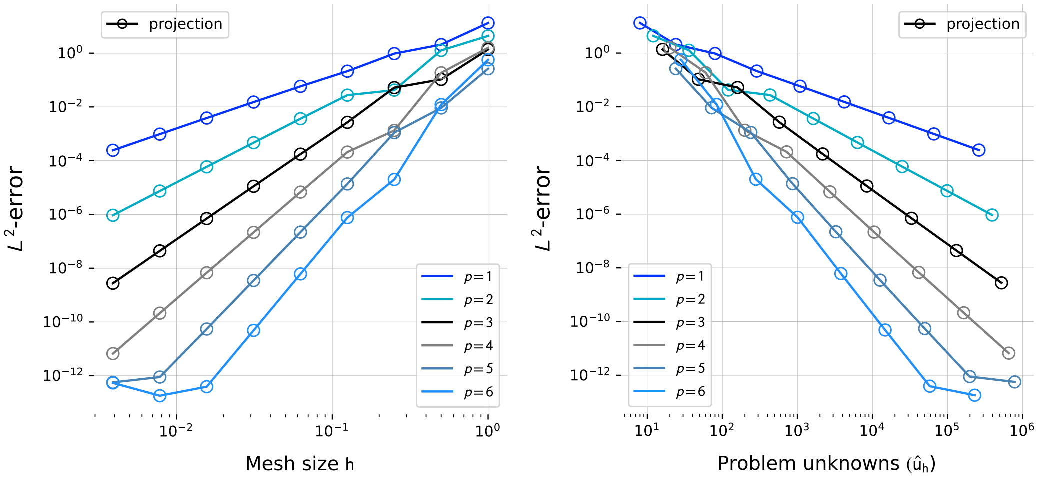
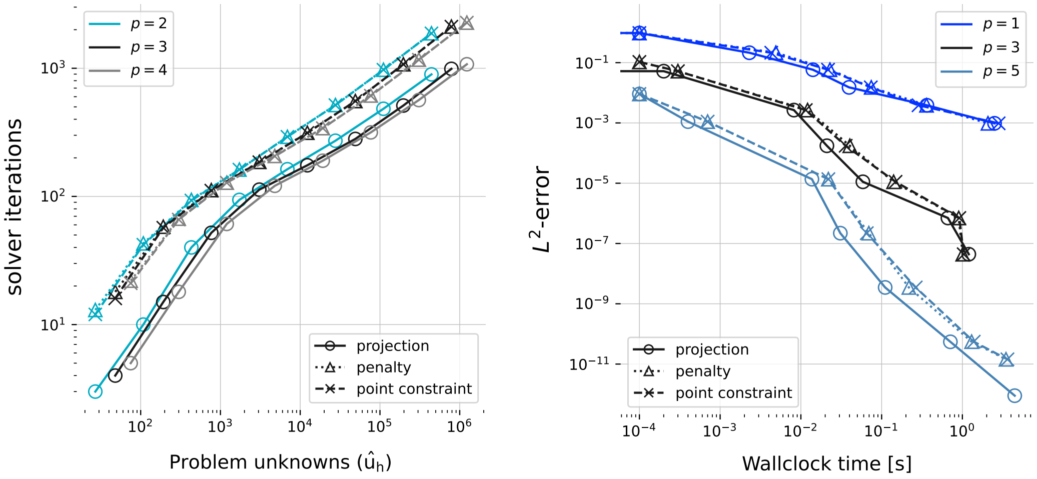


Figure 2-21: Errors in the primal unknown $\left\|\delta_p^* - \delta_{p,h}\right\|_{L^2(\Omega)}$ for the singular Neumann problem (2D) treated with subspace projection. Convergence is shown with respect to mesh size as well as number of globally-coupled unknowns. Optimal convergence is achieved for all polynomial orders. Convergence history data is tabulated in §7.6.

[2] We compute the directional error $\nabla\delta_p^* - \boldsymbol{q}_{\delta_p,h}$ over each element and plot the x_1-component. The results are similar for the gradient in the x_2 direction. We consider the gradient rather than the primal variable, because the gradient is the quantity that is used in the projection method detailed in §3.

other two methods over a wide range of problem sizes, with the advantage increasing as a function of polynomial order.



(a) (b)

Figure 2-22: Performance comparisons of the different singularity treatments. (Left) Number of conjugate gradient iterations required for convergence for each method at different problem sizes and polynomial orders. (Right) L^2-error achieved in the primal unknown $\delta_{p,h}$ as a function of wall clock time-to-solution of the global linear system for the trace unknown, shown at different problem sizes spanning from 10^1 to 10^6 trace degrees of freedom, and different polynomial orders.

In conclusion, we have found the subspace projection approach to be a stable, robust, and accurate method for numerically solving the singular system arising from the discretization of the pure Neumann problem occurring in the case of pure Dirichlet boundary conditions on the velocity. Furthermore, we have demonstrated it to be a superior method computationally, delivering better accuracy with a shorter time-to-solution than both the point-constraint and penalization approaches, and without the need to calibrate any additional hyperparameters or to single out a particular problem degree of freedom for the imposition of a constraint.

2.6 Conclusion

In this chapter, we formulated and derived the various DG-FEM and HDG-FEM weak forms that we will make use of in our nonhydrostatic finite element model, along with the finite element solvers used in other parts of this thesis, as referenced in sections §4 and §5. We provided a framework for treating inhomogeneous HDG boundary conditions at the level of the weak form that allows us to maintain the symmetry of linear systems that arise from our discretizations, a critical factor in ensuring the accuracy and efficiency of our computational models. Turning our attention to the computational aspects of HDG-FEM implementation, we completed benchmarking to analyze the multi-threaded parallel implementation of the HDG assembly and reconstruction algorithms. Our investigations spanned multiple problem sizes and polynomial orders for 2D and 3D problems, allowing us to thoroughly examine the effect of polynomial order on the parallel performance of our models. In addition to these investigations, we derived and implemented a novel matrix-free method tailored for HDG discretizations. This extended the research presented in [80], expanding upon the concepts and methodologies explored in that body of work. We then conducted a benchmarking exercise and comparison of different approaches to singularity removal. This was specifically focused on the rank-one-deficient linear systems that emerged from the discretization of the Poisson problem under pure Neumann boundary conditions, a common problem encountered in CFD but very seldom addressed in the literature. Our numerical experiments convincingly demonstrated that subspace projection was the superior method in terms of both accuracy and computational cost. Next, we will apply the theory, schemes, and results of the computational investigations in this chapter to the formulation of a high-order DG-FEM ocean model, allowing to see how our methods and approaches perform when applied to complex, real-world modeling scenarios.

Chapter 3

A Novel Nonhydrostatic HDG Ocean Model with a Free Surface

3.1 Introduction

Nonhydrostatic models are necessary to address the limitations of hydrostatic models when simulating certain types of fluid flows, especially those involving complex and highly dynamic processes occurring at the submesoscale (see §1.1). At these scales, on the order of less than 10 km in the horizontal, the hydrostatic approximation begins to lose validity, along with the hydrostatic ocean equations as a governing dynamical model. Nonhydrostatic effects become increasingly strong and presently are detectable at the small-scale end of the submesoscale [108, 172, 237, 180, 59, 240] and in deep convection [178]. At these scales, nonhydrostatic and fully three-dimensional oceanic phenomena (typically on the order of 100 m to 1 km), such as the breaking of internal waves, convection, subduction, and shear-induced overturning, become important [172, 240]. As a result, oceanic features with relatively strong vertical velocities have been observed at these scales, particularly in the presence of wind forcing or topographic variations [119, 171]. In turn, these dynamics play a crucial role in dissipative *diapycnal mixing*. This is when transient, interior regions of small-scale turbulence mix fluids of different densities. The effects of these submesoscale motions are largely unknown [83, 240], but must be investigated. This is because these nonhydrostatic dynamics may not only affect oceanic diapycnal transport, the dissipation of energy input at the surface, the spontaneous breakdown of mesoscale structures, and vertical transport in the ocean, but may also constitute a link between non-dissipative mesoscale flows and dissipative three-dimensional motions [84, 172, 174, 173]. Nonhydrostatic interactions between the submesoscale and the finescale are explored in [249].

The non-hydrostatic ocean model can make marine scientists capable of investigating almost the entire range of internal waves, convection processes, symmetric instability, etc. in realistic ocean motions and bridge the gap with LES (see [239] and references therein). It has also become a necessary tool for the simulation of surface dispersive waves internal waves [17, 252].

Relatively few ocean circulation models presently have nonhydrostatic capability, although they are increasingly becoming adopted and used within several models: MITgcm (M.I.T. General Circulation Model) [1, 178], POM (Princeton Ocean Model) [260], BOM (Bergen Ocean Model) [88], SUNTANS (Stanford Unstructured Non-hydrostatic Terrain-following Adaptive Navier–Stokes Simulator) [85, 115], ROMS (Regional Ocean Modeling System) [121], FVCOM (Finite-Volume Community Ocean Model) [140, 141], Symphonie [17], GETM (General Estuarine Transport Model) [126], and MERF3.0 (the Marine Environment Research and Forecasting model) [239] and `Oceananigans.jl` [214], to name a few.

However, the dominant technologies for solving problems of this nature are almost entirely low-order finite difference and finite volume methods; despite this, modern advances in CFD suggest that high-order methods may be better suited to the problem of nonhydrostatic ocean modeling. In addition to the numerical and computational advantages enjoyed by DG finite element methods (see §2.1-§2.2), they possess features that make them well-suited to multiscale ocean modeling applications.

On one hand, high-order finite element methods allow for arbitrarily high-order solutions on unstructured meshes and often provide higher accuracy for the same computational cost [110]. General unstructured meshes can more accurately capture complex geometrical features such as coastlines and steep bathymetry. Unstructured meshes also allow open boundaries to be moved further away from the model domain of interest for a small computational cost by using much larger elements near open boundaries, reducing the impact of the open boundary condition on the simulation. However, for dispersive wave behaviors such as those that occur at the submesoscale, evidence suggests that high-order methods may have additional advantages. These properties have already rendered DG-FEM a popular technique for solving the shallow water equations [62, 71, 92, 258, 257]. Evidence continues to emerge corroborating the effectiveness in a nonhydrostatic context as well. Additionally, the discontinuous polynomial spaces in which DG-FEM solutions are sought allow for the capture of steep gradients and wave behavior, resulting in more stable and flexible methods than the classical continuous Galerkin finite element (CG-FEM) approaches to advection-dominated problems [110], such as internal wave motion. For low-order finite volume methods specifically, the dominant mode of truncation error is dispersive, leading to a pollution of the physically correct dispersion behavior, imposing a strict resolution requirement on the method for applications such as resolving internal solitary wave behavior and coupled non-hydrostatic physical-biogeochemical dynamics; high-order methods and adaptive mesh refinement are potential solutions [251, 244].

Despite the advantages of DG-FEM, its use in nonhydrostatic ocean modeling is still in its infancy, with limited, but growing adoption thus far [202, 203, 204]. Part of the reason for its lack of adoption is due to the large number of degrees of freedom necessary to represent a high-order discontinuous polynomial solution on each element, which involves duplication of unknowns on solution faces. To address this issue, hybridizable discontinuous Galerkin (HDG) finite elements were devised by parametrizing the DG problem onto a finite element space on the mesh skeleton, as is discussed at great length in §2. This advance allows for substantial reduction of globally-coupled unknowns while retaining all the advantages of DG-FEM methods. The inclusion of HDG methods into multi-scale ocean modeling could significantly improve forecast accuracy and computational efficiency, motivating the developments of this chapter.

The contributions of this chapter are as follows. We present a novel HDG spatial discretization of the nonhydrostatic ocean equations with a free surface using a temporal projection scheme inspired by the scheme first derived in [243, 246, 247], applying recent findings on stability and robustness for DG-FEM for incompressible flow problems [77, 78] to HDG discretizations. We show that the DG-FEM model formulation allows for seamless nesting in larger hydrostatic models, and provide numerical evidence of the effectiveness of such an approach for multi-scale ocean modeling.

This chapter is structured in the following way. In §3.2, we introduce the mathematical model of the nonhydrostatic ocean equations with a free surface. Section §3.3 is devoted to the description of the temporal discretization of the governing equations using a projection method, and the spatial discretization of the projection method using a hybridizable discontinuous Galerkin method is detailed in §3.4. Modification of the HDG spatial discretization to the hydrostatic ocean equations is given in §3.4.9. Implementation details are discussed in §3.5.1. Verification of model convergence is given in §3.5.2. We provide numerical experiments to validate the model with several idealized non-hydrostatic test cases in §3.5, and provide comparison to a finite volume nonhydrostatic Boussinesq incompressible Navier–Stokes solver. We describe a method to nest the nonhydrostatic HDG model in a larger hydrostatic model in §3.6 and demonstrate the robustness of the DG-FEM model to the hydrostatic boundary conditions. In subsections §3.6.2 and §3.6.4, we show 2D and 3D numerical results of the HDG model nested within a larger hydrostatic model to perform hindcasts in the Alboran sea during a weather event in March of 2019, and highlight the nonhydrostatic dynamics the high-order model is able to capture. We summarize our results and provide conclusions in §3.7.

3.2 Mathematical model

We consider the non-hydrostatic ocean equations with a free surface on a two or three-dimensional domain $\Omega \subset \mathbb{R}^d$, derived from the incompressible Navier–Stokes equations with the Boussinesq approximation in a rotating reference frame:

$$\frac{\partial \boldsymbol{u}}{\partial t} - \nabla \cdot \boldsymbol{F}_v(\boldsymbol{u}) + \nabla \cdot \boldsymbol{F}_a(\boldsymbol{u}) + \nabla p' + g\nabla_{xy}\eta + \frac{1}{\rho_0}\int_z^\eta g\nabla_{xy}\rho'\,dz' = -\boldsymbol{f}_{\text{cor}} + \frac{1}{\rho_0}\boldsymbol{f} \tag{3.1}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.2}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot \left(\int_{-H}^{\eta} \mathbf{u}\,dz\right) = 0 \tag{3.3}$$

$$\frac{\partial \rho'}{\partial t} - \nabla \cdot (\kappa \nabla \rho') - \nabla \cdot (\boldsymbol{u}\rho') = f_{\rho'} \tag{3.4}$$

where the unknowns are the velocity $\boldsymbol{u} = (u_1, \ldots, u_d)^T$, the nonhydrostatic pressure p', the density perturbation ρ', and the free-surface elevation η defined over the top surface of the domain boundary $\partial\Omega_\eta \subset \mathbb{R}^{d-1}$, which coincides with the surface $z = 0$ in this work. We denote the depth of the bathymetry as the positive quantity $H(x, y)$. Figure 3-1 depicts these definitions for three-dimensional problems, and the equivalent for two-dimensional problems is shown in Figure 3-4. The density perturbation is defined in relation to the background density such that the total density $\rho = \rho_0 + \rho'$. The nonhydrostatic pressure is defined in relation to the total pressure and density

$$p = p_{\text{hyd}} + \rho_0 p', \qquad p_{\text{hyd}} = \int_z^\eta \rho g \, dz'. \tag{3.5}$$

We will refer to the velocity in the $\hat{\boldsymbol{z}}$ as w, and the velocity fields $\boldsymbol{u} = (u,\, w)$ and $\boldsymbol{u} = (u,\, v,\, w)$ in two and three spatial dimensions, respectively. We define the horizontal velocity $\mathbf{u} = u$ and $\mathbf{u} = (u, v)$ in two and three dimensions, respectively.

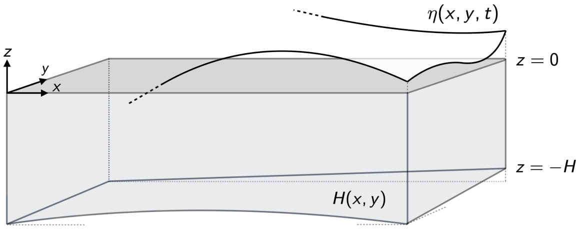


Figure 3-1: Schematic of domain. The interior domain Ω is shown in light grey and the domain $\partial\Omega_\eta$ over which the free surface $\eta(x, y, t)$ is defined is indicated using darker grey. The domain bathymetry $H(x, y)$ is taken to be positive.

Similarly, we define the horizontal and vertical gradient operators as

$$\nabla_{xy} \equiv \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, 0\right], \qquad \nabla_z \equiv \left[0,\, 0,\, \frac{\partial}{\partial z}\right] \tag{3.6}$$

in three spatial dimensions, with analogous definitions for two; complete reductions of the projection schemes in 2D are given in Appendix 7.8. The generic body force vector is denoted by $\boldsymbol{f}$ and the Coriolis force by $\boldsymbol{f}_{\text{cor}} = f_c \hat{\boldsymbol{z}} \times \mathbf{u}$ with Coriolis parameter f_c. The advection term is written in conservative (divergence) formulation, where the advective flux is $\boldsymbol{F}_a(\boldsymbol{u}) = \boldsymbol{u} \otimes \boldsymbol{u}$. The diffusion term is written in the Laplace form, with the diffusive flux given as $\boldsymbol{F}_v(\boldsymbol{u}) = \nu \nabla \boldsymbol{u}$, where the quantity $\nu \in \mathbb{R}^{d\times d}$ is a diagonal second-order tensor describing the direction-dependent kinematic viscosity. Similarly, $\kappa \in \mathbb{R}^{d\times d}$ describes the direction-dependent tracer diffusivity. For a detailed discussion and justification of the modeling approximations leading to this model, see [243].

The ocean equations (3.1)-(3.4) are subject to the initial conditions

$$\boldsymbol{u}(\boldsymbol{x}, t=0) = \boldsymbol{u}_0(\boldsymbol{x}),\ \rho'(\boldsymbol{x}, t=0) = \rho_0'(\boldsymbol{x}) \text{ in } \Omega, \qquad \eta(\boldsymbol{x}, t=0) = \eta_0(\boldsymbol{x}) \text{ in } \partial\Omega_\eta, \tag{3.7}$$

and on the boundary $\partial\Omega = \Gamma$, Dirichlet and Neumann boundary conditions are prescribed for the velocity, free surface, and density perturbation:

$$\begin{aligned} \boldsymbol{u} = \boldsymbol{g}_u \text{ on } \Gamma_D \times [0,T], &\qquad \boldsymbol{F}_v(u) \cdot \boldsymbol{n} = g_N \text{ on } \Gamma_N \times [0,T] \\ \eta = g_\eta \text{ on } \Gamma_D \times [0,T], &\qquad \nabla\eta \cdot \boldsymbol{n} = g_{N_\eta} \text{ on } \Gamma_N \times [0,T] \\ \rho' = g_{\rho'} \text{ on } \Gamma_D \times [0,T], &\quad \kappa \nabla\rho' \cdot \boldsymbol{n} = g_{N_{\rho'}} \text{ on } \Gamma_N \times [0,T] \end{aligned} \tag{3.8}$$

where $\boldsymbol{n}$ denotes the outward unit normal vector.

The hydrostatic equations are the result of neglecting the non-hydrostatic pressure variation by taking $p' \approx 0$ in equations (3.1) - (3.4):

$$\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} - \nabla\cdot\boldsymbol{F}_v(\boldsymbol{u}) + \nabla\cdot\boldsymbol{F}_a(\boldsymbol{u}) + g\nabla_{xy}\eta + \frac{1}{\rho_0}\int_z^{\eta} g\nabla_{xy}\rho'\,dz' &= -\boldsymbol{f}_{\mathrm{cor}} + \frac{1}{\rho_0}\boldsymbol{f} \\
\nabla\cdot\boldsymbol{u} &= 0 \\
\frac{\partial \eta}{\partial t} + \nabla\cdot\left(\int_{-H}^{\eta}\boldsymbol{u}\,dz\right) &= 0 \\
\frac{\partial \rho'}{\partial t} - \nabla\cdot(\kappa\nabla\rho') - \nabla\cdot(\boldsymbol{u}\rho') &= f_{\rho'}
\end{aligned} \tag{3.9}$$

with the same boundary and initial conditions.

3.3 Temporal discretization

The following subsections detail the temporal discretization for the nonhydrostatic ocean equation projection schemes, following [243]. The scheme is a pressure correction method similar to the schemes in [95] that splits the pressure into a nonhydrostatic component and a free-surface component, applying a pressure correction for each. For more detailed treatment as well as theoretical properties of these schemes, we refer the reader to [95] and the references therein. The algorithm is shown for 3D problems, but its explicit reduction for 2D problems is given in Appendix §7.8.

To discretize the ocean equations (3.1)-(3.4) in time, the time interval $[0, T]$ is divided into N_T uniformly-sized time steps $\Delta t = T/N_T$, admitting the set of discrete solution times $\{t_i\}_{i=0}^{N_T} = \{i\Delta t\}_{i=0}^{N_T}$, where we denote the time-step number $k = 0, \dots, N-1$. The equations are advanced from time $t_k = k\Delta t$ to $t_{k+1} = (k+1)\Delta t$ during time step k. Since the nonlinear terms in the Navier–Stokes equations does not affect the convergence rate of the error incurred due to operator splitting [95], we apply a semi-explicit treatment to the governing equations, treating the diffusion terms implicitly and the advection, pressure gradient, and density perturbation terms explicitly over each time step.

3.3.0.1 Velocity predictor

The momentum equation is solved in the first sub-step. We construct an intermediate velocity field $\bar{\boldsymbol{u}}^{k+1}$ which is not divergence free

$$\frac{\bar{\boldsymbol{u}}^{k+1}}{a\Delta t} - \left(\nabla\cdot\left(\nu_{xy}\nabla_{xy}\bar{\mathbf{u}}^{k+1}\right) + \nabla\cdot\left(\nu_z\nabla_z\bar{\mathrm{w}}^{k+1}\right)\right) + \nabla p'^{,k} + g\nabla_{xy}\eta^k = \boldsymbol{f}_{\bar{\mathbf{u}}}^{k,k+1}, \tag{3.10}$$

where the term $\boldsymbol{f}_{\bar{\mathbf{u}}}^{k,k+1}$ contains the advective, density perturbation, and Coriolis forcing terms in (3.1). The boundary conditions on the intermediate velocity field $\bar{\boldsymbol{u}}$ are

$$\bar{\boldsymbol{u}}^{k+1} = \boldsymbol{g}_D \qquad \text{on } \Gamma_D, \tag{3.11}$$

$$\nabla\bar{\boldsymbol{u}}^{k+1}\cdot\boldsymbol{n} = g_N \qquad \text{on } \Gamma_N. \tag{3.12}$$

we write (3.10) in a direction-split manner to emphasize that several choices exist to treat the diffusive terms; in the original scheme laid out in [243], the horizontal diffusion terms are treated explicitly while the vertical terms are treated implicitly. In this work, we treat the entire diffusion operator implicitly, for the following reasons. First, it is well-known that for both the incompressible Navier–Stokes equations [110] as well as the nonhydrostatic ocean equations [252], the pressure-correction equation §3.3.0.4 is the dominant computational cost for projection-based schemes, due to the poor conditioning of the pressure Poisson system, an assertion our experience has corroborated. The velocity predictor equations, by contrast, tend to be well-conditioned due to the presence of the time derivative term. Therefore, any computational gain offered by the explicit treatment of the horizontal diffusivity does not ultimately address the limiting computational bottleneck of the projection scheme, and introduces significant complexity in the derivation and implementation of a consistent and stable finite element spatial discretization.

3.3.0.2 Free-surface correction

In the second sub-step, we solve a $d-1$ dimensional problem for the change in the free-surface, $\delta\eta \in \partial\Omega_\eta$:

$$\frac{\delta\eta^{k+1}}{a\Delta t} - \nabla\cdot\left(a\Delta t g\left(\eta^k + H\right)\nabla\delta\eta^{k+1}\right) = -\nabla\cdot\int_{-H}^{\eta^k}\bar{\mathbf{u}}^{k+1}\,dz \tag{3.13}$$

subject to boundary conditions

$$\begin{aligned} \delta\eta^{k+1} &= g_\eta^{k+1} - g_\eta^k && \text{on } \Gamma_D, \\ \nabla\delta\eta^{k+1}\cdot\boldsymbol{n} &= \frac{1}{a\Delta t g\left(H+\eta^k\right)}\int_{-H}^{\eta^k}\left(\bar{\mathbf{u}}^{k+1} - \boldsymbol{g}_u\right)\cdot\boldsymbol{n}\,dz && \text{on } \Gamma_N, \end{aligned} \tag{3.14}$$

the derivation of which is provided in Appendix §7.7. Application of the Neumann condition imposes the stress at the boundary of the free surface, and a common choice is either a free stress condition imposed as $g_N = 0$ on closed boundaries (since $\mathbf{u}$ and $\boldsymbol{g}_u$ vanish), or a known stress imposed on the free-surface correction directly. Modeling open boundaries, in which both free surface and interior stresses are unknown, remains a challenging open research problem, and many approaches exist [227], any of which can be used to choose the values of g_η or a stress.

3.3.0.3 Intermediate velocity projection, free-surface update

The third sub-step of the projection method involves performing a correction to the predictor velocity due to the free surface pressure to compute the intermediate velocity $\bar{\bar{\boldsymbol{u}}}$ in addition to the corrected free surface value η^{k+1}.

$$\bar{\bar{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - (a\Delta t)g\nabla_{xy}\delta\eta^{k+1} \tag{3.15}$$

$$\eta^{k+1} = \eta^k + \delta\eta^{k+1}, \tag{3.16}$$

We note that the correction to the predictor velocity $\bar{\boldsymbol{u}}$ occurs only in the horizontal, and we refer the reader to [243] for a detailed derivation and discussion of this intermediate projection.

3.3.0.4 Pressure correction

In the fourth sub-step, we solve a classical pressure-Poisson equation to project the stage-final velocities into the space of divergence-free vector fields [95, 77]

$$\nabla^2\delta p^{\prime,k+1} = \frac{\nabla\cdot\bar{\bar{\boldsymbol{u}}}^{k+1}}{a\Delta t}. \tag{3.17}$$

subject to the boundary conditions

$$\begin{aligned} \nabla\delta p^{\prime,k+1}\cdot\boldsymbol{n} &= 0 && \text{on } \Gamma_D \\ \delta p^{\prime,k+1} &= 0 && \text{on } \Gamma_N \end{aligned} \tag{3.18}$$

where the boundaries Γ_D and Γ_N in (3.18) refer to those on the velocity $\boldsymbol{u}$ in (3.8).

3.3.0.5 Velocity projection, pressure update

In the fifth sub-step, we project the stage-final velocities $\boldsymbol{u}^{k+1}$ onto the space of divergence-free vector fields and update the nonhydrostatic pressure

$$\boldsymbol{u}^{k+1} = \bar{\bar{\boldsymbol{u}}}^{k+1} - a\Delta t \nabla \delta p^{\prime,k+1}, \tag{3.19}$$

$$p^{\prime,k+1} = p^{\prime,k} + \delta p^{\prime,k+1}. \tag{3.20}$$

3.3.0.6 Tracer evolution equation

Once the stage-final velocities $\boldsymbol{u}^{k+1}$ have been computed, we solve an advection-diffusion equation to update the density perturbation field ρ'

$$\frac{{\rho'}^{k+1}}{a\Delta t} - \nabla \cdot \left(\kappa \nabla {\rho'}^{k+1}\right) - \nabla \cdot \left(\boldsymbol{u}^{k+1} {\rho'}^{k+1}\right) = \frac{{\rho'}^{k}}{a\Delta t} + f_{\rho'}. \tag{3.21}$$

with boundary conditions exactly as in (3.8).

3.4 Spatial discretization

3.4.1 Notation

We now provide the notation for the computational mesh and finite-element operations (§3.4). We let $\mathcal{T}_h = \cup_i K_i$ be a finite collection of non-overlapping elements K_i that discretizes the entire problem domain $\Omega \subset \mathbb{R}^d$. We refer to the boundary of the problem domain as Γ. The set $\partial\mathcal{T}_h = \{\partial K : K \in \mathcal{T}_h\}$ refers to all boundary edges and interfaces of the elements, where ∂K is the boundary of element K. For two elements K^+ and K^- sharing an edge, we define $e = \partial K^+ \cap \partial K^-$ as the edge between elements K^+ and K^-. Each edge can be classified as belonging to either ε° or ε^∂, the set of interior and boundary edges, respectively, with $\varepsilon = \varepsilon^\circ \cup \varepsilon^\partial$. These geometric relationships are summarized in Figure 3-2a.

The elements K^+ and K^- have outward pointing unit normals $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$, respectively. The quantities $a^\pm$ denote the traces of a on the edge e from the interior of $K^\pm$. When relevant for element-wise operations, we take as convention that the element K^- refers to the local element, and K^+ to the neighboring element. The jump $[\![\cdot]\!]$ operator for scalar quantities are then defined as $[\![a]\!] = a^- - a^+$ on the interior faces $e \in \varepsilon^\circ$. On the edges described by $\partial\mathcal{T}_h$, we can uniquely define the normal vector $\boldsymbol{n}$ as outward for a given cell and inward for its neighbor.

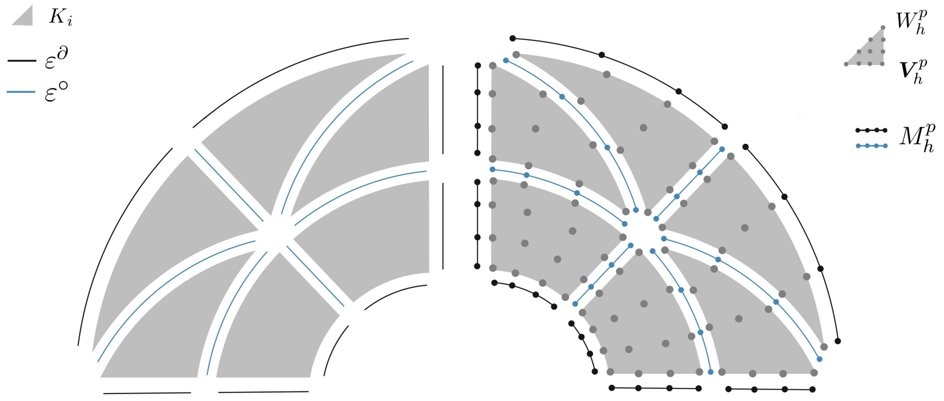
We define the inner products over a set $D \subset \mathbb{R}^d$ and its boundary $\partial D \subset \mathbb{R}^{d-1}$ using typical discontinuous Galerkin finite element notation as

$$(c, d)_D = \int_D cd \, \mathrm{d}D, \qquad \langle c, d\rangle_{\partial D} = \int_{\partial D} cd \, \mathrm{d}\partial D.$$

Let $\mathcal{P}^{p_{\text{order}}}(D)$ denote the set of polynomials of degree p_{order} on a domain D. We consider the discontinuous finite element spaces

$$\begin{aligned}
\boldsymbol{G}_h^{p_{\text{order}}} &= \left\{ \boldsymbol{G} \in \left[L^2(\Omega)\right]^{d\times d} : \boldsymbol{G}\big|_K \in [\mathcal{P}^{p_{\text{order}}}(K)]^{d\times d} \, \forall K \in \mathcal{T}_h \right\}, \\
\boldsymbol{V}_h^{p_{\text{order}}} &= \left\{ \boldsymbol{v} \in \left[L^2(\Omega)\right]^{d} : \boldsymbol{v}\big|_K \in [\mathcal{P}^{p_{\text{order}}}(K)]^{d} \, \forall K \in \mathcal{T}_h \right\}, \\
W_h^{p_{\text{order}}} &= \left\{ w \in L^2(\Omega) \colon w\big|_K \in \mathcal{P}^{p_{\text{order}}}(K) \, \forall K \in \mathcal{T}_h \right\}, \\
M_h^{p_{\text{order}}} &= \left\{ \mu \in L^2\left(\varepsilon_h\right) : \mu\big|_e \in \mathcal{P}^{p_{\text{order}}}(e) \, \forall e \in \varepsilon_h \right\},
\end{aligned}$$

where $L^2(\Omega)$ is the space of square-integrable functions on the domain Ω. Informally, $W_h^{p_{\text{order}}}$ represents the space of piecewise discontinuous polynomials of degree at most p_{order} on every element in the mesh (see Figure 3-2b). We will also refer to the analogous vector-valued trace space $\boldsymbol{M}_h^p$ for vector-valued quantities on the edge space. We refer the reader to [193] for extended discussion of these finite element spaces.



(a) Domain geometry

(b) Finite element spaces

Figure 3-2: Summary schematics of spatial discretization notation in exploded view.

3.4.2 Discretization of vertical integral terms

Forcing due to the free-surface pressure in the momentum equation can be expressed component-wise as

$$F_\eta(z) = -\int_z^\eta g\nabla_{xy}\rho'(x,y,z',t)\,dz' = -\begin{bmatrix} \int_z^\eta g\frac{\partial\rho'(x,y,z',t)}{\partial x}\,dz' \\ \int_z^\eta g\frac{\partial\rho'(x,y,z',t)}{\partial y}\,dz' \\ 0 \end{bmatrix} \tag{3.22}$$

Therefore, it suffices to compute explicit terms of the form

$$\Phi(x,y,z) = \int_z^\eta \phi(x,y,z')\,dz', \tag{3.23}$$

which allows the computation of the vertical integral terms on the right-hand side of equations (3.10) and (3.13). In what follows, we develop a simple and efficient DG scheme to compute these terms.

3.4.2.1 Depth-marching approach

For every horizontal location (x_0, y_0) and time t, we want to evaluate the integral $\Phi(z) = \int_z \phi(z')\,dz'$. Applying the fundamental theorem of calculus, $\frac{\partial}{\partial z}\Phi(z) = \phi(z)$. Since $\phi(\boldsymbol{x})$ can be evaluated at every point in space, this relation yields an ODE $\frac{d\Phi}{dz} = \phi(z)$ in depth. Choosing an extruded mesh guarantees that the elemental degrees of freedom are vertically aligned, and therefore the ODE could be solved as a set of 1D problems on each element. However, since the finite element solution is three-dimensional, doing so could violate the conservative nature of each DG field, and could not be applied to curvilinear elements. However, we can formulate a 3D finite element problem which overcomes both difficulties.

By considering the vector field $\Phi\hat{\boldsymbol{z}}$, we can discretize the ODE using a standard discontinuous Galerkin scheme with the following weak form: we seek $\Phi \in W_h^p$ such that

$$-\left(\nabla w,\, \Phi\hat{\boldsymbol{z}}\right)_K + \left\langle w,\, \widehat{\Phi}(\hat{\boldsymbol{z}}\cdot\boldsymbol{n})\right\rangle_{\partial K} = (w,\, \phi)_K \tag{3.24}$$

for all $w \in W_h^p$.[1]The presence of the dot product $\hat{\boldsymbol{z}}\cdot\boldsymbol{n}$ cancels all edge terms except those on the top and bottom of the element, but it remains to specify the numerical flux $\hat{\Phi}$. For a DG formulation, choosing $\hat{\Phi}(z_n) = \lim_{\epsilon\to 0}\Phi(z_n - \epsilon)$—that is, taking the interface value of Φ from the element above—leads to an explicit DG scheme that is consistent, stable, and can be evaluated element-by-element

[1] The formulation in [243] is the strong DG form of the problem. Here the standard weak form given to agree with standard DG literature.

from top to bottom [39].[2]Given that the topmost element coincides with the free surface η, we can simply solve the problem (3.24) on every cell element-by-element in the vertical column, iterating from top to bottom.[3]

On each cell, we solve the following DG problem: seek $u \in W_h^p$ such that

$$-\left(\nabla w,\, u\hat{\boldsymbol{z}}\right)_K + \left\langle w,\, \widehat{u}(\hat{\boldsymbol{z}}\cdot\boldsymbol{n})\right\rangle_{\partial K} = \left(w,\, f\right)_K \tag{3.25}$$

for forcing data f. Since the cells are extruded in the z-direction, the dot product $(\hat{\boldsymbol{z}}\cdot\boldsymbol{n})$ cancels for all faces except the top and bottom faces. We take the flux $\hat{u}$ to be defined as the value of u on the neighboring cell above on the top face, and take the interior value of u on K for the bottom face, leading to

$$-\left(\nabla w,\, u\hat{\boldsymbol{z}}\right)_K + \left\langle w,\, u(\hat{\boldsymbol{z}}\cdot\boldsymbol{n})\right\rangle_{\partial K_{\text{bottom}}} = \left(w,\, f\right)_K - \left\langle w,\, \widehat{u}(\hat{\boldsymbol{z}}\cdot\boldsymbol{n})\right\rangle_{\partial K_{\text{top}}} \tag{3.26}$$

3.4.3 Velocity predictor

3.4.3.1 Coupled-velocity approach

Since HDG methods are mixed methods, we first express each of the equations in §3.3 as first-order systems. We will refer to the velocity gradient $\boldsymbol{L} = \nabla\boldsymbol{u}$ with the tensor conventions

$$(\nabla\boldsymbol{u})_{ij} = \frac{\partial u_i}{\partial x_j}, \qquad (\nabla\cdot\boldsymbol{L})_i = \frac{\partial L_{ij}}{\partial x_j}$$

For equation (3.10) we define the auxiliary variable $\bar{\boldsymbol{L}} = \nu\nabla\bar{\boldsymbol{u}}$. Then the weak problem is as follows: find $\left(\bar{\boldsymbol{u}}_h^{k+1},\, \bar{\boldsymbol{L}}_h,\, \hat{\boldsymbol{u}}_h\right) \in (\boldsymbol{G}_h \times \boldsymbol{V}_h \times \boldsymbol{M}_h)$ such that

$$\begin{aligned}
&(\boldsymbol{G},\, \nu^{-1}\bar{\boldsymbol{L}}_h)_{\mathcal{T}_h} + (\nabla\cdot\boldsymbol{G},\, \bar{\boldsymbol{u}}_h^{k+1})_{\mathcal{T}_h} - \langle\boldsymbol{G}\cdot\boldsymbol{n},\, \hat{\boldsymbol{u}}_h\rangle_{\partial\mathcal{T}_h} = 0\\
&\left(\boldsymbol{v},\, \frac{\bar{\boldsymbol{u}}^{k+1}}{a\Delta t}\right)_{\mathcal{T}_h} - \left(\boldsymbol{v},\, \nabla\cdot\bar{\boldsymbol{L}}_h\right)_{\mathcal{T}_h} + \langle\boldsymbol{v},\, \tau\left(\bar{\boldsymbol{u}}_h^{k+1} - \hat{\boldsymbol{u}}_h\right)\rangle_{\partial\mathcal{T}_h}\\
&\qquad = -a_h(\boldsymbol{u}^k, \boldsymbol{g}_D) - \mathrm{pg}_h\left(\boldsymbol{v}, p_h^k\right) - (\boldsymbol{v},\, \boldsymbol{F}_{\rho'} + \boldsymbol{F}_\eta + \boldsymbol{F}_{\text{cor}})_{\mathcal{T}_h} + (\boldsymbol{v},\, \boldsymbol{F}_t)_{\mathcal{T}_h}\\
&\left\langle\boldsymbol{\mu},\, (-\nu\bar{\boldsymbol{L}}_h)\boldsymbol{n} + \tau(\bar{\boldsymbol{u}}_h^{k+1} - \hat{\boldsymbol{u}}_h)\right\rangle_{\partial\mathcal{T}_h\setminus\Gamma_D} + \left\langle\boldsymbol{\mu},\, \widehat{b}_h\right\rangle_{\Gamma_N} = 0
\end{aligned} \tag{3.27}$$

for all $(\boldsymbol{G}, \boldsymbol{v}, \boldsymbol{\mu}) \in (\boldsymbol{G}_h \times \boldsymbol{V}_h \times \boldsymbol{M}_h)$. Here, ν is a second order tensor contracted with L_h as $\nu_{ij}L_{ij}$ to apply the appropriate diffusivity in the horizontal or vertical direction. We define the body forcing terms

$$\boldsymbol{F}_{\rho'} = 1/\rho_0 \int_z^{\eta^k} g\nabla_{xy}\rho'^{,k}\, dz', \qquad \boldsymbol{F}_\eta = g\nabla_{xy}\eta^k, \qquad \boldsymbol{F}_{\text{cor}} = f_c\hat{z}\times\boldsymbol{u}^k, \qquad \boldsymbol{F}_t = \frac{1}{a\Delta t}\boldsymbol{u}^k, \tag{3.28}$$

which are all elements of the space $\boldsymbol{V}_h$ and evaluated at time k.

Advection operator. To derive the discontinuous Galerkin formulation of the advection term, we integrate the advection term $(\boldsymbol{v},\, \nabla\cdot\boldsymbol{F}_a(\boldsymbol{u}_h^k))_K$ by parts to yield

$$a_h(\boldsymbol{v}, \boldsymbol{u}_h^k,\, \boldsymbol{g}_D) = -\left(\nabla\boldsymbol{v},\, \boldsymbol{F}_a(\boldsymbol{u}_h^k)\right)_K + \left\langle\boldsymbol{v},\, \boldsymbol{F}_a^*(\boldsymbol{u}_h^k, \boldsymbol{g}_D)\right\rangle_{\partial K}, \tag{3.29}$$

where $\boldsymbol{F}_a^*(\boldsymbol{u}_h)$ is the numerical flux. Several choices exist, including an upwind, central, or local Lax-Friedrichs flux.

[2] The choice of numerical flux $\hat{\Phi}$ is not a strictly trivial one. Simply choosing $\hat{\Phi}$ to be the trace value on every element leads to a completely decoupled system which is not solvable. Other choices of numerical flux can lead to schemes that must be solved implicitly on the entire 1D domain, or schemes that are not consistent, so the choice of numerical flux is not pedantic.

[3] It is also possible to derive a Poisson problem for Φ on each vertical column and solve it using an HDG scheme. However, each of these problems must be solved implicitly over the entire column, and requires two boundary conditions.

Pressure gradient operator. We apply the same procedure as above to obtain the DG formulation of the pressure gradient term $(\boldsymbol{v},\, \nabla p')_K$:

$$\mathrm{pg}_h(\boldsymbol{v}, p_h^k) = -\left(\nabla\cdot\boldsymbol{v},\, p_h^{\prime,k}\right)_K + \langle \boldsymbol{v},\, (p_h')^*\boldsymbol{n}\rangle_{\partial K}\,, \tag{3.30}$$

where the numerical flux is taken to be a central flux: $(p_h')^* = \{\{p_h'\}\}$. We remark that DG formulations in which the pressure gradient term is not integrated by parts, i.e., $(\boldsymbol{v},\, \nabla p_h')_K$ as in [110] have been shown to lead to instabilities in the context of high-order discontinuous Galerkin discretizations of projection methods [77].

3.4.3.2 Component-wise velocity approach

In the case where the velocity components are not coupled on the boundary of the mesh due to the boundary conditions, for example, when a zero Dirichlet no-slip condition is applied to the velocity along a non-trivial bathymetry, and the vertical sides of the domain are axis-aligned, the linear system arising from equation (3.27) is decoupled, and the predictor velocity can be solved component-by-component. This situation also commonly arises in modeling of the mixed layer, where the domain is flat-bottomed and axis aligned. Computationally, this approach is attractive, and allows the solution of smaller element-local systems and a smaller globally-coupled linear system than the weak form given in §3.4.3.1.

Let $\bar\phi$ represent any component of the velocity $\bar{\boldsymbol{u}}$. If we were to write the usual form of the HDG problem where $\boldsymbol{q} = \nabla\bar\phi$ represents the complete gradient of the component $\bar\phi$, then we want to find $(\bar\phi, \boldsymbol{q}) \in W_h \times \boldsymbol{V}_h$ such that

$$\begin{aligned} \left(\boldsymbol{v},\, \nu^{-1}\boldsymbol{q}\right)_{\mathcal{T}_h} + \left(\nabla\cdot\boldsymbol{v},\, \bar\phi\right)_{\mathcal{T}_h} - \left\langle \boldsymbol{v}\cdot\boldsymbol{n},\, \hat\phi\right\rangle_{\partial\mathcal{T}_h} &= 0 \\ \left(w,\, \frac{\bar\phi}{a\Delta t}\right)_{\mathcal{T}_h} + \left(w,\, \nabla\cdot\boldsymbol{q}\right)_{\mathcal{T}_h} - \left\langle w,\, \tau\left(\bar\phi - \hat\phi\right)\right\rangle_{\partial\mathcal{T}_h} &= \mathrm{rh}_h(w, \boldsymbol{u}^k, p_h^k, \boldsymbol{F})_K \\ \langle \mu,\, \widehat{\boldsymbol{q}}\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} &= \langle \mu,\, g_N\rangle_{\Gamma_N} \end{aligned} \tag{3.31}$$

where the operator $\mathrm{rh}_h(\cdot)$ contains all the componentwise right-hand side terms of the momentum equation. This formulation is consistent with the full 3D velocity predictor if $\nu_{xy} = \nu_z$. This also couples the domain together on the interfaces. The velocity predictor $\bar{\boldsymbol{u}}_h^{k+1}$ is the vector $(\bar\phi_u,\, \bar\phi_v,\, \bar\phi_w)$ resulting from each of the HDG solves.

In the decoupled case, the formulation of the DG right-hand side operators can be obtained by considering only the relevant component w of the test function $\boldsymbol{v} \in \boldsymbol{V}_h^p$ as appears in equation (3.27). For the advective term, we have

$$a_h(w, \bar\phi, \boldsymbol{u}_h^k, \boldsymbol{g}_D) = -\left(w,\, \nabla\cdot\left(\bar\phi\boldsymbol{u}_h^k\right)\right)_K = -\left(\nabla w,\, \bar\phi\boldsymbol{u}_h^k\right)_K + \left\langle w,\, \left(\bar\phi\boldsymbol{u}_h^k\right)^*\cdot\boldsymbol{n}\right\rangle_{\partial K}, \tag{3.32}$$

The numerical flux in this work $\left(\bar\phi\boldsymbol{u}_h^k\right)^*$ is taken to be an upwind flux. Since $\boldsymbol{u}_h^k$ is multiply-valued, either the average value $\{\{\boldsymbol{u}_h\}\}$ or the HDG flux $\hat{\boldsymbol{u}}_h$ can be used.

Similarly, the pressure gradient term for the component i of $\bar{\boldsymbol{u}}_h^{k+1}$, $\bar\phi_i$, is

$$\mathrm{pg}_h\left(w, p_h^k\right) = -\left(\frac{\partial w}{\partial x_i},\, p_h^{\prime,k}\right) + \left\langle w,\, \left(p_h^{\prime,k}\right)^* n_i\right\rangle \tag{3.33}$$

where n_i denotes the i^{th} component of the outward unit normal $\boldsymbol{n}$, and where the numerical flux is again chosen to be the central flux, $\left(p_h^{\prime,k}\right)^* = \left\{\left\{p_h^{\prime,k}\right\}\right\}$.

3.4.4 Free-surface corrector

Define the gradient $\boldsymbol{q}_{\delta\eta} \equiv \nabla\delta\eta$. We seek $(\delta\eta_h^{k+1},\, \boldsymbol{q}_{h,\delta\eta},\, \hat{\delta\eta}_h) \in [(\boldsymbol{V}_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}})]^{d-1}$ such that

$$\begin{aligned}
\left(\boldsymbol{v},\, \frac{\boldsymbol{q}_{h,\delta\eta}}{a\Delta t g\left[\eta^k + H\right]}\right)_{\mathcal{T}_h} + \left(\nabla\cdot\boldsymbol{v},\, \delta\eta_h^{k+1}\right)_{\mathcal{T}_h} - \left\langle \boldsymbol{v}\cdot\boldsymbol{n},\, \hat{\delta\eta}_h\right\rangle_{\partial\mathcal{T}_h} &= 0,\\
\left(w,\, \frac{\delta\eta_h^{k+1}}{a\Delta t}\right)_{\mathcal{T}_h} - \left(w,\, \nabla\cdot\boldsymbol{q}_{h,\delta\eta}\right)_{\mathcal{T}_h} + \left\langle w,\, \tau\left(\delta\eta_h^{k+1} - \hat{\delta\eta}_h\right)\right\rangle_{\partial\mathcal{T}_h} &= -d_h(w,\, \bar{\boldsymbol{U}}_h),\\
\left\langle \mu,\, \boldsymbol{q}_{h,\delta\eta}\cdot\boldsymbol{n} + \tau\left(\delta\eta_h^{k+1} - \hat{\delta\eta}_h\right)\right\rangle_{\partial\mathcal{T}_h} &= \langle \mu,\, g_N\rangle_{\partial\mathcal{T}_h}
\end{aligned} \tag{3.34}$$

for all $(\boldsymbol{v},\, w,\, \mu) \in [(\boldsymbol{V}_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}})]^{d-1}$. The notation $[\,\cdot\,]^{d-1}$ refers to the fact that these finite element spaces are defined on the surface mesh $\partial\Omega_\eta$ and are of different dimension than the other finite element problems specified in this section. The quantity $\bar{\boldsymbol{U}}_h \equiv \int_{-H}^{\eta^k} \bar{\boldsymbol{u}}_{xy,h}^{k+1}\, dz$ is the depth-integrated horizontal velocity, and the operator $d_h(w, , \bar{\boldsymbol{U}})$ is

$$d_h(w,\, \bar{\boldsymbol{U}}) = \left(w,\, \nabla\cdot\bar{\boldsymbol{U}}\right)_K = -\left(\nabla w,\, \bar{\boldsymbol{U}}\right)_K + \left\langle w,\, \left(\bar{\boldsymbol{U}}\right)^*\cdot\boldsymbol{n}\right\rangle_{\partial K} \tag{3.35}$$

For the numerical flux to on the edges, we use a central flux for the divergence operators: $\left(\bar{\boldsymbol{U}}\right)^* = \left\{\!\left\{\bar{\boldsymbol{U}}\right\}\!\right\}$ to maintain discrete consistency with both the discrete divergence on the right-hand side of the weak pressure corrector equation in §3.4.6 and the discrete pressure gradient in equation (3.30) as investigated in [77, 78], where it was found that this consistency, as well as an integration by parts formulation, is necessary for the stability of the scheme.

3.4.5 Intermediate velocity projection, free-surface update

From the mixed formulation of the free-surface corrector equation, we have that $\boldsymbol{q}_{\delta\eta} = \kappa_{\delta\eta}\nabla\delta\eta = a\Delta t g(H+\eta)\nabla\delta\eta$, which, combined with the intermediate update, yields the horizontal velocity correction and free-surface correction

$$\bar{\bar{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - \frac{1}{(\eta^k + H)}\boldsymbol{q}_{\delta\eta,h} \tag{3.36}$$

$$\eta^{k+1} = \eta^k + \delta\eta^{k+1} \tag{3.37}$$

3.4.6 Pressure corrector

Define the gradient of the pressure corrector $\boldsymbol{q}_{\delta p'}$. The weak form of equation (3.17) for the correction to the pressure $\delta p'$ is as follows: seek $(\boldsymbol{q}_{\delta p'},\, \delta p',\, \hat{\delta p'}) \in (\boldsymbol{V}_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}})$ such that

$$\begin{aligned}
\left(\boldsymbol{v},\, \boldsymbol{q}_{\delta p'}^{k+1}\right)_{\mathcal{T}_h} + \left(\nabla\cdot\boldsymbol{v},\, \delta p'^{,k+1}\right)_{\mathcal{T}_h} - \left\langle \boldsymbol{v}\cdot\boldsymbol{n},\, \widehat{\delta p'}\right\rangle_{\partial\mathcal{T}_h} &= 0,\\
-\left(w, \nabla\cdot\boldsymbol{q}_{\delta p'}^{k+1}\right)_{\mathcal{T}_h} + \left\langle w,\, \tau_p\, \delta p'^{,k+1}\right\rangle_{\partial\mathcal{T}_h} - \left\langle w,\, \tau_p \widehat{\delta p'}\right\rangle_{\partial\mathcal{T}_h} &= \frac{1}{a\Delta t}\left[-\left(\nabla w,\, \bar{\bar{\boldsymbol{u}}}^{k+1}\right)_{\mathcal{T}_h} + \left\langle w,\, \left(\bar{\bar{\boldsymbol{u}}}^{k+1}\right)^*\cdot\boldsymbol{n}\right\rangle_{\partial\mathcal{T}_h}\right],\\
\left\langle \mu,\, \boldsymbol{q}_{\delta p'}^{k+1}\cdot\boldsymbol{n} + \tau_p\, \left(\delta p'^{,k+1} - \hat{\delta p'}\right)\right\rangle_{\delta\mathcal{T}_h\setminus\Gamma_d} &= 0
\end{aligned} \tag{3.38}$$

for all $(\boldsymbol{v},\, w,\, \mu) \in (\boldsymbol{V}_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}})$, where $\left(\bar{\bar{\boldsymbol{u}}}^{k+1}\right)^*$ is a numerical flux which we choose to be a central flux $\left(\bar{\bar{\boldsymbol{u}}}^{k+1}\right)^* = \left\{\!\left\{\bar{\bar{\boldsymbol{u}}}^{k+1}\right\}\!\right\}$ in order to maintain consistency with the discrete divergences in the free-surface corrector (3.34) equation and the discrete pressure gradient in the momentum equation (3.33) [77].

3.4.7 Velocity projection, pressure update

$$\boldsymbol{u}^{k+1} = \bar{\bar{\boldsymbol{u}}}^{k+1} - a\Delta t \boldsymbol{q}_{\delta p'} \tag{3.39}$$

$$p'^{,k+1} = p'^{,k} + \delta p'^{,k+1}. \tag{3.40}$$

A feature of using a so-called "mixed method" such as HDG or the local discontinuous Galerkin (LDG) method for the spatial discretization of a second-order PDE is that the gradients of all unknowns are solved for simultaneously along with their primal counterparts. As in §3.4.5, we can perform the update by making use of the gradient unknowns directly to perform the projection in (3.20), which we will refer to as a "strong" projection update as in (3.39). However, we can also perform the projection updates weakly using, in this case, the discrete pressure gradient $\mathrm{pg}_h(\boldsymbol{v}, p_h^k)$ (3.30), leading to the following variational problem. We seek $\boldsymbol{u}^{k+1} \in \boldsymbol{V}_h^{p_{\text{order}}}$ such that

$$\left(\boldsymbol{v},\, \boldsymbol{u}_h^{k+1}\right)_{\mathcal{T}_h} = \left(\boldsymbol{v},\, \bar{\bar{\boldsymbol{u}}}_h^{k+1}\right)_{\mathcal{T}_h} - a\Delta t \left[\left(\nabla\cdot\boldsymbol{v},\, p_h'^{,k+1}\right)_{\mathcal{T}_h} + \left\langle \boldsymbol{v},\, \left\{\left\{ p_h'^{,k+1} \right\}\right\} \boldsymbol{n} \right\rangle_{\partial\mathcal{T}_h} \right] \tag{3.41}$$

for all $\boldsymbol{v} \in \boldsymbol{V}_h^{p_{\text{order}}}$. The variational problem does not require the solution of a linear system, as $p'^{,k+1}$ is available and the space $\boldsymbol{V}_h^{p_{\text{order}}}$ has no shared degrees of freedom between elements. Therefore, this weak update can be performed as a reconstruction on each element independently. This not only provides increased stability, as the hydrostatic pressure gradient is coupled between elements by virtue of the numerical flux, but also allows for the addition of penalty terms to provide stabilization in the case of modeling under-resolved turbulent flows: for details, see [78]. Addition of penalization terms in this manner can prevent growth of the discrete divergence and numerical energy over time, dampening spurious oscillations due to under-resolution and providing an alternative to slope limiting.

3.4.8 Density perturbation update

For the HDG spatial discretization of (3.21), it is possible to treat the tracer advection implicitly in a spatial sense as in [191], which allows the stabilization parameter τ to be chosen according to the physical relationship between diffusion and advection in the problem of interest. However, doing so comes at the cost of the symmetry of the variational form and hence that of the linear system representing its discretization, requiring the use of iterative solvers for non-symmetric linear systems. Therefore, in this work we treat the advection term in (3.21) explicitly in an HDG sense, preserving problem symmetry. Serendipitously, this choice also results in the same weak form as (3.31) in §3.4.3.2, if we take $\bar{\phi} = {\rho'}^{k+1}$ and modify the right-hand side to include only the weak advection term $a_h(w, {\rho'}_h^{k+1}, \boldsymbol{u}_h^{k+1}, g_{\rho'})$, which, for clarity, we write here

$$\begin{aligned} a_h(w, {\rho'}_h^{k+1}, \boldsymbol{u}_h^{k+1}, g_{\rho'}) &= -\left(w,\, \nabla\cdot\left({\rho'}_h^{k+1}\boldsymbol{u}_h^{k+1}\right)\right)_K \\ &= -\left(\nabla w,\, {\rho'}_h^{k+1}\boldsymbol{u}_h^{k+1}\right)_K + \left\langle w,\, \left({\rho'}_h^{k+1}\boldsymbol{u}_h^{k+1}\right)^* \cdot \boldsymbol{n}\right\rangle_{\partial K}. \end{aligned}$$

We use an upwind flux according to the average value of $\boldsymbol{u}_h^{k+1}$ on each edge.

3.4.9 Recovery of the hydrostatic vertical velocity

For the recovery of the vertical velocity in the 2D case, we can rewrite the continuity equation as

$$\nabla\cdot(w\hat{\boldsymbol{z}}) = -\nabla\cdot(u\hat{\boldsymbol{x}}),$$

the weak DG form is then (for a test function q) is

$$-(\nabla q,\, w^{k+1}\hat{\boldsymbol{z}})_K + \left\langle q,\, \left(w^{k+1}\right)^* \hat{\boldsymbol{z}}\cdot\boldsymbol{n}\right\rangle_{\partial K} = -(q,\, \nabla\cdot(u^{k+1}\hat{\boldsymbol{x}}))_K,$$

where the right-hand side is data, since u^{k+1} is known. Leaving the right-hand side as is, without integrating by parts is in some sense not a principled approach, as the horizontal velocities are completely decoupled column-by-column. However, comparing the discretization in equation (3.4.9) to the form of the generic vertical integration in equation (3.24), we have that the recovery of the vertical velocity is equivalent to a discrete depth integration operation. Namely,

$$w^{k+1} = \int_z^{\eta^{k+1}} \frac{\partial w^{k+1}}{\partial z'}\, dz' = \int_z^{\eta^{k+1}} -\frac{\partial u^{k+1}}{\partial x}\, dz',$$

where the first equality follows from the fundamental theorem of calculus, and the second from the continuity equation. Therefore, applying the discrete derivative $\nabla \cdot (u^{k+1}\hat{\boldsymbol{x}})$ and discretely depth integrating the resulting field is a convenient way to perform the recovery.

A more consistent approach would be to perform the integration by parts for the right-hand side, resulting in

$$-(\nabla q,\, w^{k+1}\hat{\boldsymbol{z}})_K + \left\langle q,\, \left(w^{k+1}\right)^* \hat{\boldsymbol{z}} \cdot \boldsymbol{n} \right\rangle_{\partial K} = (\nabla q,\, u^{k+1}\hat{\boldsymbol{x}})_K - \left\langle q,\, \left(u^{k+1}\right)^* \hat{\boldsymbol{x}} \cdot \boldsymbol{n} \right\rangle_{\partial K}$$

where we emphasize that the right-hand side is still data, however, the numerical flux $(u^{k+1})^*$ adds horizontal coupling between elements. A reasonable choice that preserves consistency is an average flux $(u^{k+1})^* = \left\{\!\left\{u^{k+1}\right\}\!\right\}$. If, similar to the depth-integration weak form, we take the directional numerical flux in the negative $\hat{\boldsymbol{z}}$ direction, $(w^{k+1})^* = \lim_{\epsilon\to 0} w(\boldsymbol{x}, z+\epsilon)$ at every interface, which amounts to taking the value of w^{k+1} approaching the interface from the top. Using these numerical flux choices, we can write the weak form as

$$\begin{aligned} -(\nabla q,\, w^{k+1}\hat{\boldsymbol{z}})_{\mathcal{T}_h} &+ \left\langle [\![q]\!],\, \left(w^{k+1}\right)^* \hat{\boldsymbol{z}} \cdot \boldsymbol{n} \right\rangle_{F_h^{i,\mathrm{nv}}} \\ &= (\nabla q,\, u^{k+1}\hat{\boldsymbol{x}})_{\mathcal{T}_h} - \left\langle [\![q]\!], \left(u^{k+1}\right)^* \hat{\boldsymbol{x}} \cdot \boldsymbol{n} \right\rangle_{F_h^i} - \left\langle q, u^{k+1}\hat{\boldsymbol{x}} \cdot \boldsymbol{n} \right\rangle_{F_h^\Gamma} \end{aligned}$$

where F_h^i refers to the set of inter-element interfaces, and $F_h^{i,\mathrm{nv}}$ refer to the non-vertical interfaces between elements. For simplicity, we show the derivation for a flat-bottomed case, but incorporation of a non-trivial bottom follows directly from the application of the Leibniz integral rule to (3.4.9) the integrated continuity equation at the topography.

3.5 Numerical experiments

In this section, we present numerical results from a set of test cases that demonstrate the convergence and accuracy of the proposed model. The test cases employed here are: (1) a free-surface seiche and (2) a continuously-stratified internal seiche, (3) the formation of internal solitary waves [252].

3.5.1 Implementation

Unless otherwise stated, we take the background density $\rho_0 = 1024\text{kg/m}^3$. Volume and surface integrals that appear in the weak forms above are calculated using Gaussian quadrature. Namely, we use $n_q = p_u + 1$ quadrature points to integrate the velocity mass-matrix inertial terms, the advection and diffusion terms, the velocity divergence terms, as well as the body forcing term. We use $n_q = p_p + 1$ quadrature points to integrate the Laplace operator occurring in the pressure correction equation. Aliasing effects arising due to the nonlinearity of the advection term are often treated by over-integration using $n_q = \lfloor 3p_u/2 \rfloor$ quadrature points for both the volume and surface integrals related to the advection term. In the subsequent test cases, we have found this over-integration to be unnecessary. We use $n_q = p_u + 1$ quadrature points to integrate all dimension $d - 1$ integrals arising as a result of the discretization of the free-surface, and all depth-integrations as detailed in §3.4.2 of the non-hydrostatic density perturbation are computed with $n_q = p_u + 4$ quadrature to limit depth integration error.

The linear systems of equations that arise as a result of the spatial discretization are solved using Krylov subspace iterative methods—in particular, the conjugate gradient (CG) method and the biconjugate gradient stabilized method (BiCGSTAB). Unless otherwise stated, the linear systems are solved without preconditioning and to a relative tolerance of 10^{-11}.

Our code is implemented in `C++` and makes use of the finite-element library `deal.II` [10, 9] as well as the scientific computing libraries `NumPy` and `SciPy`. We use the template meta-programming paradigm to write performant, dimension-independent code.

3.5.1.1 CFL condition

Due to the explicit time integration of the advection operator, the time step size Δt is restricted according to the Courant–Friedrichs–Lewy (CFL) condition

$$\Delta t < \frac{h_{\min} \, \mathrm{Cr}}{p_{\mathrm{order},u}^{1.5} \, \|\boldsymbol{u}_{\max}\|},$$

where $h_{\min}$ is the minimum representative element length scale, Cr is the empirically determined Courant number, and $p_{\mathrm{order},u}$ is the polynomial order of the discrete velocity space. This is a global bound on the local condition that the resolution must be such that the physical domain of dependence is captured by the discretization.

The effective spatial discretization length scale $h_{\min}/p_{\mathrm{order},u}^{1.5}$ suggests the dependence of the time step restriction on the polynomial degree of the finite element space. The exponent value of 1.5 was experimentally found to allow a constant Courant number over a wide range of polynomial degrees for simulations of under-resolved turbulent flows in [78]. Additional discussion can be found in [110, 93].

3.5.2 Spatial convergence: verification

The discrete derivative operators in DG-FEM schemes require care to ensure high-order convergence; incorrect implementation of any term can prevent convergence to the true solution at the optimal order [110]. Inspired by the method of manufactured solutions, to provide verification of the spatial derivative operators and numerical fluxes in the momentum equation (3.1) and its discrete equivalent (3.31), we choose the quantity $\Upsilon = (\boldsymbol{u}, p', \eta, \rho')$ and analytically deduce the resultant forcing term $\boldsymbol{f}$ which balances equation (3.42)

$$\frac{\boldsymbol{\theta}_h}{a\Delta t} - \nabla \cdot (\nu \nabla \boldsymbol{\theta}_h) = -\nabla p' - \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) + g \nabla_{xy} \eta - \frac{1}{\rho_0} \int_z^0 g \nabla_{xy} \rho' \, dz' + \frac{\boldsymbol{u}}{a\Delta t} + \frac{1}{\rho_0} \boldsymbol{f}, \tag{3.42}$$

and solve the discretized PDE for $\boldsymbol{\theta}_h$, allowing measurement of the rate of convergence to the analytical solution $\boldsymbol{u}$. We take as manufactured solution

$$\Upsilon = \Big((\sin(\pi x_2) \sin(\pi x_1), \cos(2\pi x_2)), \ \cos(2\pi x_2) \sinh(x_1), \ 1 - e^{-x_1^2}, \ \sin^2(\pi y) \Big) \tag{3.43}$$

on the domain $\Omega = [-1, 1]^2$ with mixed boundary conditions $\Gamma_D = \partial\Omega \cap \{\boldsymbol{x} \colon x_1 < x_2\}$ and $\Gamma_N = \partial\Omega \setminus \Gamma_D$ with values deduced from the analytical solution and domain geometry.

Results of the spatial convergence test are shown in Figure 3-3 for the primal unknown u_h at polynomial orders in the range $p_{\mathrm{order}} = 1, \ldots, 6$. For both the primal unknown u_h and gradient $\boldsymbol{q}_h$ (not shown), optimal $p_{\mathrm{order}} + 1$ convergence rates in the L^2-error norm are observed at all polynomial degrees, verifying the spatial discretization and its implementation. We use the same procedure to similarly verify the spatial discretizations in equations (3.34), (3.38), and (3.24).

3.5.3 Validation: linear gravity waves

We verify the nonhydrostatic model using results from gravity-wave theory. We model both liquid-surface gravity waves occurring in a free-surface seiche §3.5.3.1 and internal wave behavior occurring

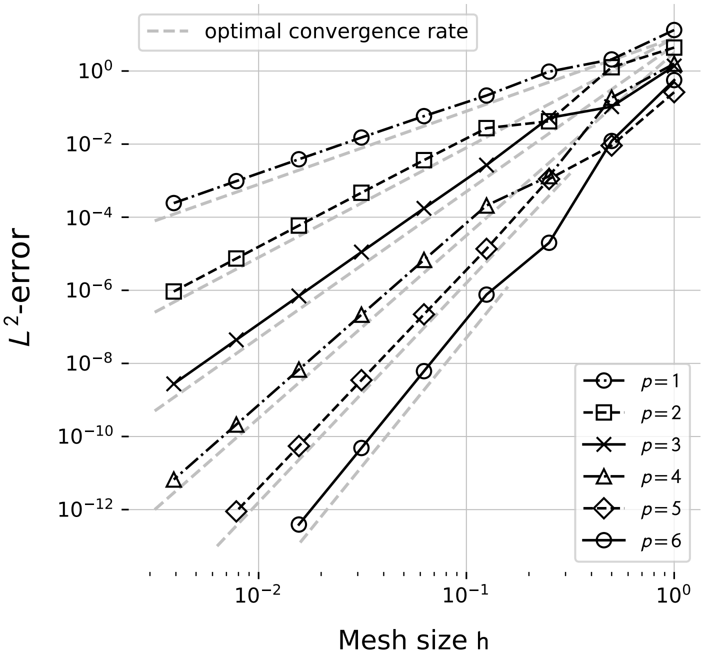


Figure 3-3: Convergence study: spatial errors in the primal unknown $\|u - u_h\|_{L^2(\Omega)}$ for the test case in equation (3.43) at different polynomial orders p, compared against optimal $p+1$ rates of convergence.

	top	bottom	sides
$\bar{u}$	Γ_N	Γ_N	Γ_D
$\bar{w}$	Γ_N	Γ_D	Γ_N
$\delta\eta$	-	-	Γ_N
$\delta p'$	Γ_D	Γ_N	Γ_N
ρ'	Γ_N	Γ_N	Γ_N

Table 3.1: Boundary condition specifications for the gravity wave test cases. All boundary conditions are homogeneous.

in a continuously-stratified internal seiche §3.5.3.2.

For each test case, we consider the enclosed, flat-bottomed domain $\Omega = [0, L] \times [-H, 0]$ of length L and depth H shown in Fig. 3-4. Both the free-surface and continuously-stratified seiche exhibit the formation of a standing wave at the free-surface and the density interface, respectively. Each standing wave is characterized by a fundamental wavelength $\lambda_w = 2L$ with corresponding wave number $k = 2\pi/\lambda_w$, as well as an initial amplitude a and frequency ω. The degree of nonhydrostasy is described by the domain aspect ratio L/H; the hydrostatic shallow-water limit is obtained as $L/H \to \infty$ and the nonhydrostatic deep-water limit is obtained as $L/H \to 0$.

For both the free-surface waves and internal waves (as measured by the center of the pycnocline), we compare the numerical dispersion behavior to the dispersion relations predicted by linear gravity wave theory [138, 252]. We refer to the theoretical nonhydrostatic and hydrostatic wave speeds as c_{NH} and c_H, respectively. The wave speeds predicted by the model are determined by measuring the numerical period of oscillation T_h, defined as the elapsed time between the initial condition and the subsequent peak wave amplitude at the $x = 0$ domain boundary. The corresponding numerical wave speed is then computed using the general periodic relation $c = \omega/k = 2\pi/(kT)$.

We take care to choose model parameters and initial conditions such that the linear wave theory to which we compare our model is valid; namely, that the initial wave amplitudes are small in relation to the domain depth ($a \ll H$) and that the numerical viscosity is small in order to model inviscid flow. The numerical experiments are run without Coriolis forcing, $F_{\text{cor}} = 0$.

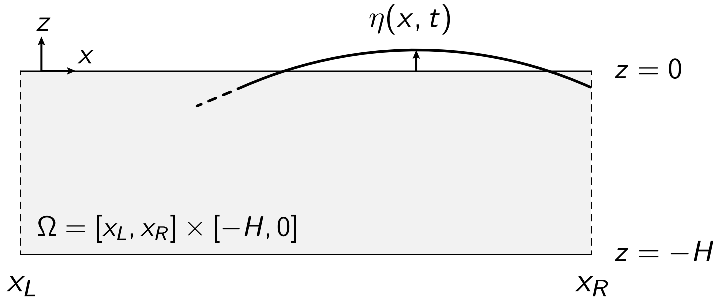


Figure 3-4: Computational domain for the nonhydrostatic gravity wave test cases.

The boundary condition types for each partial differential equation for the projection scheme detailed in §3.3 are given in Table 3.1, and are the same for the free-surface seiche and internal seiche test cases. All boundary condition values are homogeneous, and, taken together, physically represent a free-slip condition for the tangential velocity on the sides and bottom of the computational domain. The zero-gradient boundary condition on the top of the computational domain allows for communication between the one-dimensional free-surface grid and the domain-interior velocity field.

All gravity wave simulations are discretized on a numerical grid consisting of polynomial order $p = 3$ quadrilateral elements, $N_x = 15$ and $N_z = 15$ elements in the horizontal and vertical, respectively. We remark that this is an exceptionally coarse mesh and is meant to illustrate the capability of high-order models.

3.5.3.1 Free-surface seiche

In this test case, the fluid density is constant and the free-surface is initially perturbed from equilibrium according to the initial wave $\eta(x, t = 0) = a\cos(kx)$ with amplitude a. The resulting flow is driven by the action of gravity on the initial surface wave.

The analytical behavior of the free-surface is given by $\eta = a\cos(kx)\cos(\omega t)$, with the corresponding analytical solutions for the horizontal and vertical velocities

$$u = ag\frac{k}{\omega}\frac{\cosh k(z+H)}{\cosh(kH)}\sin(kx)\sin(\omega t), \qquad v = -ag\frac{k}{\omega}\frac{\sinh k(z+H)}{\cosh(kH)}\cos(kx)\sin(\omega t),$$

approach the hydrostatic solution

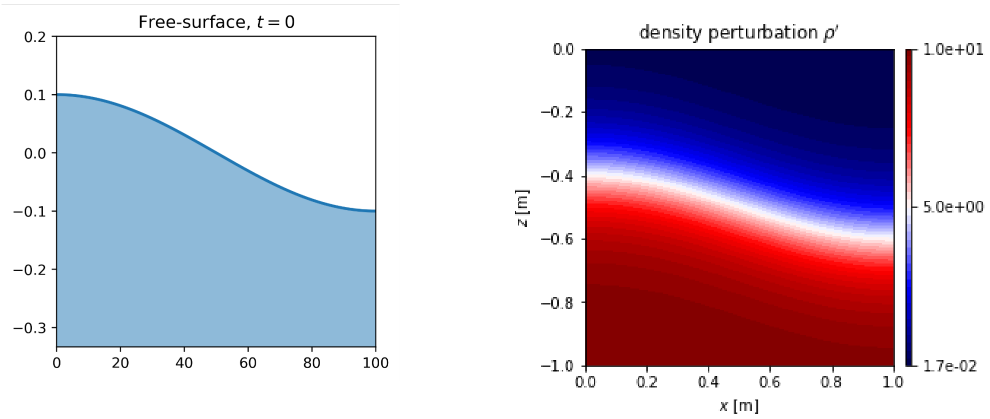
$$u = ag\frac{k}{\omega}\sin(kx)\sin(\omega t), \qquad v = -ag\frac{k}{\omega}\cos(kx)\sin(\omega t),$$

in the shallow-water limit. With constant density, the analytical wave speed of a free-surface wave is given by

$$c = \sqrt{\frac{g}{k}\tanh(kH)}, \tag{3.44}$$

resulting in wave speeds of $c_{\text{hs}} = \sqrt{gH}$ in the hydrostatic limit and $c_{\text{nhs}} = \sqrt{g/k}$ in the nonhydrostatic limit.

We initialize the numerical model with the free-surface perturbation $\eta(t = 0) = a\cos(kx)$, with seiche amplitude $a = 0.01$ m, on a domain with depth $H = 1$. To mimic the inviscid assumption in linear gravity wave theory, we run our model with a kinematic diffusivity of $\nu_x = \nu_z = 10^{-8}$ Pa·s.



(a) Initial conditions for the free-surface seiche (density ρ' constant). To illustrate the shape of the free surface, only the top of the domain is shown.

(b) Initial conditions for density perturbation ρ' for the continuously stratified internal seiche test case in §3.5.3.2 (free-surface $\eta(x,\, t_0) = 0$.

Figure 3-5: Initial conditions for the gravity wave test cases.

The background density is taken to be $\rho_0 = 1000$ kg m^{-3} without density perturbation ($\rho'(\boldsymbol{x}, t) = 0$). For the temporal discretization, we use a simple first-order backward Euler scheme with a constant time step $\Delta t = 0.005$. We run the model for two seiche periods, $t_{\max} = 2T$, where the period T is computed from the analytical solution.

In Fig. 3-6, we compare the normalized velocity profiles of the numerical model along the vertical slice $x = L/10$ to that of the theoretical solution at time $t = T/4$ on a domain with equal aspect ratio, $H/L = 1$ so that the analytical behavior is strongly nonhydrostatic. Both the horizontal and vertical velocity profiles show excellent agreement with the theoretical solutions over the domain. As expected, the most significant physical difference between the hydrostatic and nonhydrostatic velocity profiles is the depth-variation of the horizontal velocity; the hydrostatic horizontal velocity profile does not vary with depth, in contrast to the nonhydrostatic profile, which decays rapidly from its maximum value at the free-surface.*

Fig. 3-6 depicts the same comparison of normalized velocity profiles, but on a domain with depth $H = 1$ and length $L = 9.8$ m. Once again, the numerical solution shows agreement with the nonhydrostatic solution, but the differences between the hydrostatic and nonhydrostatic velocity profiles are less apparent, as, physically, the numerical experiment begins to approach the shallow-water limit.

To further verify the model, we vary the degree of nonhydrostasy by fixing the domain height at $H = 1$ m and changing the length L of the domain. Figure 3-7 compares the numerical wave speed to that of the analytical wave speed predicted by hydrostatic and nonhydrostatic theory as a function of the domain aspect ratio. The numerical predictions agree with the relationships given in equation (3.44), which indicates that the model achieves the correct dispersive behavior. As the domain aspect ratio L/H shrinks, nonhydrostatic effects become important and the wave speed deviates from the hydrostatic solution, which drastically over-predicts the true wave speed. However, as the domain length L increases relative to the depth H, the hydrostatic approximation becomes valid and the nonhydrostatic and hydrostatic solutions are similar, which is also captured by the model. On the whole, these diverse test cases provide extensive verification for the numerical solver. They encompass a broad spectrum of both hydrostatic and nonhydrostatic physics, which provides a comprehensive overview of the solver's capabilities and performance.

All numerical simulations show excellent agreement with the nonhydrostatic theory, as well as convergence to the expected hydrostatic behavior in the shallow-water limit.

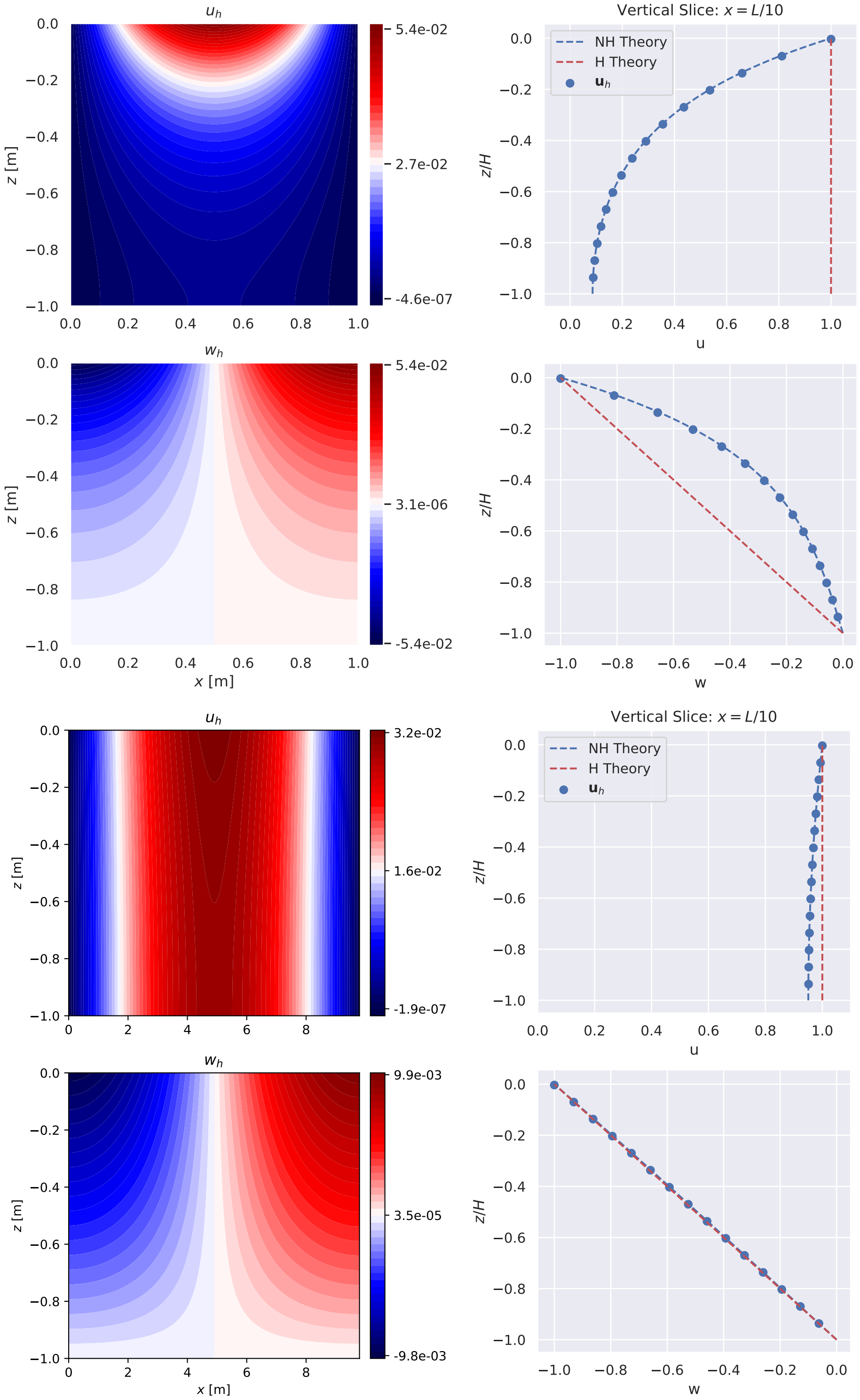


Figure 3-6: Numerical velocities $\boldsymbol{u}_h$ and their comparison to nonhydrostatic and hydrostatic theory along a vertical slice located at $x = L/10$ at time $t = T/4$ for different simulations with domains $\Omega = (0, L) \times (-1, 0)$ with domain lengths of $L = 1$m (left) and $L = 9.8$m (right).

3.5.3.2 Continuously-stratified internal seiche

In this case, we compute the oscillations of a nonhydrostatic internal seiche to highlight fundamental aspects of nonhydrostatic physics that must be captured in simulations of internal waves. This test case illustrates the coupling of the density perturbation ρ' as well as the free-surface to the nonhydrostatic pressure.

The initial density perturbation is given by

$$\rho' = \frac{\Delta\rho}{2}\left[1 - \tanh\left(\frac{2\tanh^{-1}\alpha_s}{\delta_\rho}(z + H/2 - \xi)\right)\right]$$

where δ_ρ represents the pyclocline thickness, $\alpha_s = 0.99$ [252], and $\Delta\rho$ represents the top-bottom density difference. The interface is characterized by an initial wave $\xi = a_i\cos(kx)$ with amplitude a_i.

The internal wave for a continuously-stratified internal seiche with interface thickness δ_ρ obeys a modified dispersion relation [252, 138]

$$c = \sqrt{\frac{g'}{2k}\tanh\left(\frac{kH}{2}\right)f_i(k\delta\rho)}$$

where $f_i(k\delta_\rho)$ accounts for effects due to the interface thickness; from (Thorpe, 1968), this function is given to first order as $f_i(k\delta_\rho) = (1 + k\delta_\rho/2)^{-1}$. The initial configuration is shown in Fig. 3-5b, and the free surface is initialized as $\eta(x, t = 0) = 0$.

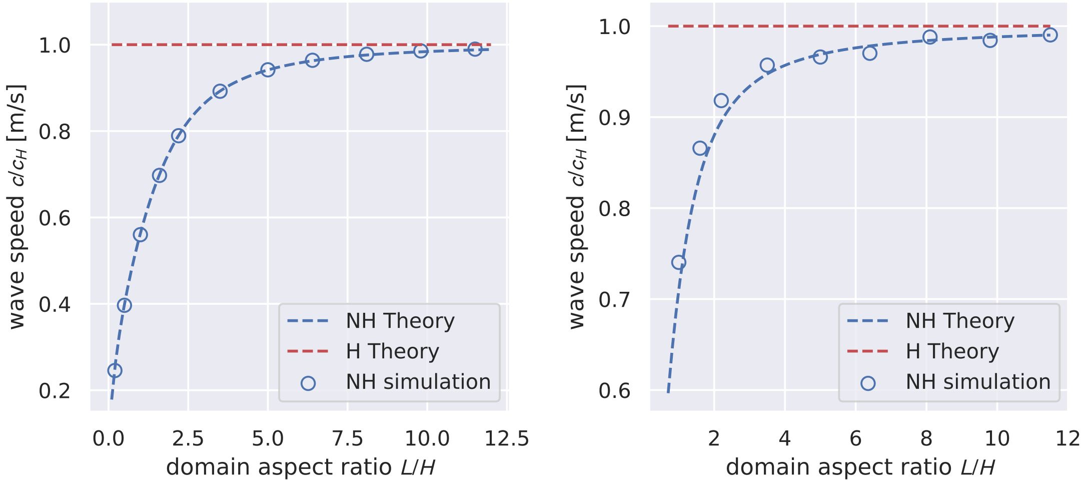


Figure 3-7: The normalized numerical free-surface wave speed (left) and internal wave speed (right) for different nonhydrostatic free-surface seiche simulations run on domains with varying aspect ratio L/H, as compared to the analytical predictions of free-surface wave speed given by nonhydrostatic and hydrostatic theory.

As above, we vary the degree of nonhydrostasy by fixing the domain height at $H = 1$ m and changing the length L of the domain. Figure 3-7 compares the numerical wave speed to that of the analytical wave speed predicted by hydrostatic and nonhydrostatic theory as a function of the domain aspect ratio. The numerical predictions agree with the relationships given in equation (3.44), which indicates that the model achieves the correct dispersive behavior. As the domain aspect ratio L/H shrinks, nonhydrostatic effects become important and the wave speed deviates from the hydrostatic solution, which drastically over-predicts the true wave speed. However, as the domain length L increases relative to the depth H, the hydrostatic approximation becomes valid and the nonhydrostatic and hydrostatic solutions are similar, which is also captured by the model.

3.5.4 Idealized formation of Rayleigh-Taylor instabilities

As we are interested in nonhydrostatic mixed-layer instabilities, we consider an idealized two-dimensional test where heavier water is advected over lighter water by wind forcing, leading to instabilities. One specific type of gravity-driven instability seen at interfaces between fluids of different densities are referred to as Rayleigh-Taylor (RT) instabilities. These instabilities are observed in processes that span a wide range of length scales ranging from supernova explosions [219] to turbulent mixing [27, 263]. RT instabilities have garnered a lot of attention over the years and there have been many investigations into the linear stability analysis [137, 265], nonlinear [65] stability analysis and RT instabilities in the context of ocean dynamics [57, 60, 198, 248]. The instabilities can be explained in terms of a out-of-the-plane vorticity Specifically, in an ocean context, this is commonly referred to as the creation of a baroclinic torque.

These gravity driven instabilities represent an idealized analogue to the results seen in §3.6.3 and §3.6.4, where at various points in the model nested runs in the Alboran sea, heavier water is advected over lighter water, resulting in the formation of mixed-layer instabilities. While the instabilities occurring in the mixed layer are much more complicated than these idealized test cases [172, 174], these simulations can both be used in parameter studies, as well as a benchmark for hydrostatic-nonhydrostatic multi-model implementations [228]. In this case we consider a domain of $\Omega = [0, 30\text{km}] \times [-50\text{m}, 0]$, simulating a 2D cross-sectional slice of the mixed layer. Figure 3-8 illustrates the setup of the domain, including a to-scale comparison. The boundary conditions for the nonhydrostatic HDG model are given in Table 3.2.

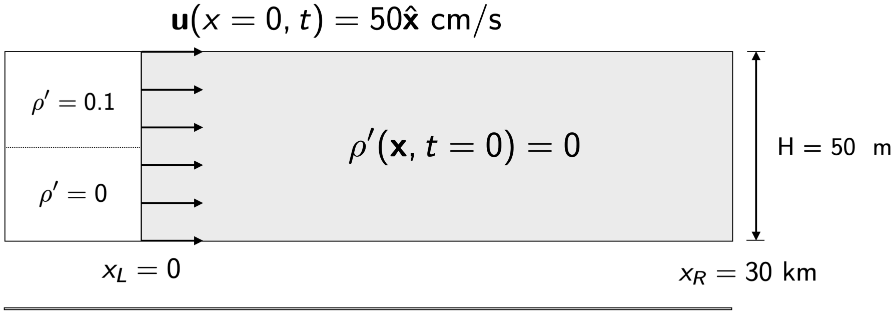

$\mathbf{u}(x=0,t) = 50\hat{\mathbf{x}}$ cm/s
$\rho' = 0.1$
$\rho' = 0$
$\rho'(\mathbf{x}, t=0) = 0$
H = 50 m
$x_L = 0$
$x_R = 30$ km

Figure 3-8: Domain setup for idealized RTI simulations conducted in §3.5.4.The schematic is not to-scale; a to-scale representation of the domain is shown directly underneath.

	top	bottom	inflow	ouflow
$\bar{u}$	Γ_N	Γ_N	Γ_D	Γ_N
$\bar{w}$	Γ_N	Γ_D	Γ_D	Γ_N
$\delta\eta$	-	-	Γ_N	Γ_N
$\delta p'$	Γ_D	Γ_N	Γ_N	Γ_N
ρ'	Γ_N	Γ_N	Γ_N	Γ_N

Table 3.2: HDG NHS model boundary condition specifications used for the idealized mixed-layer RTI instability simulations.

3.5.4.1 Verification with a finite-volume model

To perform additional validation of the model described in this paper, we compare the results to that of a legacy finite volume incompressible Navier—Stokes code. Specifically, we use a MATLAB code (the 2.29 finite volume MATLAB framework) for solving the incompressible Navier–Stokes equations with Boussinesq approximation for buoyancy [245]. The solver is based on several projection methods; for a review, we refer the reader to [94, 95, 96]. The run parameters and boundary conditions are chosen to be as close as possible, with only the run resolutions substantially differing. Both models use a time step size of $\Delta t = 100$ seconds and run for a simulation time of 4 days. We use the momentum and tracer diffusivity tensors

$$\nu_{\text{mom}} = \begin{pmatrix} 100 & 0 \\ 0 & 0.008 \end{pmatrix}, \qquad \nu_{\text{tracer}} \begin{pmatrix} 100 & 0 \\ 0 & 0.004 \end{pmatrix},$$

where all units are expressed in m^2/s. Additional run parameters, including resolution and polynomial order are summarized in Table 3.3. We remark that due to the underlying nature of the governing equations, both models are considered nonhydrostatic.

	HDG	FV	units
N_x, N_z	$(25, 10)$	$(540, 25)$	
ν_x	$1 \cdot 10^2$	$1 \cdot 10^2$	m^2/s
ν_z	$8 \cdot 10^{-3}$	$8 \cdot 10^{-3}$	m^2/s
Δt	100	100	s
order	4	2	

Table 3.3: HDG NHS and 2.29 FV model parameters used for the idealized mixed-layer RTI instability simulations.

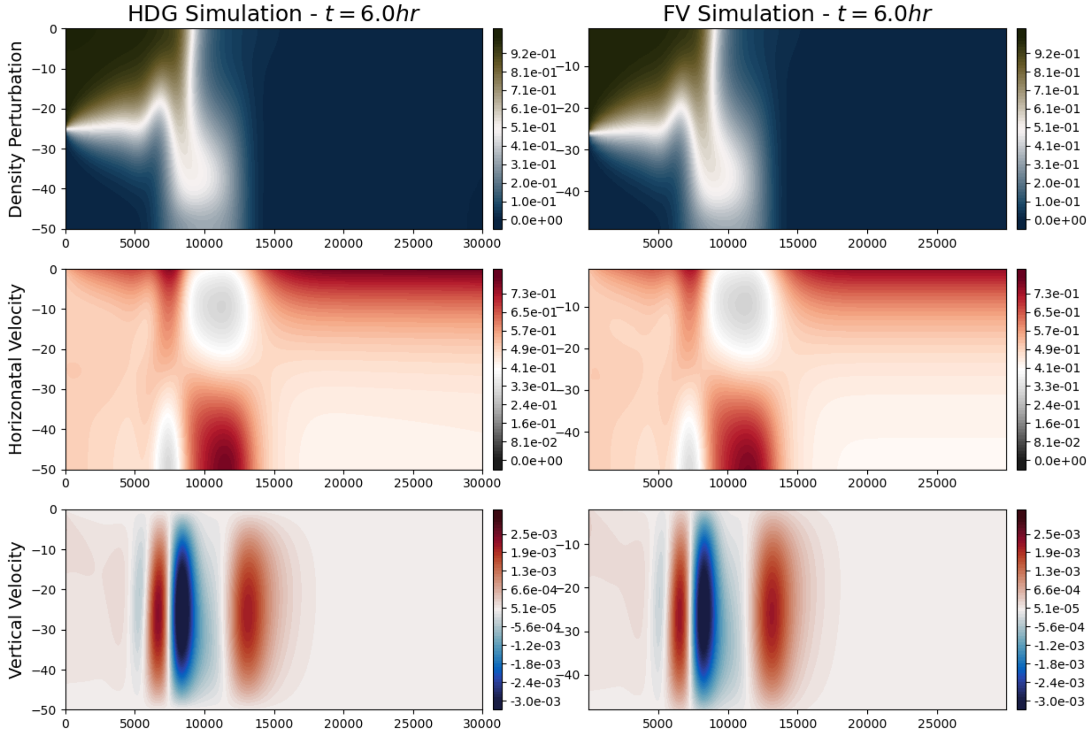


Figure 3-9: Comparison between the high-order nonhydrostatic model (left) and the MSEAS FV229 incompressible Navier—Stokes with Boussinesq approximation.

In our experiment, we operated the high-order nonhydrostatic solver at a significantly coarser resolution compared to that of the finite volume code. Interestingly, despite this considerable difference in resolution, the results produced by the two methods were almost indistinguishable. The results of

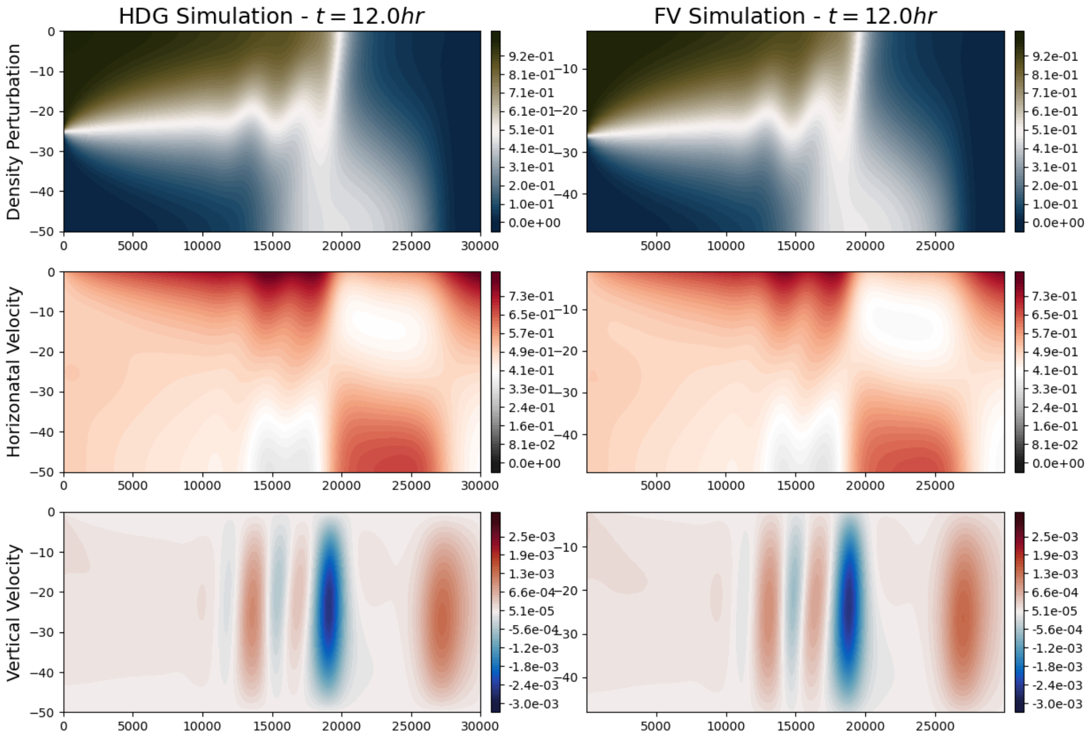


Figure 3-10: Comparison between the high-order nonhydrostatic model (left) and the MSEAS FV229 incompressible Navier—Stokes with Boussinesq approximation.

the two simulations can be seen at time $t = 6$ hours in Figure 3-9 and at time $t = 12$ hours in 3-10, and show excellent agreement between the finite volume model and our nonhydrostatic HDG model. This outcome is promising, given the disparity in resolution levels, and showcases the high-order nonhydrostatic solver's capacity to deliver accurate results even under less refined conditions. This observation lends credence to the assertion that high-order methods often yield better accuracy for the same computational cost as was argued in §1.

In summary, we conducted an investigation of the formation of RTI instabilities, comparing the results of our model against that of a second-order-in-time finite volume model with a different numerical scheme and boundary condition representation. We nonetheless see excellent agreement between the two models, lending credence to the high-order model's spatial and temporal discretizations, and its nonhydrostatic predictive capabilities in the case of mixed layer instabilities.

3.6 Model nesting and initialization from primitive equation data

Hydrostatic models, while incapable of resolving instabilities which are reliant on the vertical momentum equation, such as the Kelvin-Helmholtz billows in the lock exchange problem [101, 243], can capture large-scale features of the flow such as bore speeds and interface thicknesses. In what follows, we corroborate this assertion with numerical experiments conducted by nesting the high-order nonhydrostatic ocean model described in this chapter within the MSEAS-PE hydrostatic model [107]. We extend this approach to real ocean data simulated from the CALYPSO experiment described in §3.6.2.

Some of the algorithmic implementation details of our model nesting approach is specific to MSEAS, but the approach and schemes are general and applicable to other modeling systems.

3.6.1 Notation

To refer to the different field variables present in the different models, we will use the following notation. As elsewhere in the paper, for a generic field variable ϕ we will use bolding to distinguish between scalar-valued and vector-valued quantities as ϕ and $\boldsymbol{\phi}$, respectively. Simulation fields ϕ from the hydrostatic MSEAS PE model will be referred to as ϕ_{PE}. Simulation fields generated from the high-order HDG model will be denoted using ϕ_{HS} and ϕ_{NHS} to refer to output from the hydrostatic and non-hydrostatic models, respectively.

3.6.1.1 Boundary conditions and model initialization

Nesting ocean models governed by different physics and mathematical models, and which use different numerical schemes presents a range of challenges.

Physical inconsistencies and boundary conditions. When one model (the smaller, or "child" model) is nested within another (the larger, or "parent" model), there needs to be a mechanism for passing information between the two at the shared boundary. This exchange can be difficult to manage when the models are based on different sets of equations, as they may have different variables or use the same variables in different ways. This is the case with communication between hydrostatic and nonhydrostatic models; the domain boundaries of the non-hydrostatic model are initialized with no notion of nonhydrostatic pressure p'; however, this is dynamically incorrect. These inconsistencies can cause problems when the models are nested, as it may not be clear how to translate the results of one model into terms the other model can understand. Inaccurate or poorly managed boundary conditions can lead to artifacts or instabilities in the model outputs.

Resolution mismatch. If the models have very different resolutions, it can be challenging to accurately communicate information between them. The finer-scale model might capture small-scale processes that the coarser model misses, but these can be difficult to incorporate into the coarser model in a meaningful way. Conversely, the coarser model may smooth over or average out features that are important at the finer scale.

Discontinuous Galerkin methods are an excellent candidate method to address these difficulties. Their flexibility stems from the fact that they allow for discontinuities at element boundaries. On one hand, the discontinuous representation of the solution is well-suited complex geometries and different types of boundary conditions, making them ideal for multi-model communications. On the other hand, the discontinuous nature of the polynomial spaces provides an inherent stabilizing effect, where discontinuities between elements allow for steep gradients in the numerical solution. Informally speaking, the jumps in the numerical solution act as dampening that stabilizes the DG method when the problem physics can not be well approximated [39], a feature which affords DG-FEM methods increased stability without degrading the accuracy of the solution [39, 45]. DG methods are known for their stability properties, even in the case of complex or irregular domains. This stability makes them a reliable choice for model coupling where instability could lead to significant errors or numerical artifacts. Together, these advantages have made DG-FEM methods exceptionally popular for modeling shocks and crack propagation, and render DG methods an attractive option for model nesting. The discontinuous approximation spaces allow for seamless transfer of data between different resolutions, and the stabilization properties allow for robust and accurate modeling of physics at the boundaries between two different models, addressing both difficulties above.

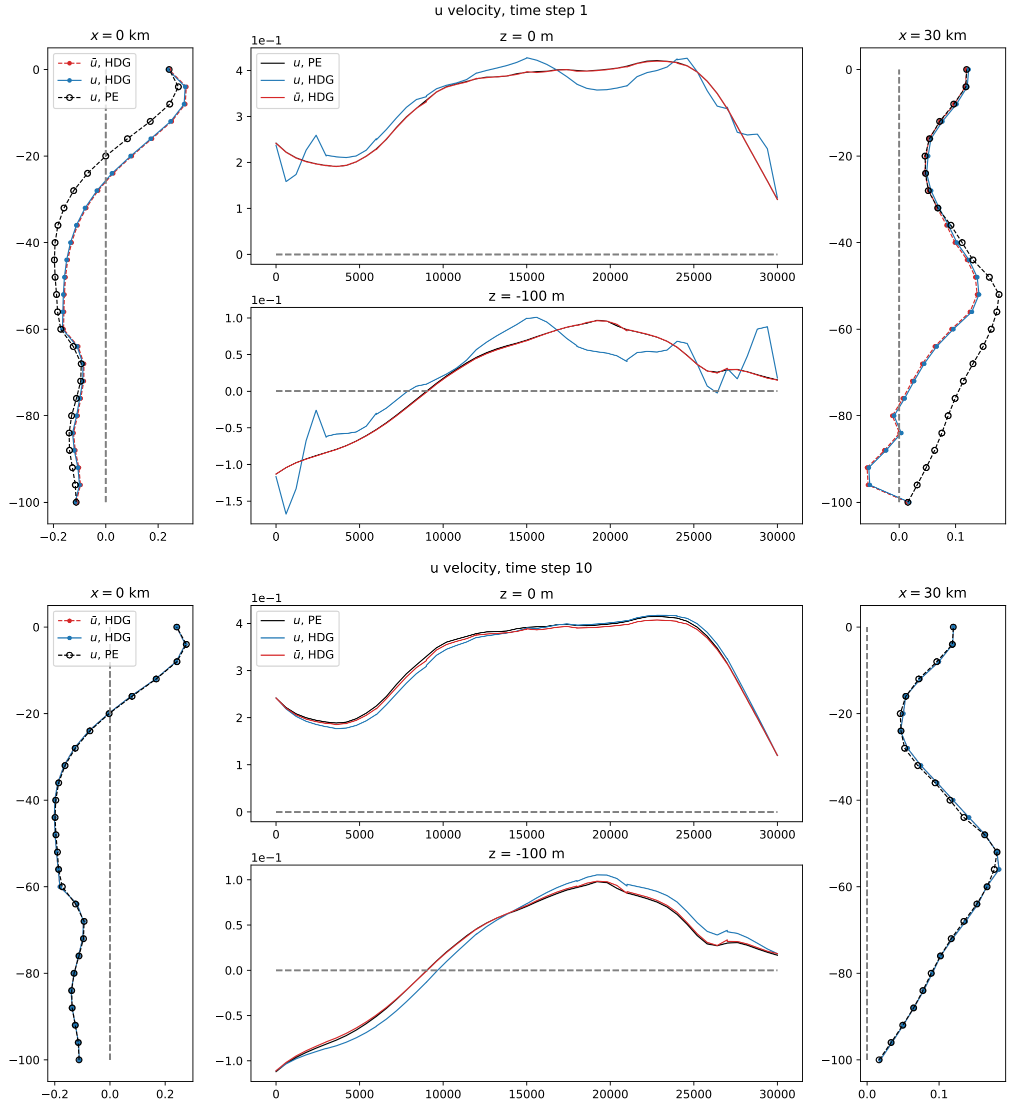


Figure 3-11: Horizontal velocities at the boundaries of the nested models at time step $k = 1$ (left) and $k = 10$ (right).

To validate these assertions, we plot the velocity field $\boldsymbol{u}_{\mathrm{NHS}}$ along the boundary of the domain for the 2D simulation described at length in §3.6.3, as well as the PE data $\boldsymbol{u}_{\mathrm{PE}}$ that the model takes as boundary conditions. The two panels in Figure 3-11 show quantities relating to the u-component of the total velocity: the predictor velocity $\bar{u}_{\mathrm{NHS}}$ and the stage-final velocity u_{NHS}, as well as the boundary data u_{PE} at both the first time step and time step number $k = 10$. It is clear that there are differences between the nonhydrostatic model velocities and the boundary condition u_{PE}, despite u_{PE} representing the Dirichlet boundary value $\boldsymbol{u}_{\mathrm{PE}} = \boldsymbol{g}_D$ in (3.12). The stage-final velocity u_{NHS} need not agree with the boundary condition on the velocity predictor $\bar{u}_{\mathrm{NHS}}$ as a consequence of the projection method. However, the fact that the velocity predictor $\bar{u}_{\mathrm{NHS}}$ does not agree with the boundary condition value is a consequence specific to the HDG methodology. Unlike other DG schemes, boundary conditions are not applied on the solution degrees of freedom directly, but

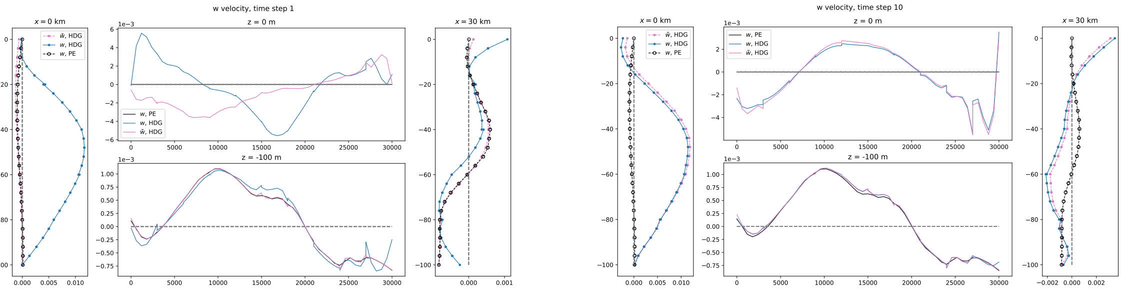


Figure 3-12: Vertical velocities at the boundaries of the nested models at time step $k = 1$ (left) and $k = 10$ (right).

rather, on the trace quantity $\hat{\bar{\boldsymbol{u}}}$. Like all HDG primal unknowns, the actual predictor velocity $\bar{\boldsymbol{u}}$ is reconstructed by solving a local-problem independently on each element using the HDG flux $\hat{\bar{\boldsymbol{u}}}$ as boundary conditions; for details see [191, 193] and references therein. Therefore, the predictor velocity on the boundary is not obligated to agree exactly with the boundary condition $\hat{\boldsymbol{u}}_{\mathrm{PE}}$, but is a compromise between the underlying PDE, as represented by the element-local system, and the boundary condition. This is a attractive feature in the context of model nesting, as it allows the two different sets of physics to equalize through weak imposition of the boundary conditions. By time $t = 10$, the differences are substantially smaller as the two different models have balanced dynamically, and the boundary conditions are well-represented by the high-order nonhydrostatic model.

A similar result can be seen in the vertical velocity; by the tenth time step, the oscillations and disagreements with the PE-supplied boundary data are very small on the top and bottom boundary. On the sides of the domain, the HDG solution is not in close agreement with the PE boundary condition. This is a consequence of the extreme difference in scales between the horizontal and the vertical, making the numerical solution very sensitive to the vertical velocity at the boundary–the difference between the two is a direct numerical representation of the dynamic balance between the models.

In summary, the relaxation of the numerical solution from the boundary conditions specified by a different model is beneficial from both a stability and a modeling perspective, rendering both the discontinuous representation of the solution and the HDG-specific imposition of boundary conditions advantageous numerical features in the context of compatibility between nested models governed by different PDEs.

3.6.2 CALYPSO Alboran Sea Experiment 2019

The Coherent Lagrangian Pathways from the Surface Ocean to Interior (CALYPSO) initiative addresses the challenge of identifying and quantifying three-dimensional, time-dependent transport pathways of water parcels and particles from the surface ocean to the interior, a process known as subduction. The project focuses on the southwest Mediterranean Sea region, depicted in the black shaded region in Figure 3-13. The approach taken by this project to tackle this challenge is to combine *in situ* observations (using state-of-the-art observation technologies) with the latest modeling and analysis tools. These observations were collected during several real-time sea experiments, including one taking place in the Alboran Sea (in the southwestern Mediterranean Sea) during the late winter of 2019 (March 28 to April 11); see modeling regions, Figure 3-13. While this experiment occurred, real-time, high-resolution, deterministic Eulerian simulations were completed using the MIT Multidisciplinary Simulation, Estimation, and Assimilation System primitive equation (MSEAS-PE) modeling system [107, 105]. At its core, MSEAS-PE is a solver of governing fluid and ocean dynamics equations. It is part of an extensive modeling system for hydrostatic primitive-equation dynamics with a nonlinear free surface, based on second-order structured finite volumes [107]. It is used to study and quantify tidal-to-mesoscale processes over regional domains with complex geometries and varied interactions. Among its many subsystems are: initialization schemes [105], nested data-assimilative tidal prediction and inversion [169]; subgrid-scale models [145, 146]; and advanced data assimilation [143, 144, 149, 147]. MSEAS has been validated for fundamental research and for realistic simulations in varied regions of the world's ocean [164, 199, 104, 89, 215, 154, 52, 151, 155, 235, 135, 97, 157, 116]. It has many applications include monitoring [152], real-time acoustic prediction and data assimilation [259, 142, 68, 3, 4], ecosystem prediction and environmental management [22, 55], and interactive visualization schemes [6, 5, 90].

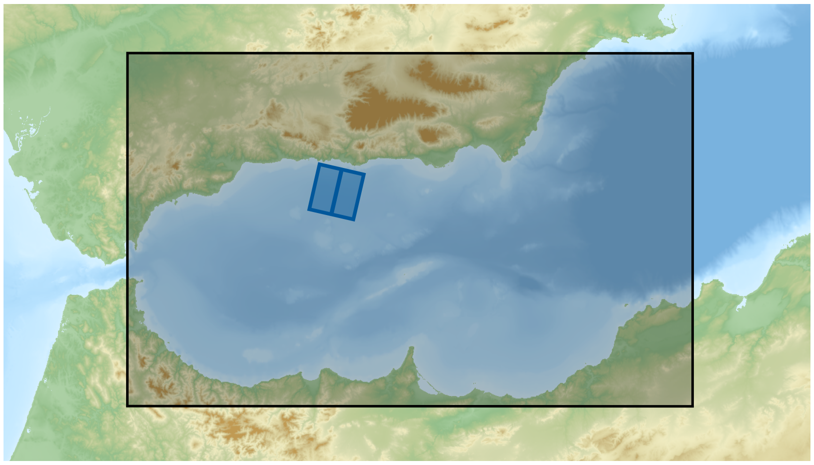


Figure 3-13: Modeling domain for model nested runs in the Alboran Sea§3.6.3, showing the MSEAS-PE domain (shaded, black), and the HDG NHS nested modeling domain (blue) in 2D (center slice) and 3D (box).

During the 2019 experiment period in the Alboran Sea, two gales and several minor events occurred. A gale, with wind stress in excess of 0.5 $\mathrm{N} \cdot \mathrm{m}^{-2}$, began on March 26, with the prevailing winds directed toward the west-southwest. Between March 27 and March 31, the winds slackened, but remained directed west-southwest. From March 31 to April 5, winds were light and variable, but reversed direction. A second gale occurred from April 5 to April 7. Following this gale, the winds gradually weakened, especially after April 11, but remained directed east. The presence or absence of significant wind stress was seen to be strongly correlated with both the subduction processes observed and with the development of submesoscale ripples in the 2 m vorticity forecasted by the MSEAS-PE model. Such ripples, located near the coast at right angles to the prevailing wind direction (consistent with Ekman transport), are hypothesized to arise from the interaction of seawater masses of differing densities forced by wind and/or currents. However, the underlying (hydrostatic and non-hydrostatic) dynamical mechanisms are not yet well understood. Thus, we aim in this section to use our high-resolution, non-hydrostatic HDG solver to simulate these submesoscale non-hydrostatic dynamic effects (such as Rayleigh–Taylor instabilities) not resolved by the MSEAS-PE model during and just after these strong wind events.

The 27-day MSEAS-PE model output used for ICs and BCs to the non-hydrostatic HDG model was based on the following setup. The Alboran Sea modeling domain (Figure 3-13), a 480×773 grid spanning approximately 267 km by 430 km, had a horizontal resolution of $1/200°$ (557 m) and 70 vertical levels with optimized level depths. The momentum and tracer diffusivity parameters were $\nu_x = 100\ \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, $\nu_z = 0.008\ \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, $\kappa_x = 100\ \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, and $\kappa_z = 0.004\ \mathrm{m}^2 \cdot \mathrm{s}^{-1}$, the vertical values applicable to the mixed layer only. The time step size was 100 s, with hourly output.

3.6.3 Two-dimensional nested model comparisons

For our first set of nested model numerical experiments, we consider a 2D nonhydrostatic domain 30 km wide and 100m deep, stretching along a slice in the Alboran sea shown in Figure 3-13. The nonhydrostatic HDG model uses as initial conditions and boundary conditions the MSEAS-PE model output over the slice.

The simulation begins on March 31, 2019 at 15:00:00Z, corresponding to the "simulation time" $t = 0$, and are run for a five-day period, ending on April 5, 2019 at 15:00:00Z. We discretize the domain with 300 elements in the horizontal and 30 elements in the vertical, using polynomial degree $p_{\text{order}} = 3$ for all finite element fields and we use a constant time step of $\Delta t = 25$ seconds to respect the CFL condition as computed in §3.5.1.1. We use a horizontal diffusivity $\nu_x = 30\,\text{m/s}^2$ and a vertical diffusivity of $\nu_z = 0.008\,\text{m/s}^2$ for both the momentum and the tracer equations.

Figure 3-14 shows a comparison between the two model outputs over the same two-dimensional slice, with the same effective resolution during the weather event beginning on April 5. Both the density perturbation ρ' and the velocity field $\boldsymbol{u}$ show substantial departures from the hydrostatic prediction. While the hydrostatic velocity $\boldsymbol{u}_{\text{PE}}$ includes excitation due to the weather event, the degree and scales of overturning are substantially understated as compared to that predicted by the nonhydrostatic model. Another prominent difference can be seen in the vertical component w of the hydrostatic and nonhydrostatic vertical velocities. The hydrostatic velocity w_{PE} is more columnar in nature as a consequence of the lack of the nonhydrostatic component of the pressure p'. By contrast w_{NHS} is characterized by less columnar striations over the domain, which contribute strongly to the RTI-like instabilities that develop in the density perturbation field ρ'_{PE}.

We can see the effects of the nonhydrostatic instabilities more clearly in derived quantities. The squared Brunt-Väisälä (or buoyancy) frequency N^2 is a well-known measure of the stability of a fluid, defined as

$$N^2 = -\frac{g}{\rho_0}\frac{\partial \rho}{\partial z}, \tag{3.45}$$

which describes the oscillation frequency of a parcel of water with density ρ_0 displaced from equilibrium in a stratified background density ρ. With z defined as positive upward, a negative value of N^2 corresponds to locally statically unstable seawater, where higher density water resides on top of lower density seawater. In Figure 3-15b, we plot the scaled buoyancy frequency $\partial\rho'/\partial z = -\rho_0 N^2/g$ to indicate regions of local dynamic instability for both models. The nonhydrostatic density gradient indicates the presence of larger and more spatially variable instabilities over the entire domain. The planar vorticity

$$\nabla \times \boldsymbol{u} = \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right)\hat{\boldsymbol{y}}$$

for both models is shown in Figure 3-15a and indicates the presence of strong overturning in both the top and bottom of the domain for the nonhydrostatic model.

In analyzing the outputs of the two models, it is evident that nonhydrostatic dynamics play a critical role in the simulation results. Yet, despite these differences, there is remarkable alignment between the timescales and length scales produced by each model. This suggests that both models successfully capture the essential dynamics and spatial-temporal structures, thereby demonstrating their robustness and reliability. Moreover, the exceptional level of agreement that exists between the two models both before and after the weather event further lends credence to this assertion. The MSEAS PE model and non-hydrostatic HDG model demonstrate strong synchronicity and consistency outside the event window, establishing their credibility as useful tools in ocean forecasting. Consequently, these similarities corroborate the fact that both models, though divergent in their approach to nonhydrostatic dynamics, nonetheless provide accurate and reliable predictions for the given weather event.

3.6.4 Three-dimensional model comparisons

The same procedure used for initializing and nesting the HDG model within the larger PE model can be readily applied to 3D simulations. We do so here, instantiating both models with a domain enclosing the slice shown in §3.6.2. The 3D nonhydrostatic model is initialized using the MSEAS PE model for initial and boundary conditions over the run duration, beginning on March 26th, 2019, 15:00:00Z. The nested HDG model was run for a simulation duration of three days as part of a

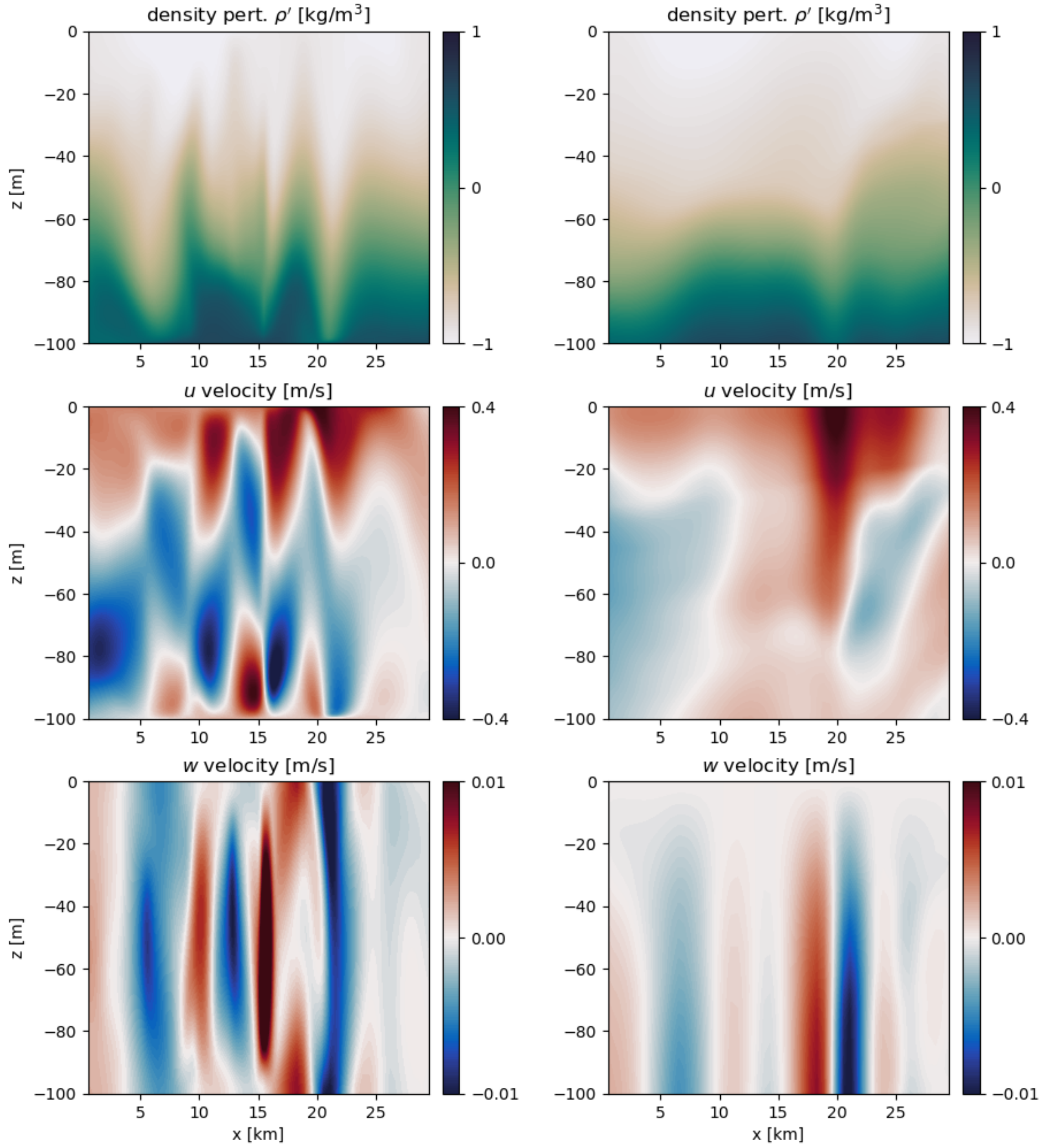
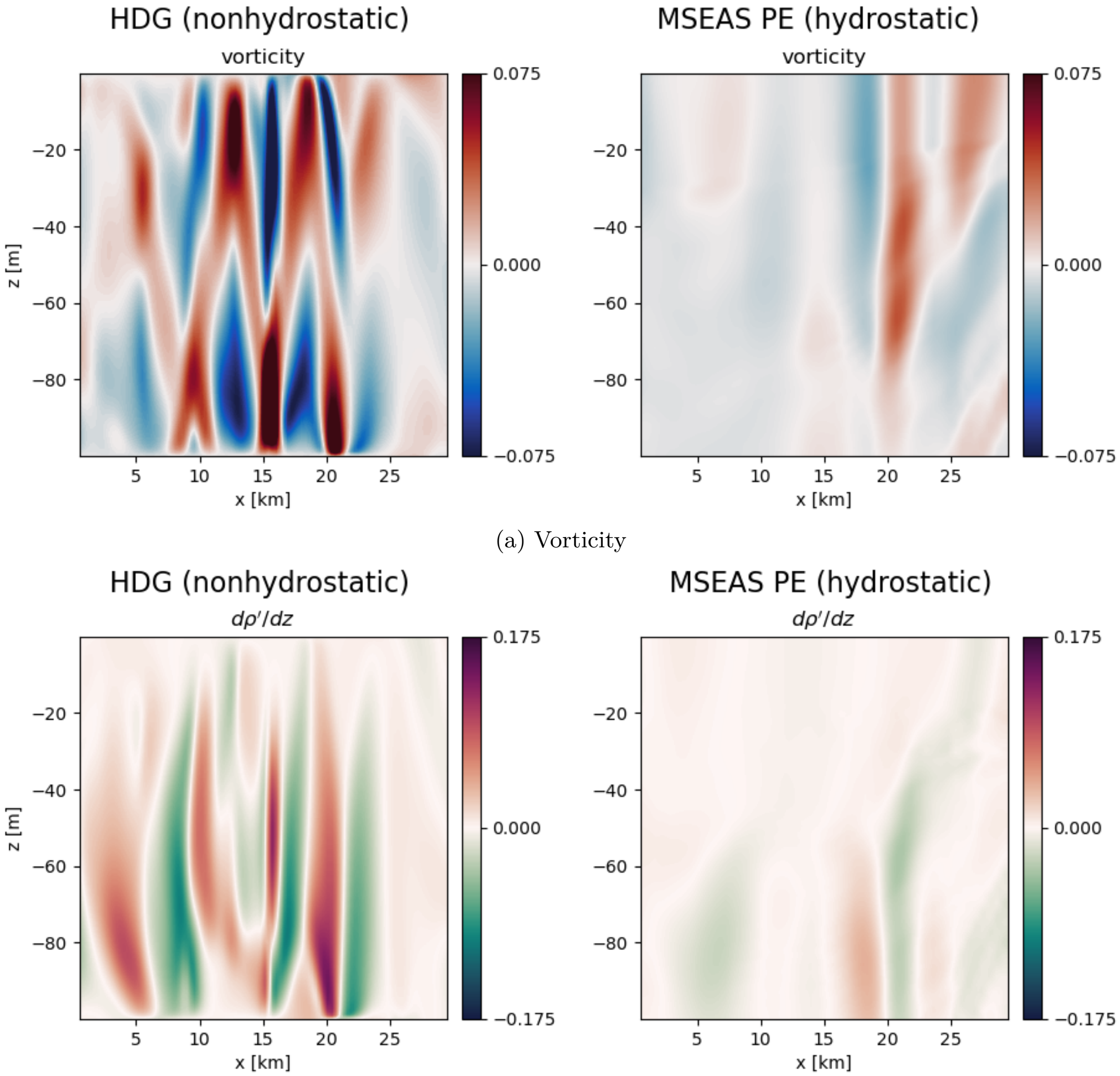


Figure 3-14: Comparison between the high-order nonhydrostatic model (left) initialized from primitive equation (PE) data and the hydrostatic primitive equation model (right) during the Alboran Sea weather event on April 5.



(a) Vorticity

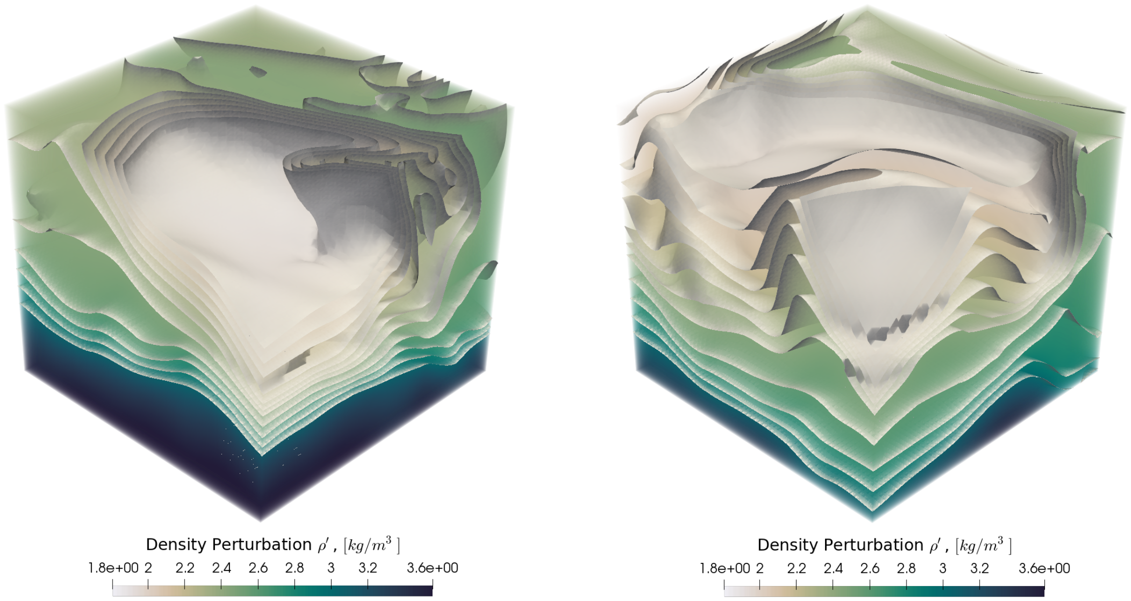
(b) Instability metric $\partial\rho'/\partial z$

Figure 3-15: Comparisons between the high-order nonhydrostatic model initialized from PE data and the hydrostatic PE model during the Alboran Sea weather event on April 5 at simulation time $t = 89.2$ hours, shown for planar vorticity (left) and vertical density gradient $\partial\rho'/\partial z$ (right).

project reanalysis.

The simulation begins on March 26, 2019 at 15:00:00Z, corresponding to the "simulation time" $t = 0$, and is run for a four-day period, ending on March 30, 2019 at 15:00:00Z, capturing the first weather event. We focus on the simulating the instabilities that develop during the first gale and immediately afterward. We discretize the domain with 50 elements in each horizontal direction and 15 elements in the vertical, using polynomial degree $p_{\text{order}} = 3$ for all finite element fields. We use a constant time step of $\Delta t = 50$ seconds to respect the CFL condition as computed in §3.5.1.1. We use a horizontal diffusivity $\nu_x = 100\,\mathrm{m/s^2}$ and a vertical diffusivity of $\nu_z = 0.008\,\mathrm{m/s^2}$ for both the momentum and the tracer equations, to match the MSEAS-PE model.

Simulation results for the density perturbation field ρ' are shown for two separate times in Figure 3-16. The first visualization corresponds to a time close to the start of the simulation, where the wind-cooled, heavier water from the Mediterranean Sea is advected over lighter water from the Atlantic, forming an overhang clearly visible in the density perturbation. The second visualization corresponds to a time close to the end of the simulation, showing internal waves advected into the nonhydrostatic domain.


Density Perturbation ρ' , $[kg/m^3]$
1.8e+00 2 2.2 2.4 2.6 2.8 3 3.2 3.6e+00
Density Perturbation ρ' , $[kg/m^3]$
1.8e+00 2 2.2 2.4 2.6 2.8 3 3.2 3.6e+00

(a) Simulation time 0.55 days.

(b) Simulation time 2.9 days.

Figure 3-16: Density perturbation fields for the MSEAS-PE-nested nonhydrostatic HDG model visualized in terms of iso-surfaces of ρ' with volumetric coloring at different simulation times. The right panel shows heavier water in the process of being advected over lighter water, leading to unstable stratification. The left panel shows waves that form due the mixed-layer instabilities.

Focusing on simulation times around the first visualization depicting the wind-driven unstable stratification (Figure 3-16a), we attempt to analyze the nonhydrostatic dynamics that arise under these conditions. We remark that the wind-forcing is not implemented in the HDG model directly, but rather, is indirectly communicated through the velocity forcing at the top of the domain due to the MSEAS-PE model.

Figure 3-17 through Figure 3-19 are a set, corresponding to the same time steps of the HDG nonhydrostatic simulation (left to right, top to bottom). These times are tabulated by sub-figure number in Table 3.4. Each sub-figure in Figure 3-18 depicts a clipped plot of the interior density perturbation ρ' between the 2.4 kg/m^3 and 2.5 kg/m^3 isopycnals. Figure 3-18 depicts slices of vertical velocity at 25 meter and 50 meter depths in the mixed layer, whereas Figure 3-19 solely depicts the vertical velocity at a depth of 50 meters to complement the plots in Figure 3-18. This

Sub-figure	Time step	Time [s]	Time [days]	Date/Time
-	-	0	0.0	2019-03-26 15:00:00Z
1	420	21000	0.243	2019-03-26 20:50:00Z
2	576	28800	0.333	2019-03-26 23:00:00Z
3	792	39600	0.458	2019-03-27 02:00:00Z
4	1044	52200	0.604	2019-03-27 05:30:00Z
5	1576	78800	0.912	2019-03-27 12:53:20Z
6	1840	92000	1.065	2019-03-27 16:33:20Z

Table 3.4: Simulation times corresponding to the sub-figures in Figure 3-17 through Figure 3-19.

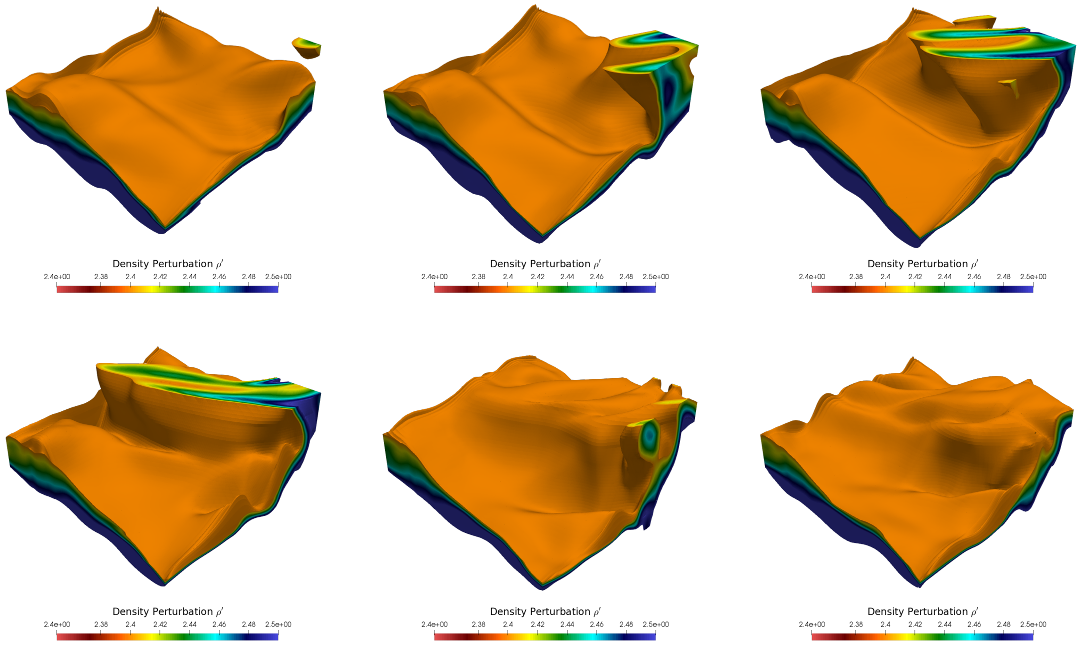


Figure 3-17: Density perturbation field ρ' taking values in the range [0.24, 0.25] at time the steps tabulated in Table 3.4 (left-to-right, top-to-bottom), depicting the wind advection of heavier water over lighter water, the formation of instabilities accompanied by stronger vertical velocities, and re-establishment of stable stratification.

standalone portrayal allows for an unobstructed comparison with the other figures, providing an opportunity to examine the attributes present at this specific depth. To provide additional context, the sub plots of Figure 3-20 show slices of the entire range of the density perturbation, covering the range between the 1.6 kg/m^3 and 2.7 kg/m^3 isopycnals at the start of, and immediately after the development of the unstable stratification. Similarly, Figure 3-21 and Figure 3-22 depict the vertical velocities along the domain center-lines, and at depths of 33 and 67 meters, along with the $\rho' = 2.45$ kg/m^3 isopycnal, to provide a unified picture of the density perturbation and vertical motion.

Frontal instabilities in the MSEAS-PE code lead to the development of "feathering", which can be seen entering the nonhydrostatic model domain in sub-figures 2-3. These frontal instabilities result in the development of the overhang, and the corresponding visualizations of the vertical velocity w shown in Figure 3-18 illustrate the strong vertical velocities that emerge around the instabilities, contributing to mixing and re-stratification. Figure 3-21, depicting the same time as sub-figure 3, shows these vertical velocities along with the $\rho' = 2.45$ kg/m^3 isopycnal, which also displays the development of overhang features. By 2019-03-27 05:30:00Z, shown in sub-figure 4, vertical stability has been re-attained, and the feathering has been consolidated into an outcropping. In sub-figure 5, at 2019-03-27 12:53:20Z, a parcel of denser water is advected into the domain and is quickly subducted into the interior of the domain, which is visible in the strong downward vertical velocities which arise close to the instability, visible in Figure 3-18. Sub-figure 6 shows that by 2019-03-27 16:33:20Z, the entire clipped region is stably stratified once again. However, Figure 3-22 which depicts the same time, shows the presence of vertical striations that develop immediately after the subduction event. These instabilities are reminiscent of the 2D slice simulation. The cause of, and implications of these striations are not yet fully understood, and constitute the subject of ongoing research.

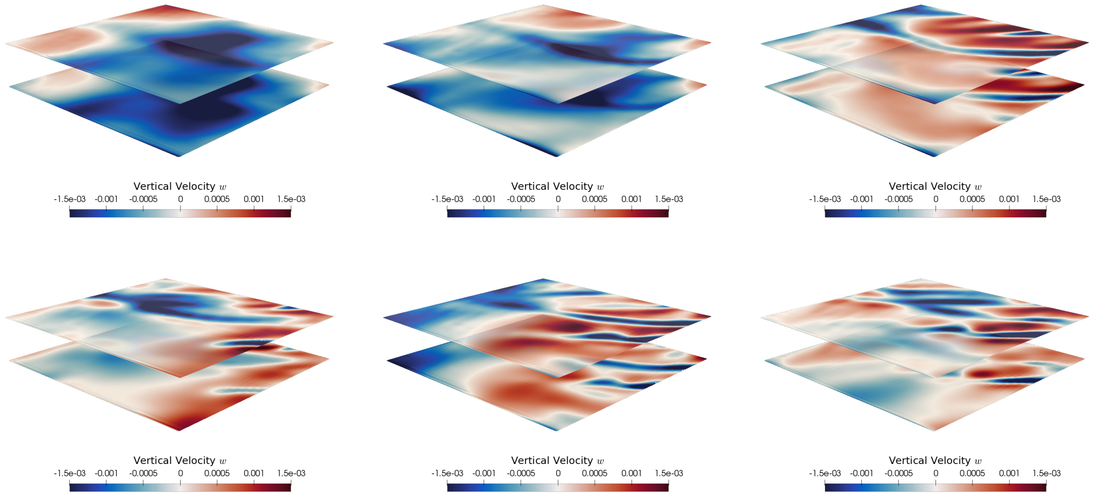

Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical Velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03

Figure 3-18: Vertical velocity w slice, shown at 25 meter and 50 meter depths, corresponding to the sub-figures in Figure 3-17, at the times in Table 3.4

In summary, as we observe the unstably stratified region being advected into the domain, there is a noticeable emergence of strong motions in the vertical direction. However, these vertical motions do not cease once the stratification regains stability. While less pronounced, these movements persist even in a more stable environment, demonstrating the lasting impact of the initial advection and the resulting instabilities on the domain. The striations in the vertical velocities, such as those visible in Figure 3-22, show additional instability as compared to the hydrostatic PE model, but exhibit similar length and time scales overall. Overall, the high-order nonhydrostatic model was able to capture a wide-range of nonhydrostatic behaviors in the modeling domain, providing comparisons to the corresponding hydrostatic simulation and allowing investigation of nonhydrostatic submesoscale

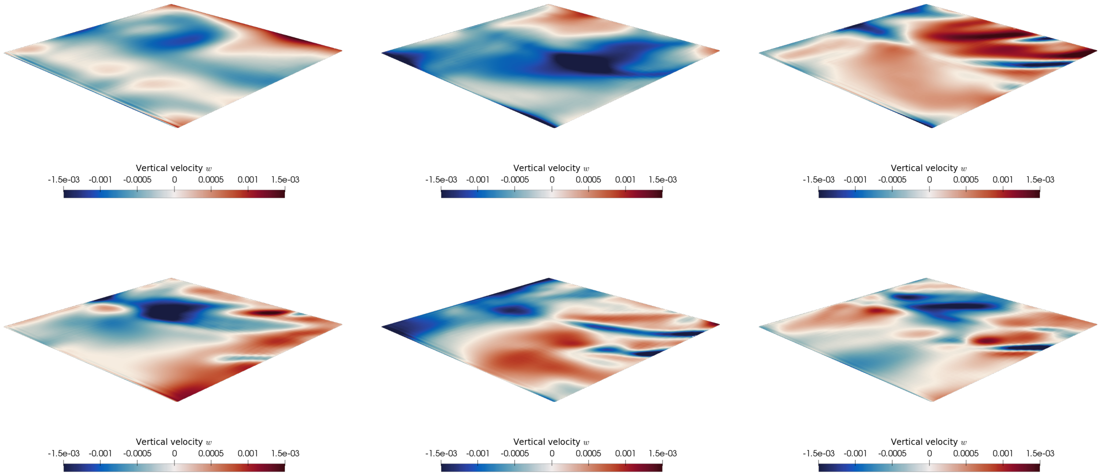

Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03
Vertical velocity w
-1.5e-03 -0.001 -0.0005 0 0.0005 0.001 1.5e-03

Figure 3-19: Vertical velocity w slice, shown as slices at 50m depth.

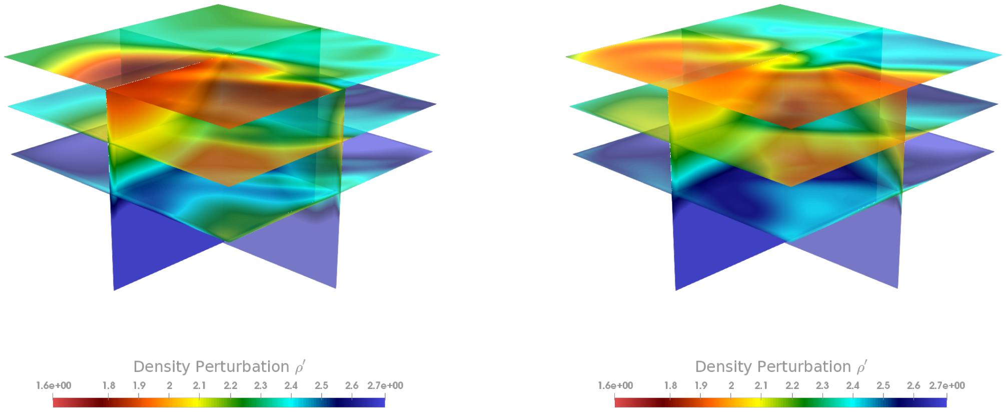

Density Perturbation ρ'
1.6e+00 1.8 1.9 2 2.1 2.2 2.3 2.4 2.5 2.6 2.7e+00
Density Perturbation ρ'
1.6e+00 1.8 1.9 2 2.1 2.2 2.3 2.4 2.5 2.6 2.7e+00

Figure 3-20: Density perturbation at 2019-03-27 02:00:00Z (left) and 2019-03-27 07:56:40Z (right), depicting the advection of denser water pulled over lighter water by wind-forcing in the right-hand side of domain.

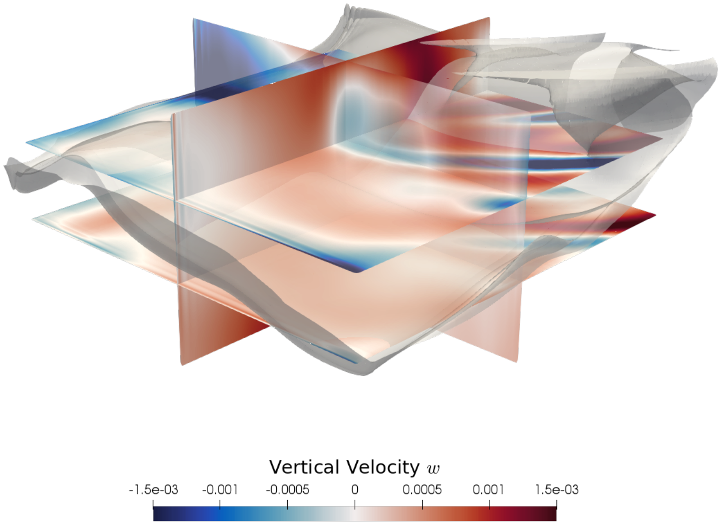


Figure 3-21: Vertical velocity slice schematic with 2.45 kg/m^3 isopycnal overlaid, 2019-03-27 02:00:00Z.

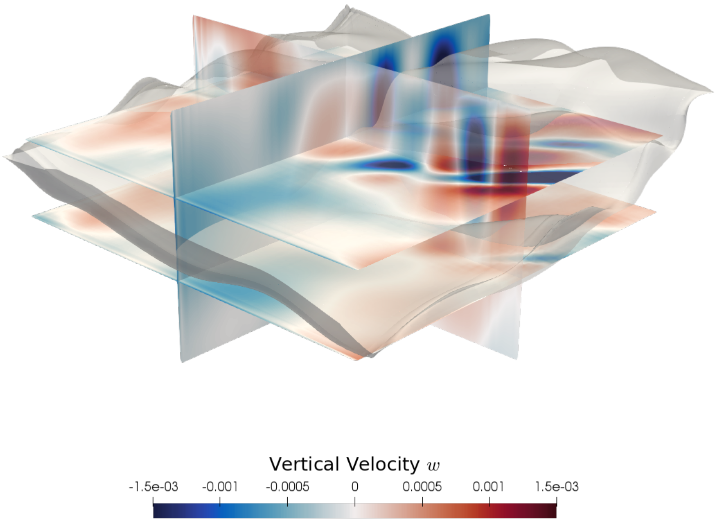


Figure 3-22: Vertical velocity slice schematic with 2.45 kg/m^3 isopycnal overlaid, 2019-03-27 16:33:20Z.

phenomena.

3.7 Conclusion

In this chapter we derived and implemented a novel high-order HDG-FEM nonhydrostatic and hydrostatic model based on the projection scheme originally proposed in [243]. To verify the spatial and temporal convergence of the model, we verified that the model achieves optimal spatial and temporal convergence using manufactured solutions. We additionally validated the results against results from linear gravity wave theory, by running simulations of free surface seiches and internal seiche test cases. We recovered the physically correct behavior for the interior fields and the free surface over wide variety of domains, capturing both hydrostatic and nonhydrostatic behavior correctly in all cases. To provide additional verification of the model, we compare it to a legacy finite volume code which implements the incompressible Navier–Stokes equations with Boussinesq approximation and find that our model shows excellent agreement with its predictions but at substantially reduced cost, lending additional credence to the ability of the model to accurately capture nonhydrostatic behavior using high-order methods.

Then we developed and implemented a model nesting approach, running our nonhydrostatic code within the larger MSEAS-PE hydrostatic ocean model. We ran nested nonhydrostatic simulations from realistic ocean data from the Alboran Sea in both 2D and 3D, and provided comparisons between the hydrostatic and nonhydrostatic models in the mixed layer.

Future work will constitute additional hydrostatic/nonhydrostatic comparisons between the two models in order to scientifically investigate the effect of nonhydrostatic dynamics on regional mesoscale and submesoscale ocean processes.

Chapter 4

Applications to Uncertainty Quantification and Data Assimilation

4.1 Introduction

The importance of data assimilation (DA) and uncertainty quantification (UQ) in ocean modeling cannot be overstated. Both these facets play a critical role in improving the accuracy, reliability, and predictive capabilities of ocean models, which, in turn, have profound implications for understanding our planet's climate systems, predicting weather patterns, and managing marine resources.

Data assimilation, at its core, is a method of integrating observations and model simulations in order to improve the initial conditions and parameters of a model. In the context of ocean modeling, this implies combining real-time observational data from sources like buoys, satellites, and ship-based instruments with computational models. By doing so, data assimilation enhances the accuracy of the ocean models by correcting biases and errors, providing a more detailed and precise representation of oceanic conditions. This, in turn, significantly improves our ability to predict future oceanic behaviors, which is essential for a range of applications from weather forecasting to naval operations. Uncertainty quantification, on the other hand, is a process that acknowledges and evaluates the inherent uncertainties present in any modeling endeavor. No model is a perfect representation of reality; all models are simplifications that introduce some degree of uncertainty. In ocean modeling, uncertainties can arise from multiple sources including the observational data, model parameters, or the model structure itself. Uncertainty quantification provides a systematic approach to assess these uncertainties, which can then be used to evaluate the confidence in the model predictions. By providing an estimate of the reliability of model outputs, uncertainty quantification helps to guide decision-making processes in areas as diverse as climate change mitigation, oceanographic research, and marine resource management.

In this chapter, while we don't introduce any new methodologies, we take a deep dive into addressing the intricacies involved in adapting data assimilation and statistical inference approaches for use with high-order discontinuous Galerkin finite element methods. The task of handling discontinuous data, particularly within the context of uncertainty quantification, presents distinct challenges related to solution representation that require careful navigation.

We focus our attention on two canonical problems often encountered in the field of uncertainty quantification: ensemble Kalman filtering, and the Bayesian inversion of a field representing unknown, spatially-variable PDE coefficients. To implement solutions to these problems, we employ the HDG finite element solvers developed in §2, demonstrating proof-of-concept viability of these uncertainty quantification methods for use in nested models making use of real ocean simulation data, as in §3.6. Furthermore, to the best of the author's knowledge, this work represents the only published approach that applies both UQ methods to HDG finite element solvers.

We address the problem of ensemble Kalman filtering in §4.2 by introducing a non-intrusive method that aligns seamlessly with the matrix-free operations and elemental parallelism described earlier in §2. This approach serves to effectively streamline the processing of ensemble Kalman filtering without necessitating any disruptive alterations to the established finite element operations. We show that such a procedure can be done robustly in the discontinuous space directly with good results, performing an investigation similar to those in [205] but in 2D instead of 1D. For applications of similar approaches to state estimation of tidal hydrodynamics in a DG-FEM context, see [238].

We address the problem of Bayesian inversion in §4.3, where we solve the inverse problem for an unknown diffusivity field using a Bayesian approach to sample from the posterior distribution directly by employing Markov Chain Monte Carlo (MCMC) methods [25]. While MCMC techniques are generally regarded as the gold-standard of Bayesian inference [190], they are often very expensive to carry out. In the context of solving the inverse problem associated with a stochastic partial differential equation, the sampling process requires the numerical solution the PDE at every iteration. In this regard, the rapid convergence of high-order HDG methods to smooth solutions as shown in previous chapters is advantageous, as it allows for the solution of the PDE on relatively coarse meshes, as we will demonstrate in §4.3.3. An additional computational challenge in any inference problem, not only those involving PDEs, is a high-dimensional parameter space in which inference must be performed. In the case of inferring a continuous diffusivity field, the problem is infinite-dimensional; naive parameterizations of the parameter space, such as nodal representations quickly become computationally intractable even at modest problem sizes and are subject to a number of failure modes [254]. To address this issue, we employ a parameterization based on polynomial chaos, which we apply to the discontinuous fields to infer relevant quantities in a reduced-order model of the underlying problem, extending the work in [179] to a two-dimensional setting. The aforementioned dimensionality reduction technique has recently been used to solve inverse problems for the log-permeability field in the context of fluid flow through porous media [217]. We compare the performance of standard Metropolis-Hastings samplers with more advanced gradient-based samplers and provide performance assessments over a range of numerical experiments. In doing so, we show that this combination of state-of-the-art HDG solvers coupled with this reduced-dimension parameterization results in the effective solution of the inverse problem, with computational performance that is competitive with or better than modern MCMC benchmarks for coefficient problems [8].

4.2 Ensemble Kalman filtering for high-order discontinuous Galerkin fields

The ensemble Kalman filter (EnKF) is commonly used for uncertainty quantification and data assimilation in high-dimensional problems. Problems in computational fluid dynamics (CFD) and geophysical fluid dynamics (GFD) often have such features, given the large number of problem degrees of freedom involved in the representation of the physical system. One such area of application is that of contaminant transport in a fluid as modeled by a convection-diffusion PDE. We examine the use of the EnKF to assimilate observed downstream contaminant data in channel flow in this section.

Accurately modeling fluid physics often requires discretization of the underlying equations, as is common in computational fluid dynamics or geophysical fluid applications. Finite element methods for CFD problems are advantageous in the sense that they allow an arbitrarily high-order solution in space; however, it is often the case that discretization error due to time marching or splitting schemes lead to numerical solutions which do not accurately predict the behavior of the dynamical system in question for long timescales compared to the numerical time step. Data assimilation techniques may provide an answer to this problem: by accounting for observed data from the real physical system and updating the numerical solution accordingly, it is possible to improve the predictive capabilities of the simulation. It is also always the case that the observations of the physical system are not certain—any measurement made without knowledge of its uncertainty is meaningless. Since CFD discretizations tend to involve a large number of degrees of freedom in the numerical solution at each time step, the ensemble Kalman Filter is a popular scheme for data assimilation.

In this investigation, we are interested in accurately predicting the spread of a contaminant in a fluid over a long time horizon compared to the numerical time step. We assimilate observed data of the scalar contaminant field corrupted by known measurement noise using the ensemble Kalman filter to improve the numerical solution. This section is organized as follows. We discuss the setup of the contaminant transport problem in 4.2.1 and provide brief discussion of the ensemble Kalman filter and the details of its implementation in 4.2.2. We describe the data assimilation setup for this investigation in 4.2.4, and we present present results of numerical experiments in 4.2.5. Finally, we offer conclusions in 4.2.7.

4.2.1 Problem setup

We consider the same contaminant transport problem used as a test case in [191]. A contaminant concentration is deposited in a fluid with kinematic viscosity $\kappa = 0.01$, flowing in a two-dimensional channel domain $\Omega = (-1.25, 1.25) \times (0, 10)$, driven by a convective velocity field

$$\boldsymbol{c} = \Big(1 - e^{\gamma x} \cos{(2\pi y)}\, , \frac{\gamma}{2\pi} e^{\gamma x} \sin{(2\pi y)}\Big)\, , \tag{4.1}$$

where the Reynolds number is taken to be Re $= 100$ and $\gamma = \frac{\mathrm{Re}}{2} - \sqrt{\frac{\mathrm{Re}^2}{4} + 4\pi^2}$, as is seen in Figure 4-1. We note that the reference Peclet number is

$$\mathrm{Pe} = \frac{u_{\max} D}{\kappa} \approx 200,$$

where $u_{\max}$ is the maximum horizontal velocity and $D = 1$ is a representative length scale of the flow. At time $t = 0$, the contaminant concentration is a superposition of Gaussian distributions

$$w_0 = e^{-\frac{(x-1)^2+y^2}{0.5^2}} + e^{-\frac{(x-1)^2+(y-0.5)^2}{0.5^2}} + e^{-\frac{(x-1)^2+(y+0.5)^2}{0.5^2}} \tag{4.2}$$

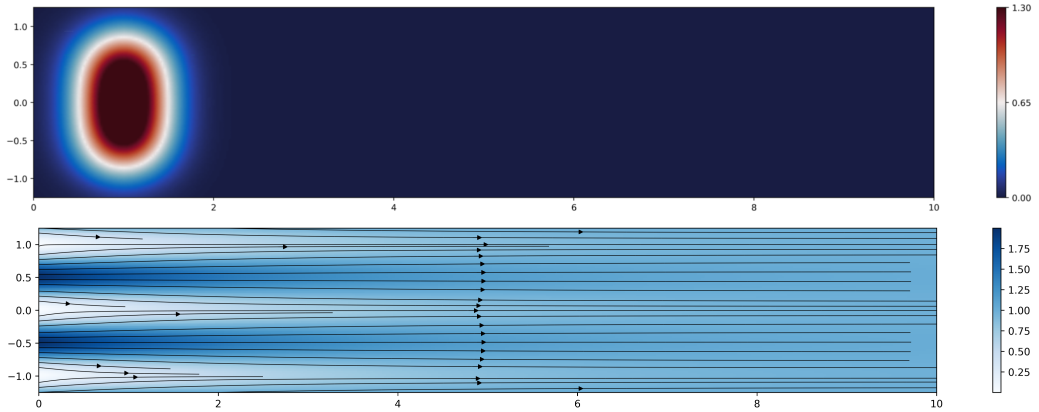


Figure 4-1: Initial condition of contaminant w_0 (top); streamlines of $\boldsymbol{c}$, colored by u (bottom).

Every two seconds $T = 2$, the same contaminant concentration w_0 is injected into the flow field again, until a final time of $T_f = 10$ s. Within each period, the contaminant transport is modeled by the time-dependent convection-diffusion equation

$$\begin{aligned} \frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{c} u) - \kappa \nabla^2 u = 0, &\qquad \text{in } \Omega \times [(j-1)T,\, jT] \\ u = u_0, &\qquad \text{in } \Omega \text{ for } t = (j-1)T, \end{aligned} \tag{4.3}$$

where the initial conditions at the start of each period are

$$u_0 = \begin{cases} w_0, & j = 1 \\ w_0 + u\left(\boldsymbol{x},\, (j-1)T\right), & j > 1 \end{cases} \tag{4.4}$$

The inflow boundary Γ_D is defined by $x = 0$, at which the contaminant concentration u satisfies a homogeneous Dirichlet boundary condition. On the remaining boundary $\Gamma_N = \partial\Omega \setminus \Gamma_D$, the concentration satisfies a homogeneous Neumann condition.

$$\begin{aligned} u = 0 &\qquad \text{on } \Gamma_D \\ \nabla u \cdot \boldsymbol{n} = 0 &\qquad \text{on } \Gamma_N \end{aligned} \tag{4.5}$$

We discretize the PDE using the hybridizable discontinuous Galerkin (HDG) finite element scheme described in [191, 246]. We treat advection explicitly and note that there is a CFL restriction. We perform a numerical simulation of the contaminant's transport over time. We present in Figure 4-2 the computed contaminant concentration u_h at selected times T, $3T$, and $T_f = 5T$. We use a grid of $N_y = 16$, $N_x = 64$ quadrilateral elements and polynomial order $p = 3$. For the temporal discretization, we use an IMEX(2,2,2) scheme with a constant time step $\Delta t = 0.01$. All statistical computations use the NumPy and SciPy numerical libraries [117].

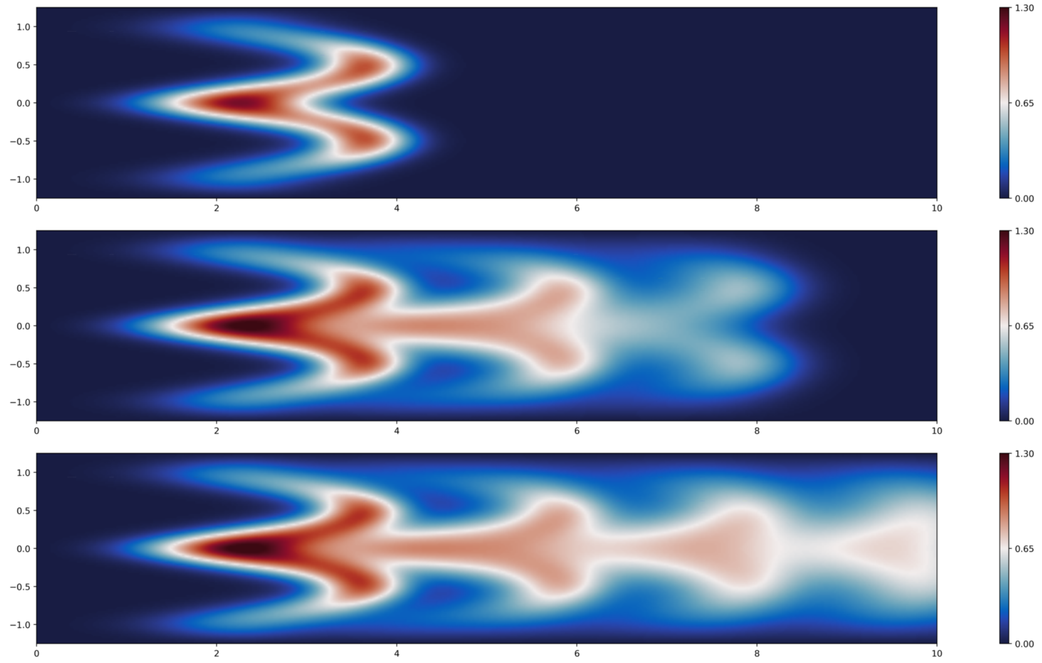


Figure 4-2: Numerical solution of contaminant concentration u at times T (top), $3T$ (middle), $5T$ (bottom)

4.2.2 Application of the ensemble Kalman filter

The ensemble Kalman filter (EnKF) is a monte carlo implementation of the Kalman filter, which is the closed form solution to the Bayesian update filtering problem under linear and Gaussian assumptions [229]. However, since explicitly computing the covariance matrix for sufficiently high-dimensional problems (in a stochastic sense).

4.2.2.1 The Kalman filter

First, we briefly review the Kalman filter. Let $\boldsymbol{x}$ denote an n-dimensional state vector of a model, which is proportional to a multivariate Gaussian prior distribution with mean $\boldsymbol{\mu}$ and covariance matrix Q.

$$p(\boldsymbol{x}) \propto \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T Q^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{4.6}$$

The prior distribution can be evolved in time—in our case by time stepping the numerical solution. At some future time, we receive observation data at certain locations and we must update $\boldsymbol{x}$ to account for the noisy sensor data $\boldsymbol{y}$, which is assumed to also have a pdf of a multivariate Gaussian with mean $H\boldsymbol{x}$ and covariance R. Here H is an "observation matrix" which transforms the state space to the observation space. For example, this matrix might select the members of $\boldsymbol{x}$ which correspond to measurement locations, but could also represent a more complicated transformation. The covariance matrix R describes the relationship of the errors of the sensor data. In the simplest case, the sensor errors are independent of one another, in which case R is a diagonal matrix with entries $R_{ii} = \sigma_i^2$, the variance of the distribution of the error for sensor i. The probability density $p(\boldsymbol{y}|\boldsymbol{x})$ of making the observation $\boldsymbol{y}$ given the current state $\boldsymbol{x}$ is called the "data likelihood" and is

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left(-\frac{1}{2}\left(\boldsymbol{y}-H\boldsymbol{x}\right)^T R^{-1}\left(\boldsymbol{y}-H\boldsymbol{x}\right)\right) \tag{4.7}$$

We use the prior and data likelihood to perform a Bayesian update. We compute the posterior pdf

$p(\boldsymbol{x}|\boldsymbol{y})$; note that since both the prior and data likelihood are multivariate Gaussian, the posterior is as well, and standard manipulations [229] provide the unnormalized posterior

$$p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{d}|\boldsymbol{x})p(\boldsymbol{x}) \quad \propto \exp\left(-\frac{1}{2}(\boldsymbol{x}-\widehat{\boldsymbol{\mu}})^T Q^{-1}(\boldsymbol{x}-\widehat{\boldsymbol{\mu}})\right) \tag{4.8}$$

with posterior mean $\widehat{\boldsymbol{\mu}}$ and covariance given by

$$\begin{aligned} \widehat{\boldsymbol{\mu}} &= \boldsymbol{\mu} + K\left(\boldsymbol{y} - H\boldsymbol{\mu}\right) \\ \widehat{Q} &= \left(I - KH\right)Q \end{aligned} \tag{4.9}$$

where $K = QH^T\left(HQH^T + R\right)^{-1}$ is the Kalman gain matrix.

4.2.2.2 The ensemble Kalman filter

The Kalman Filter describes how to perform the Bayesian update for a single state vector $\boldsymbol{x}$ in the presence of observational data $\boldsymbol{y}$. However, it assumes knowledge of the covariance matrix Q, which may not be available or feasible to compute. The EnKF is a Monte Carlo approximation of the Kalman filter. Instead of evolving the mean and covariance of a state vector $\boldsymbol{x}$, we trace an ensemble of realizations

$$X = [\boldsymbol{x}_0,\, \ldots,\, \boldsymbol{x}_{M-1}] \tag{4.10}$$

where $X \in \mathbb{R}^{n\times M}$. The data is observed in a Monte Carlo sense as well. We consider the data matrix $Y = [\boldsymbol{y}_0,\, \ldots,\, \boldsymbol{y}_{M-1}]$ where $\boldsymbol{y}_i = \boldsymbol{y} + \varepsilon_i$ where ε_i is drawn from the multivariate normal $\varepsilon_i \sim \mathcal{N}(\mathbf{0},\, R)$. Since we no longer have access to the exact covariance matrix Q, we replace it in the Kalman gain matrix with the sample covariance C computed from the realizations in the ensemble.

$$K = CH^T\left(HCH^T + R\right)^{-1} \tag{4.11}$$

The empirical posterior distribution is computed as

$$X^p = X + K\left(D - HX\right) \tag{4.12}$$

where each column corresponds to a single sample "drawn" from the posterior.

4.2.3 Matrix-free implementation

We denote the ensemble mean as $\bar{\boldsymbol{\mu}}$ and sample covariance as C. The posterior ensemble can be expressed as a set of linear operations

$$X^p = X + CH^T\left(HCH^T + R\right)^{-1}\left(D - HX\right) \tag{4.13}$$

however, due to the large number of state variables—the finite element degrees of freedom—as well as the discontinuous and multiply-defined nature of the solution field, it is neither efficient to explicitly form the observation matrix H, nor to rearrange the solution data into a state vector $\boldsymbol{x}$. Instead, we implement the EnKF in a matrix-free manner, computing the posterior without explicitly forming H or the state vector $\boldsymbol{x}$; we follow the approach in [176]. If we did have M vectors $\boldsymbol{x}_k$ representing the state vectors of the ensemble members arranged into a matrix X. We would like to compute the following quantities:

$$E(X) = \frac{1}{M}\sum_{k=1}^{M}\boldsymbol{x}_k, \qquad A = X - E(X), \qquad C = \frac{AA^T}{M-1}. \tag{4.14}$$

Suppose instead we have a representative array of the solution X_k for each of the M ensemble members, together comprising the array X, along with an n-dimensional observation vector. The key idea is to be able to compute the $n \times M$ matrix $H\boldsymbol{x}_k$, which represents the action of the observation operator on each realization. Practically, this is a function which can select the appropriate solution data corresponding to the observation vector; $h(\boldsymbol{x}) = H\boldsymbol{x}$. We note that by linearity of expectation, we can compute the matrix HA column-wise as

$$\begin{aligned} [HA]_i &= H\boldsymbol{x}_i - H\frac{1}{M}\sum_{j=1}^{M}\boldsymbol{x}_j \\ &= h(X_i) - \frac{1}{M}\sum_{j=1}^{M} h(X_j) \end{aligned} \tag{4.15}$$

In this manner we need only evaluate the observation function on the ensemble members, and the posterior can be formed using Algorithm 1.

Algorithm 1 matrix-free formation of posterior ensemble

```
Define prior ensemble array $X$
compute ensemble mean $E(X) = \frac{1}{M}\sum_{j=1}^{M} X_k$
form $A = X - E(X)$
for $k$ in $M$ do
    $[HX]_k = h(X_k)$
end for
copy $HX$ to form $HA$
compute $\mu_{HX} = \frac{1}{M}\sum_{j=1}^{M}[HX]_j$
subtract mean $HA = HA - \mu_{HX}$
compute $P = \frac{1}{M-1}HA(HA)^T + R$
assemble data matrix $D$ with column $D_i = Y + w_i$, with $w_i \sim \mathcal{N}(\mathbf{0}, R)$
define $T = (HA)^T P^{-1}(D - HX)$
for finite element $K$ in mesh $\mathcal{T}_h$ do
    compute posterior $X^p_K = X_K + \frac{1}{M-1}A_K T$
end for
return $X^p$
```

Within the context of DG-FEM methods generally, replacing the linear operator H with a function amounts to construction of a single index map for fast lookup and easily extends to the case of adaptive mesh refinement [10]. Furthermore, the computation of the posterior X^p_K can be computed in an element-by-element fashion independently, and can use similar software abstractions as that used in the element-local assembly and reconstruction steps for the HDG algorithms discussed at length in §2. Lastly, in the cases where the linear operator P is large, efficient explicit methods exist to compute the action of P^{-1} using the Sherman-Morrison-Woodbury formula at substantially reduced cost compared to the linear system representing the PDE [175].

In the case of large ensembles, how the solution data is stored in memory becomes especially relevant to performance. A cache miss is a state in which data requested for processing by a CPU is not found in the cache memory, prompting the system to fetch the data from other memory levels, such as the main RAM or even the disk drive. This can significantly slow down processing times because fetching data from these other memory sources takes longer than retrieving it from the cache. Modern CPUs are significantly faster than the memory, and thus they spend a lot of time waiting for data to be fetched when a cache miss occurs. This waiting period, known as “stall cycles,” can significantly reduce performance. Therefore, efficient cache utilization—which implies minimizing cache misses—is a critical aspect of optimizing the performance of modern computing systems. As DG-FEM implementations on modern hardware are predominantly memory bound [133], these concerns become even more performance critical. Therefore, data should be structured such that primary memory locality is organized by elemental degrees of freedom within the solution vector representation, followed by position in the state $\boldsymbol{x}$, and lastly by ensemble member. Reversing this order and organizing data by spatial locality can result in orders of magnitude slowdown due to cache misses.

4.2.4 Data assimilation setup

We consider the high polynomial order, high resolution simulation shown in Figure 4-2 ($Ny = 16$, $p = 3$), which serves as the "truth" for the purposes of observation data. We consider the coarse discretization ($Ny = 8$, $p = 2$) and a very coarse discretization ($Ny = 4$, $p = 2$) for ensemble runs—see the computational meshes in Figure 4-3. We launch 10^3 ensemble members with uncertain initial condition in x where $x(\omega) \sim \mathcal{N}(1, \sigma_\varepsilon)$. For the purposes of this investigation, we consider $\sigma_\varepsilon^2 = 10^{-4}$.[3]

We introduce the same uncertainty with every repeated injection of the contaminant—if the problem premise is that the first injection is uncertain, it stands to reason that following injections would be uncertain as well.

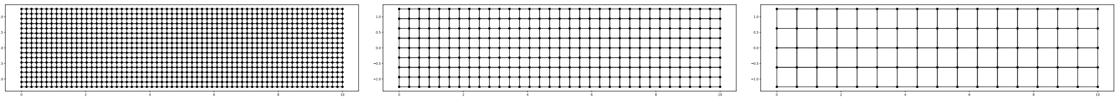


Figure 4-3: The computational meshes considered in this investigation. (Left) high resolution truth $Ny = 16$, (Center) coarse resolution $Ny = 8$, (Right) very coarse resolution $Ny = 4$. The polynomial order of numerical solution for the truth is $p = 3$ and the ensembles, $p = 2$.

We can write the stochastic representation of the initial condition as

$$w_0(\boldsymbol{x};\omega) = e^{-\frac{(x(\omega)-1)^2+y^2}{0.5^2}} + e^{-\frac{(x(\omega)-1)^2+(y-0.5)^2}{0.5^2}} + e^{-\frac{(x(\omega)-1)^2+(y+0.5)^2}{0.5^2}} \tag{4.16}$$

Our study incorporates the sensor configuration as illustrated in the reference figure (Figure 4-4). It's worth noting that the numerical solution, within our chosen framework, exhibits piecewise discontinuity across each element. This is a fundamental characteristic of the numerical scheme, resulting from the HDG discretization technique used, and forms the basis for how the sensor observations are interpreted. Each sensor observation corresponds to four separate nodal degrees of freedom in the state vector, which we denote $\boldsymbol{x}$. This means that every reading we get from a sensor provides us with four separate pieces of information about the state of the system. This multifaceted data capture enriches the volume of information we obtain from each observation, which in turn enhances the fidelity of our model.

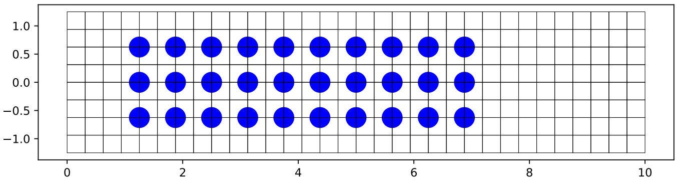


Figure 4-4: The sensor configurations considered in this investigation, overlayed on the coarse mesh.

We note that the coarse runs differ significantly from one another as they are evolved in time; see Figure 4-5, which demonstrates the qualitative difference in solution as well as the difference between the "true" contaminant at the central sensor and the coarse simulations' value of the contaminant over time. The solutions are all plotted on the same view mesh to ensure that the linear interpolation of the plotting agent does not artificially worsen the coarse solutions. It's also worth noting the particular behavior of the contaminant solution. The numerical solution exhibits a dynamic phase, characterized by changes over time, before it eventually reaches a steady state. This process is primarily driven by the repeated injections of the contaminant into the system and spatially-variable but temporally-constant background flow.

[3] In practice, tuning sensor noise is very important. See §4.2.6 for discussion.

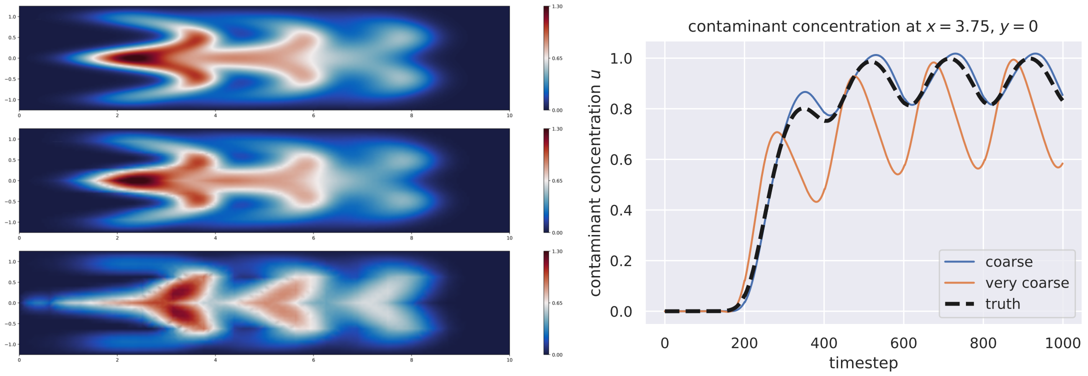


Figure 4-5: [Left] Comparison of numerical solutions u_h at $t = 3T$. (Top) "truth" (Middle) coarse (Bottom) very coarse. [Right] The contaminant as measured at the central sensor.

4.2.5 Numerical experiments

4.2.5.1 Single assimilation

In the first scenario, we aim to provide a clear understanding of the effect of a single Ensemble Kalman Filter (EnKF) update. In order to do this, we execute the coarse simulation for a duration equivalent to 400 time steps. During this run, we assimilate observation data pertinent to the sparse sensor case. Note that from Figure 4-5, $t = 400$ is sufficient time for the truth and coarse simulation to appreciably diverge. The results of this process, for a single ensemble member, are visually represented Figure 4-6. By focusing our attention on the domain center-line, denoted as $y = 0$, it becomes evident that features of the truth have been successfully recaptured, with the posterior distribution accurately centered around the true concentration value. This recovery highlights the effectiveness of our single EnKF update in steering the simulation towards the ground truth.

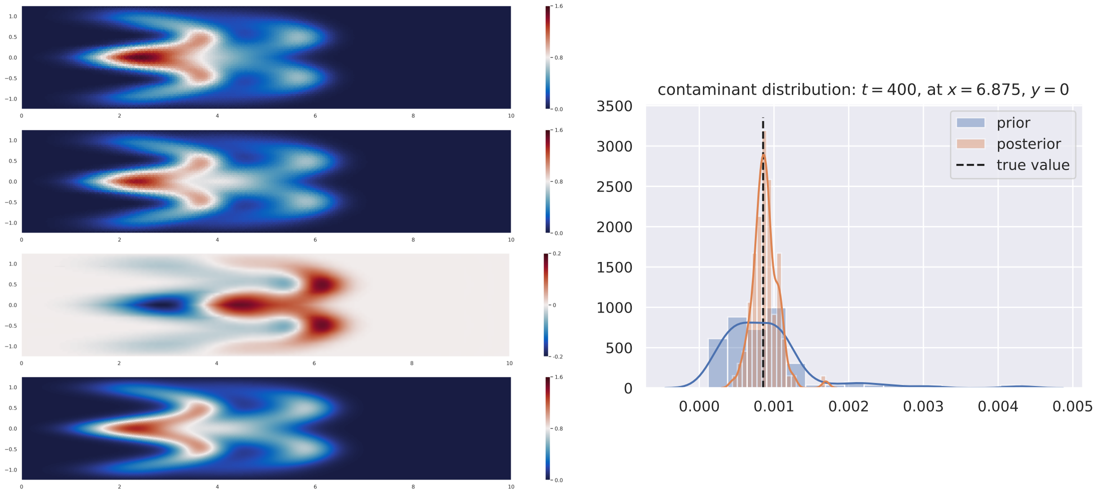


Figure 4-6: [Left] (top) coarse realization before assimilation update; (second) coarse realization after assimilation update; (third) difference in coarse field realization due to assimilation; (bottom) "truth" at $t = 400$. [Right] Prior and posterior contaminant distribution at a central sensor.

By examining the difference plot, we observe a key shift in the less-concentrated portion of the flow beyond the $x = 4$ mark along the center-line. Specifically, it has moved closer to the more-concentrated truth, a clear sign of the simulation's successful adaptation towards the ground truth following the EnKF update. A closer look at the histogram plot of contaminant distributions offers a more quantitative perspective on how the posterior ensemble members have been realigned towards the true value; this lends quantitative credence to the effectiveness of the single EnKF update but also provides a more precise understanding of its impact on the model. The extent to which the EnKF update has improved the accuracy of the simulation is clear, bringing the entire ensemble significantly closer to the true values.

It is illustrative to carry out the same procedure but at an earlier time, at $t = 210$, 10 time steps after the second injection of contaminant. As the flow is only starting to evolve, the uncertainty is highly localized and many sensors are in complete agreement between the ground truth and ensemble value. Figure 4-7 shows the ensemble mean fields with and without data assimilation as compared to the ground truth. The most dramatic alteration to the mean field occurs between the third and fourth center-line sensors, which taper and extend the concentration band, respectively. However, it's also the case that in between sensors, where data isn't available, the effects of the assimilation are much less pronounced on the concentration features, as expected.

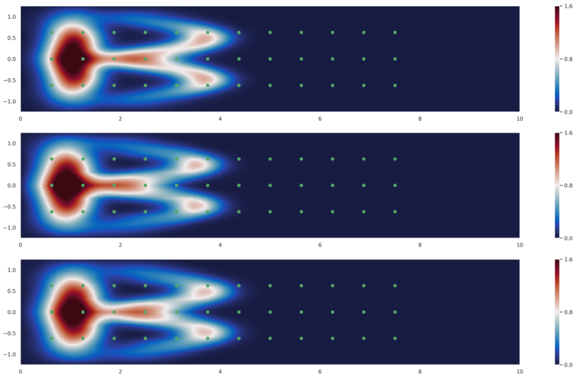


Figure 4-7: Single assimilation step at $t = 210$ using sparse sensors. (Top) truth; (Center) coarse deterministic solution, no data assimilation; (Bottom) assimilated coarse solution.

4.2.5.2 Repeated assimilation in time

To extend the approach from a single assimilation to regular assimilative updates, we consider the two different numerical ensemble resolutions (coarse and very coarse), with assimilation updates every 10 time steps. Comparisons between the non-assimilative coarse run and an EnKF run are given in Figure 4-8 at one of the domain sensors in the contaminant injection region. From the figure, it is clear that the data-assimilative run is able to track the ground truth well, both in terms of the mean field and the ensemble realization. We can see that the EnKF mean field tracks the truth well, and that the realizations are adjusted the most around the re-injection of the contaminant. The non-data assimilative deterministic run, on the other hand, diverges from the truth, with errors compounding with each additional contaminant injection.

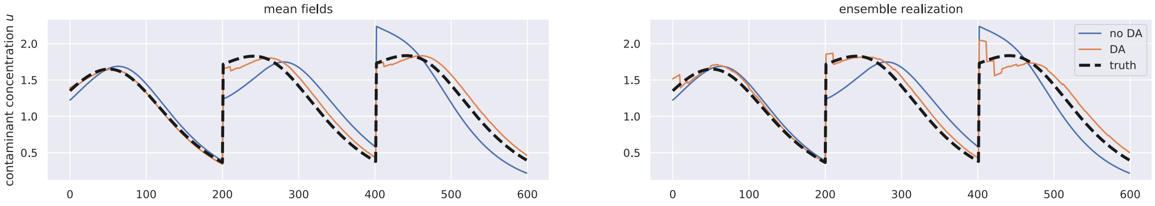


Figure 4-8: Comparison of sensor data at $x = 1.25$, $y = 0$ for coarse runs with data assimilation every 10 time steps and the deterministic run. (Left) mean fields; (Right) fields for a particular ensemble member.

However, as an important caveat, consider the very coarse resolution run, which diverges quickly from the truth and exhibits qualitatively different flow features (see Figure 4-5) due to severe under-resolution. Extensive numerical testing from this run demonstrates that the EnKF is not nearly as effective at tracking the solution as compared to the coarse run, even with frequent assimilations in time. This is unsurprising: a crucial assumption behind the EnKF is that the ensemble is large and varied enough that it contains the statistical features we wish to track. However, the very coarse ensemble members are translated and perturbed, but are all chronically under-resolved. As a result, there are no ensemble members which exhibit the true features of the flow, and so the EnKF algorithm corrects the locations such that the contaminant coincides with the correct initial conditions, but can not dramatically update qualitative properties of the ensemble, as the uncertainties are quite large. This is, of course, correct behavior; as the uncertainty in the grows too large, the EnKF algorithm correctly affects the flow features less. This idea is discussed at length in [72].

Instead, consider the alternate approach of ensembles of order $p = 5$ launched on the very coarse mesh with a much more uncertain initial condition, where we perturb the injection center according to $\Delta \boldsymbol{x} \sim \mathcal{N}([-0.1, 0], 0.1I)$. In this case the ensembles are able to represent the features of the flow, but are subject to greater uncertainty. In this case, again using assimilation every 10 time steps, we are able to recover the true behavior of the contaminant with regular assimilative updates. In this case, although the mesh is still very coarse, we have effectively increased the resolution by increasing the polynomial order of the solution, allowing the ensemble to represent the features of the flow.

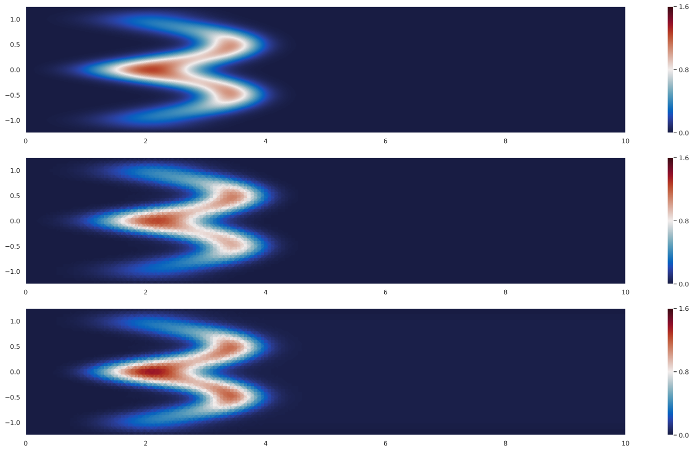


Figure 4-9: Mean field solution at $t = 180$. (Top) truth; (Middle) No data assimilation; (Bottom) Data assimilation.

For the very coarse mesh but polynomial order $p = 5$, Figure 4-9 shows the mean fields of ensembles with and without assimilation. Note that the un-assimilated mean field has had the flow features affected by the initial condition uncertainty. However, the assimilated field is able to recover the symmetric features of the truth. Figure 4-10 shows the contaminant concentration of the assimilated and non-assimilated first ensemble member at two central sensors. Note that although the tracking oscillates at the first uncertain injection, the assimilated ensembles find and are able to track the true contaminant concentration well. Lastly, if we examine the assimilated data for all times at all center-line sensors, we can see that the ensemble mean field with EnKF updates does a very good job at tracking the solution. This can be seen in Figure 4-12 Similar performance can be seen at the non-center-line sensors (not shown). Altogether, this speaks to the effectiveness of high-order methods. As long as the flow features are able to be resolved, performing assimilation over few elements at high order can track the true solution, even with a discontinuous solution representation.

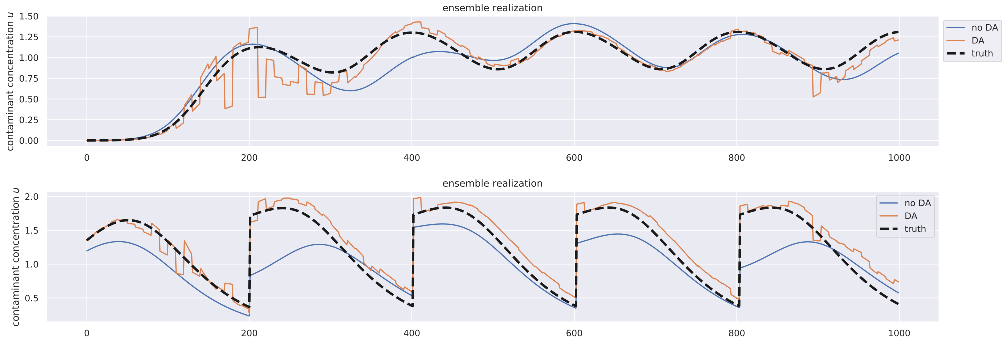


Figure 4-10: Randomly chosen ensemble realizations with and without data assimilation (very coarse, $p = 5$). Shown at two center-line locations ($x = 1.25$ and $x = 3.75$).

4.2.6 Sensitivity to measurement noise

In practice, the choice of σ_ε^2 is important in maintaining the tracking properties of the EnKF. If the variance is too small, the posterior ensemble will be very tightly clustered around the observation, as in the first plot of Figure 4-11. In that case, the first assimilation will be effective, but subsequent observations will lie outside the support of the posterior, and the filter will not successfully track new observations, shown in the second plot of Figure 4-11 for a single realization. Any agreement with the truth after the first assimilation is coincidental; we see that the solution does not jump at the assimilation steps after the first two assimilations. If σ_ε^2 is increased, the ensemble richness is preserved and we recover good tracking, as in the last plot of Figure 4-11. Note the discontinuous jumps as the assimilations continue to be effective through the first period of the problem.

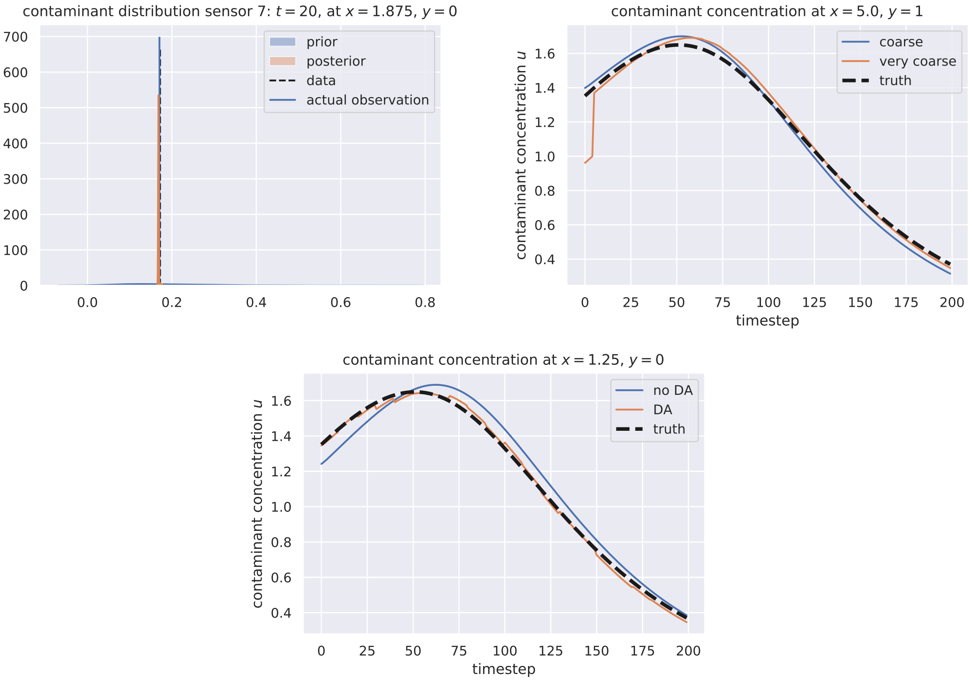


Figure 4-11: [Top] (Left) collapsed posterior distribution after single assimilation, $\sigma_\varepsilon^2 = 10^{-5}$; (Right) contaminant concentration at downstream sensor $\sigma_\varepsilon^2 = 10^{-5}$, assimilation every 10 timesteps; [Bottom] contaminant concentration at downstream sensor, $\sigma_\varepsilon^2 = 10^{-1}$, assimilation every 10 timesteps.

4.2.7 Summary

We have shown that the EnKF can be an effective method to track the physical behavior of a system from an ensemble of realizations representing the uncertainty. We have shown that if the true behavior features exist within the support of the ensemble, the EnKF will work well, whether the mesh is refined with respect to element size or polynomial order. On the other hand, if all ensemble members are not well-resolved or do not contain accurate statistics of the true physical behavior, the EnKF will track the ensemble members which are the best representatives of the observed system (those members with initial conditions and perturbations closest to the true initial state of the system), but will not be able to dramatically alter the features of the ensemble members. We found that the sensor noise can not be too small, because it collapses the ensemble posterior and prevents future observations from being regarded. Lastly, we found that EnKF updates can be performed natively over discontinuous spaces, even with duplicated sensor data, agreeing with the results in [205]. Overall, these findings suggest that the same properties that make DG-FEM methods stable and robust solvers capable of preserving feature fidelity in advection-dominated regimes contribute to their attractiveness in data-assimilative models.

Future topics to investigate include the addition of variance inflation to deal with very accurate sensors, and tracking system behavior with an ensemble of solutions with varying resolution. Secondly, it's possible to perform the data assimilation over the HDG edge space directly, however, this is the subject of ongoing research in the field, similar to that of that of multigrid methods that consider the traced edge space solution specifically.

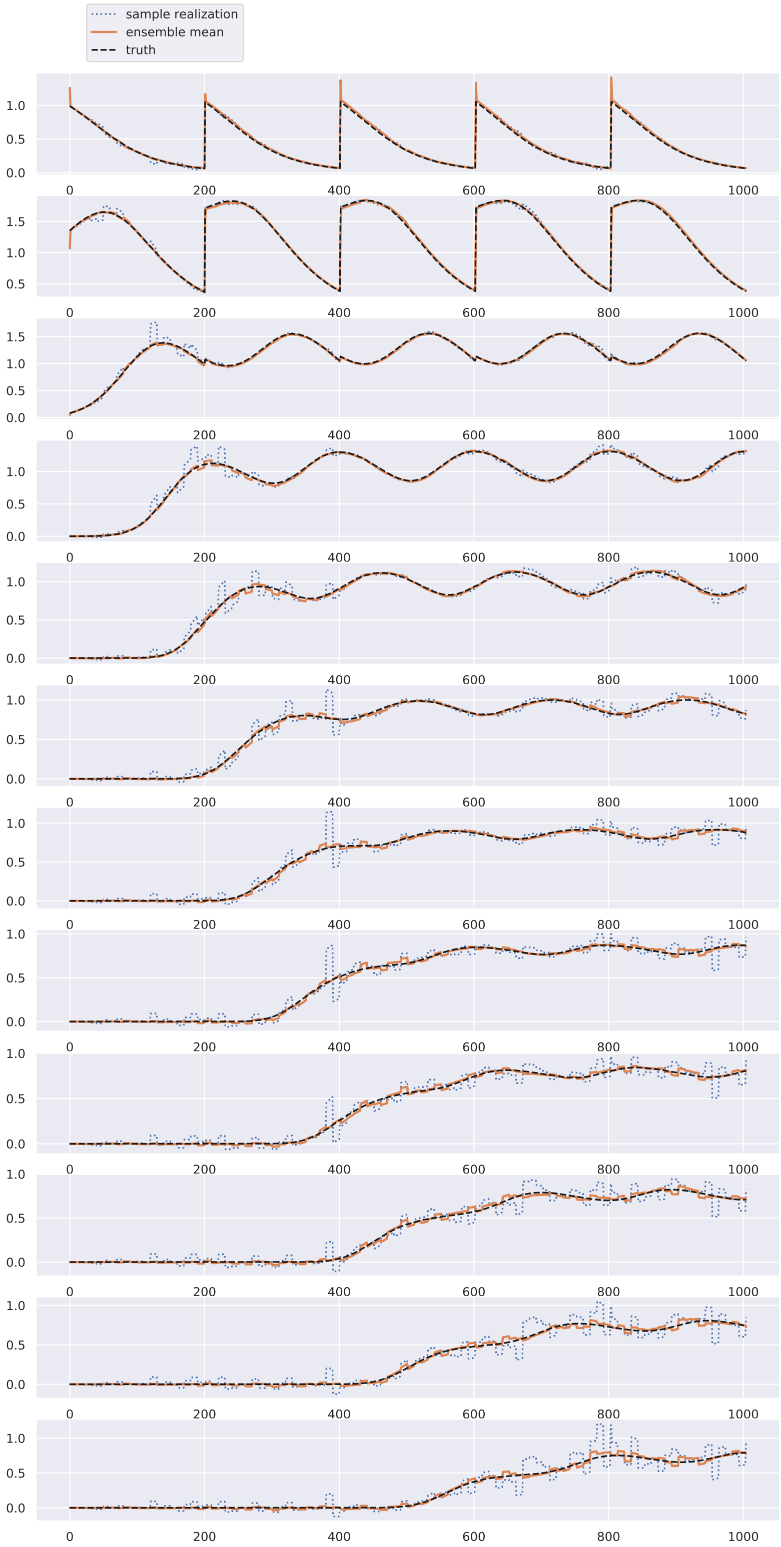


Figure 4-12: Contaminant concentration at centerline sensors (data assimilation every 10 time steps) over the complete simulation.

4.3 Bayesian inversion of coefficients in partial differential equations with HDG fields

An inverse problem involves calculating causal factors from a set of observations. In this work, the problem of interest is to infer a spatially-variable thermal diffusivity field from observations of the temperature field. Such a situation might arise in practice while performing diagnostic testing on an experimental material. As an illustrative example, take the case of designing a space shuttle: atmospheric re-entry might affect the thermal conductivity of a performance-critical shuttle part in a non-uniform way due to the part's geometry, requiring thermal stress testing. After the test, degradation of thermal conductivity can not be measured directly, as it's a non-uniform molecular property of the material. However, we could instead apply different temperatures or heat fluxes to the boundary of the stress-tested prototype and measure the resulting temperature over the prototype interior with an infrared thermometer. From the temperature measurements, we would like to infer the degraded thermal diffusivity field, allowing us to recover the thermal conductivity by direct proportionality.

Summary statistics of the inferred thermal conductivity would allow for better engineering design, namely, judicious reinforcement of the part for robustness. Since the part is performance-critical, "judicious" regions could refer to those of severe degradation or high uncertainty: both are important from a risk-mitigation perspective.

Temperature and thermal diffusivity share an intrinsic relationship expressed through the heat equation, a type of elliptic partial differential equation. The PDE encodes the physics of conservation of energy, and the relationship between diffusivity and observed temperature is not visually obvious. To provide a clearer illustration, Figure 4-13 showcases temperature fields derived from two distinct scenarios: one featuring a spatially constant diffusivity field and the other a spatially variable one. Interestingly, even though the thermal diffusivities differ widely between the two cases, these variations do not leave any easily identifiable patterns or signatures on the resulting temperature field, motivating the problem of their inference.

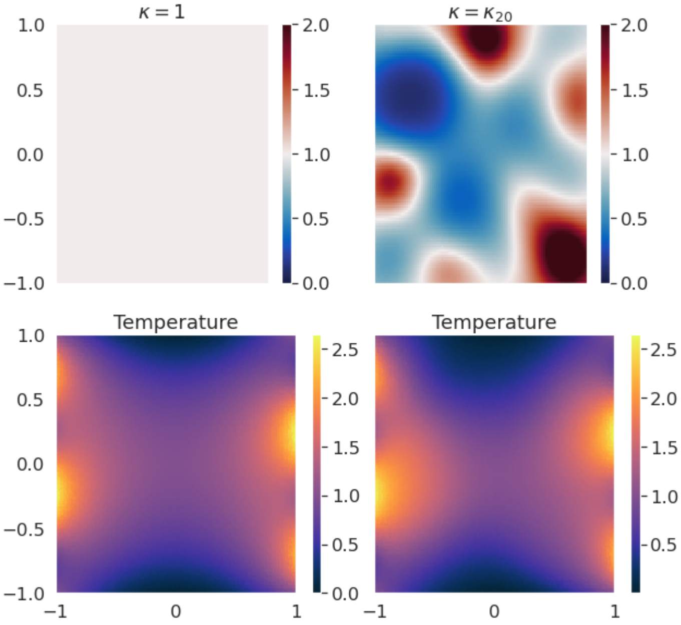


Figure 4-13: Temperature fields generated from different thermal diffusivity fields (left:constant, right:spatially variable) and identical boundary conditions.

4.3.1 Methodology

4.3.1.1 Governing equations and numerical solution

We model the physics of the problem with the heat equation on the spatial domain $\Omega \subset \mathbf{R}^n$, subject to conditions on the domain boundary $\Gamma = \Gamma_D \cup \Gamma_N$. The stochastic PDE is

$$\begin{aligned} -\nabla \cdot (\kappa(\boldsymbol{x}, \omega)\, \nabla u(\boldsymbol{x}, \omega)) = f \qquad & \text{in } \Omega, \\ u = g_D \quad & \text{on } \Gamma_D, \\ \kappa \nabla u \cdot \boldsymbol{n} = g_N \quad & \text{on } \Gamma_N, \end{aligned} \tag{4.17}$$

where $u(\boldsymbol{x})$ denotes the temperature at the point $\boldsymbol{x}$ and $\kappa(\boldsymbol{x})$ is the spatially-variable thermal diffusivity coefficient of the medium. The function $f(\boldsymbol{x})$ is a heat source or sink term, which in this case, we take to be zero. The Dirichlet boundary values g_D correspond to a deterministic temperature imposed on the boundary Γ_D, and Neumann boundary conditions correspond to a deterministic imposed heat flux on the boundary Γ_N.

To simulate the generation of temperature observations, we discretize the domain and numerically solve the PDE in equation (4.17) subject to the boundary conditions of our choosing. Therefore our forward model can be written as

$$u = \mathcal{M}(\kappa, g), \tag{4.18}$$

where the operator $\mathcal{M}$ represents the numerical solution of the heat equation, returning the temperature field $u(\boldsymbol{x})$ for a given realization of the diffusivity field $\kappa(\omega)$, and the boundary data $g = (g_D, g_N)$.

4.3.1.2 Reduced-order representation of random fields

The inverse problem targets the continuous thermal diffusivity field κ, which varies over the entire computational domain. We model the log-diffusivity $\log(\kappa)$ as a Gaussian process in our spatial domain. This makes physical sense since we assume the conductivity to be strongly spatially correlated; physically, we expect $\kappa(x)$ to be close to $\kappa(x + \Delta x)$. The choice of re-parametrization to log-diffusivity enforces the positivity of thermal diffusivity.

Gaussian processes have an infinite number of 'parameters.' Since it is not numerically possible to infer the conductivity at every single point in the domain, we need to find a way to make it possible to reduce the dimensionality of the problem. One approach would be to infer the values of conductivity on a grid - the same grid which will them be used to numerically solve for the temperature field. The dimensionality of the problem is still very high and the problem is ill-posed in additional to computationally intractable. The fact the values of parameters that are spatially correlated to each other can be used to decompose the field and reduce the dimensionality of the problem. We reduce the problem of estimating a large number of highly correlated parameters to estimating a smaller number uncorrelated parameters using a Karhunen-Loève (KL) expansion [87] of the log-diffusivity field. We use a truncated KL expansion to furnish a finite representation of the infinite-dimensional field.

$$\kappa(\boldsymbol{x}, \omega) = \exp(Y), \quad Y = \sum_{i=1}^{N_{\mathrm{KL}}} z_i(\omega) \sqrt{\lambda_i} \psi_i(\boldsymbol{x}), \tag{4.19}$$

where λ_i and ψ_i are the eigenvalues and eigenfunctions, respectively, of the covariance kernel $C(\boldsymbol{x}, \boldsymbol{x}')$

$$\int_\Omega C\left(\boldsymbol{x}, \boldsymbol{x}'\right) \psi_i\left(\boldsymbol{x}'\right)\, d\boldsymbol{x}' = \lambda_i \psi_i(\boldsymbol{x}). \tag{4.20}$$

Instead of inferring the diffusivity in the domain directly, we will infer the KL coefficients z_i in the field, which we will use to reconstruct the diffusivity field over the entire domain.

4.3.1.3 Bayesian inference

We wish to infer the vector of KL coefficients $\boldsymbol{z} \in \mathbf{R}^{N_{\mathrm{KL}}}$ defined such that $\boldsymbol{z}_i = z_i(\omega)$, given temperature measurement data $\mathbf{d}$ polluted by sensor noise.

We represent the log-diffusivity as a random field with a stationary Gaussian process prior

$$Y \sim \mathcal{GP}(\mu_Y, C),$$

Where we take $\mu_Y = 0$. The components are independent under the Gaussian process, with $z_i \sim \mathcal{N}(0, 1)$, yielding the prior likelihood

$$\pi(\boldsymbol{z}) = \prod_{i=1}^{N_{\mathrm{KL}}} \exp\left(-\frac{1}{2} z_i^2\right)$$

Since we have complete trust in the PDE (4.17) to represent the underlying physics, we presume independent additive errors account for the difference between predicted and observed data $\mathbf{d}$. Therefore our model for the data is

$$\mathbf{d} = \mathcal{M}(\boldsymbol{z}) + \boldsymbol{\eta},$$

where each component η_i are i.i.d. Gaussian random variables representing sensor noise, with density $p_\eta \sim \mathcal{N}(0, \sigma_\epsilon^2)$. yielding the likelihood

$$\pi\left(\mathbf{d}|\boldsymbol{z}\right) = \prod_{j=1}^{N_{\mathrm{obs}}} p_\eta\left(\mathsf{d}_j - u_j(\boldsymbol{z})\right)$$

where $u_j(\boldsymbol{z})$ are the temperature outputs of the forward model $\mathcal{M}(\boldsymbol{z})$ at the location of sensor j.

We use Bayes' rule to form the posterior probability density

$$\pi\left(\boldsymbol{z}|\mathbf{d}\right) \propto \pi\left(\mathbf{d}|\boldsymbol{z}\right) \pi(\boldsymbol{z}) \tag{4.21}$$

While we could have instead chosen a completely uninformative prior or encoded uncertainty within the forward model, our choices reflect our beliefs. Namely, we encode the physical notion of smoothness in our prior and our trust in the underlying physical model with the likelihood.

4.3.2 Implementation

4.3.2.1 Forward model

We evaluate the forward model by solving the PDE (4.17) numerically, using a hybridizable discontinuous Galerkin finite element scheme [191] implemented as in 2. An advantage of using high-order finite elements is that we can solve the problem accurately on a coarse computational mesh.

To verify the forward model $\mathcal{M}$, we constructed an example where the source term f and inhomogeneous boundary conditions g_D were chosen such that the exact solution with a manufactured anisotropic, spatially variable thermal diffusivity field $\kappa(\boldsymbol{x}) = [y + 1, x + 2]$ is

$$u(\boldsymbol{x}) = x^2 - (y-1)^2 \quad \text{on } \Omega \tag{4.22}$$

We solved the problem numerically using the forward model at polynomial order $p = 4$. Since the forward model is a finite-element implementation, the numerical solution should be exact, because u is chosen to be polynomial. We recovered the exact solution to machine precision, verifying the correctness of the forward model.

4.3.2.2 Computation of KL modes

The Nyström method was used to solve the eigenvalue problem (4.20), furnishing the truncated KL expansion in (4.19). The integral operators were approximated with high-order quadrature schemes [230].

While the Nyström method is agnostic to choice of kernel, we verified the implementation with a squared-exponential kernel and reconstructed a manufactured log-diffusivity field by sampling the KL-coefficients. The point-wise distribution of log-diffusivity is Gaussian and the covariance of the fields at randomly chosen locations matches that of the covariance kernel—see Figure 4-14.

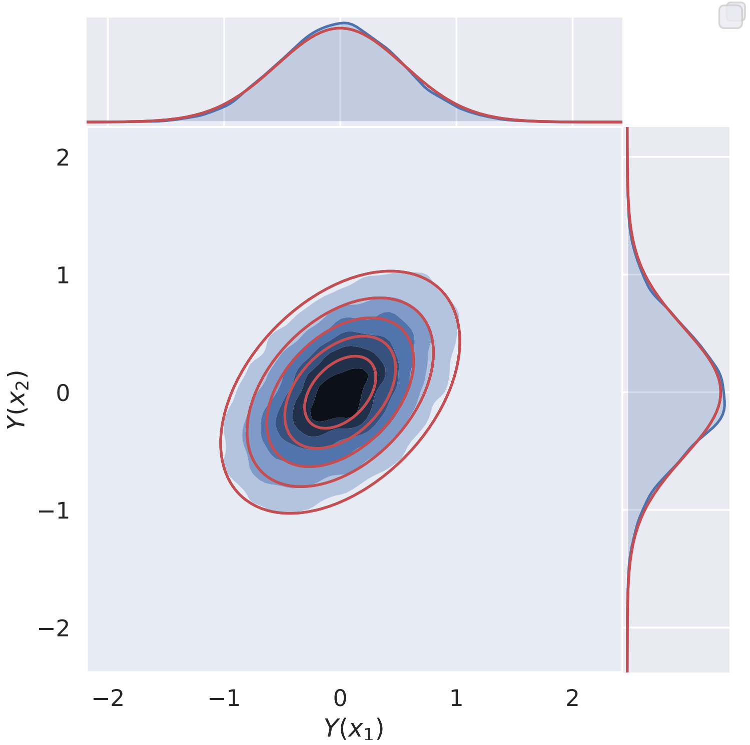


Figure 4-14: Verification of KL expansion: the blue regions correspond to the draws from the truncated KL expansion and the red contours show the expected distribution of a manufactured log-diffusivity field Y given the covariance kernel.

4.3.2.3 MCMC sampling from the posterior

We use MCMC to sample from the posterior distribution in (4.21). While the truncated KL representation of the diffusivity field dramatically reduces the dimension of the parameter space, each MCMC step requires an expensive evaluation of the forward model $\mathcal{M}(\boldsymbol{z})$ for Metropolis sampling, as well as its gradient with respect to the parameters $\nabla_{\boldsymbol{z}}\mathcal{M}(\boldsymbol{z})$ for a NUTS sampler.

We chose to implement our inference problem in the probabilistic programming library PyMC3 [226] library. We implemented a custom likelihood using the output of the forward model in Theano, which allowed us to leverage the PyMC3 library for MCMC.

We also implemented the gradient of the forward model $\nabla_z\mathcal{M}(\boldsymbol{z})$ numerically with a finite difference scheme. Doing so allowed us to obtain an accurate approximation of the gradient non-intrusively, but at the cost of N_{KL} function evaluations at every MCMC step, one for each parameter dimension. Ideally, we would have evaluated the gradient with automatic differentiation, but no open source library was capable of doing so non-intrusively with respect to the complicated forward model.

4.3.3 Numerical experiments

4.3.3.1 Setup

We benchmark the performance of the samplers by setting a ground truth for each diffusivity field by selecting ‘true’ values of the KL coefficients $\boldsymbol{z}_{true}$ to be inferred. We then pass it through the forward model to get the true temperature field for the ground truth. The values of the 'true' temperature field at certain sensor locations are the observed data used for inference.

In the following experiments we compare MCMC samplers: adaptive Metropolis-Hastings (MH) and the No-U-Turn sampler (NUTS). While we tested other options, including slice sampling and Hamiltonian Monte Carlo, we found that the MH and NUTS samplers were the primary competitors in terms of their ability to solve the problem, as well as the two most interesting to compare, since MH does not require a gradient of the likelihood with respect to the model parameters, unlike NUTS.

For our experiments, we considered test cases consisting of inferring three separate diffusivity fields generated with 5, 10, and 20 KL modes. In each test case, we compared the posterior distributions returned by the MH and NUTS samplers. As discussed above, the temperature observations were generated from the forward model evaluated on the ground truth diffusivity field κ, generated using $\boldsymbol{z}_{\text{true}}$. Doing so allowed us to compare the mean inferred diffusivity field to an analytical solution.

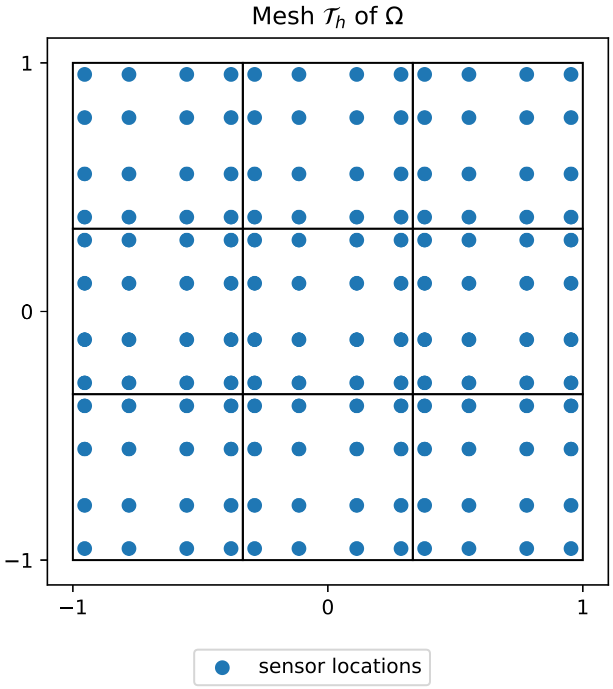


Figure 4-15: Computational mesh $\mathcal{T}_h$ used in the numerical experiments, and the sensor locations.

For all experiments, we used a coarse numerical solution of the forward model, with a computational mesh consisting of nine quadrilateral finite elements of polynomial order $p = 3$, shown in Figure 4-15. Temperature measurements u_{obs} were generated at $N_{\text{obs}} = 144$ sensor locations over the domain. We remark that while the quality of the inferred diffusivity field will depend on the number of observations, the number of sensors does not appreciably change computation time, as the vast majority of time is spent evaluating the forward model. Therefore, we simply chose enough sensors to adequately cover the length scales of the diffusivity field for all test cases. In practice, more measurements could be taken to improve the mean-field estimates; conversely, sparser temperature measurements would, correctly, result in worse mean-field estimates and greater uncertainty over the parameters.

Each MCMC result shown is a combination of four chains run in parallel on separate cores. The number of MCMC steps always refers to the number of steps per single chain. For the purposes of analysis and diagnostics, we sometimes combine the results of all four chains into an aggregate set of samples, in such cases, we will refer to total samples rather than iterations or steps.

4.3.3.2 5 KL modes

We found that both samplers were adequately able to solve the problem after 10^4 MCMC steps.

A visual comparison of the true diffusivity field and the inferred diffusivity field is shown in Figure 4-16 for the MH sampler. The results are nearly identical for the NUTS posterior. The mean inferred diffusivity qualitatively matches the actual diffusivity field quite well. Furthermore, examining the difference between the inferred field and the true diffusivity reveals that the posterior correctly assigns uncertainty to the regions where the diffusivity field most differs from the truth.

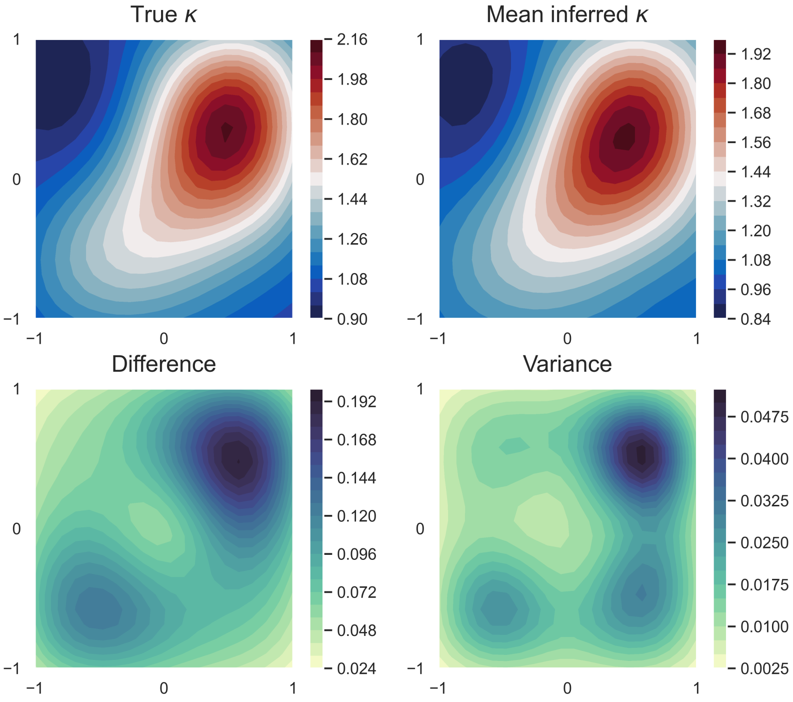


Figure 4-16: Mean-field inferred diffusivity from MH posterior.

The posterior distributions over the coefficients z_i are shown in Figure 4-17. We see that the posteriors returned by each sampler are very similar to each other, both in terms of mean and

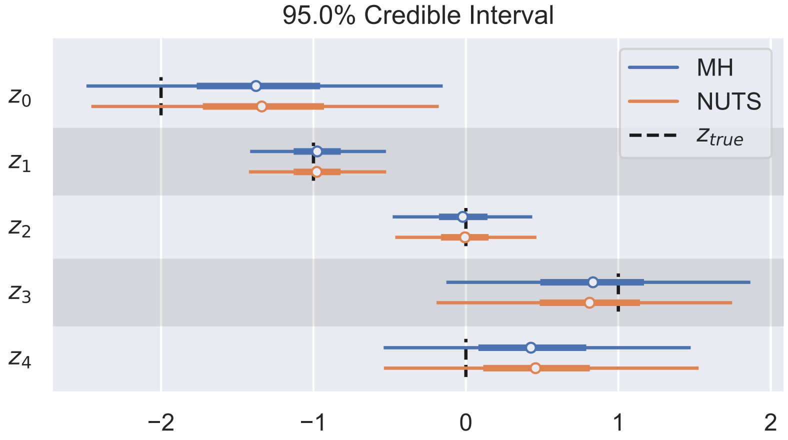


Figure 4-17: Posterior credible intervals over coefficients z_i, for MH and NUTS MCMC, along with generative coefficients z_{true} for the true diffusivity field.

variance. The actual coefficients used to generate the true diffusivity field are all within the 95% credible interval of all posteriors, and importantly, the uncertainties reflect the accuracy of the means.

The traces for the MCMC chains plotted in Figure 4-18 show good mixing for MH, with very similar results for NUTS. Neither sampler was plagued by a high rejection rate or an inability to explore the posterior space.

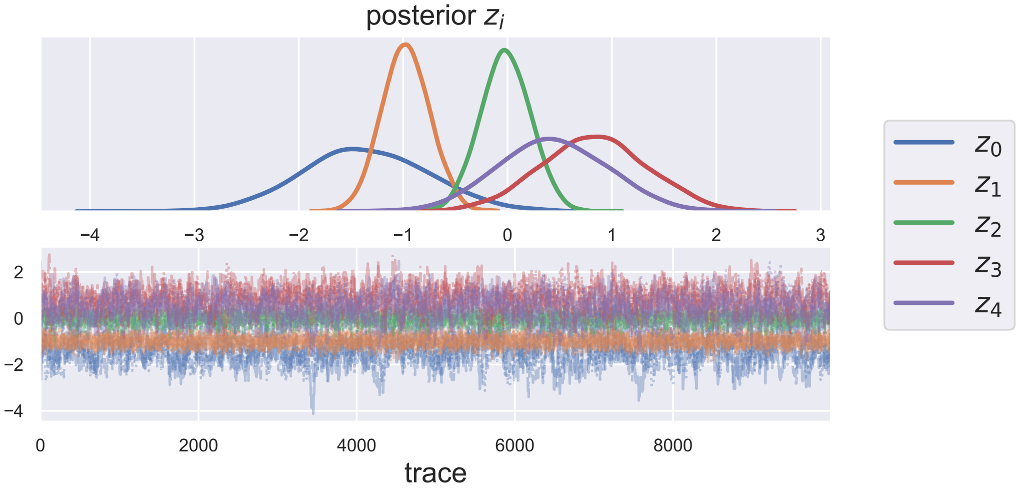


Figure 4-18: Posterior distributions and traces for MH MCMC.

	MH		NUTS	
z_i	ESS	$\widehat{r}$	ESS	$\widehat{r}$
0	467.0	1.01	285.0	1.02
1	3887.0	1.00	650.0	1.01
2	3434.0	1.00	727.0	1.00
3	935.0	1.00	406.0	1.01
4	479.0	1.01	299.0	1.02

Table 4.1: Effective sample sizes (ESS) and $\widehat{r}$ parameters for each of the MCMC samplers, $N_{KL} = 5$.

More closely comparing the two samplers by examining the MCMC diagnostics in Table 4.1, we see that both have values of $\hat{r} \approx 1$, indicating that each of the combined four chains have converged to a common distribution. We remark that this heuristic is a tool to detect non-convergence rather than convergence and good $\widehat{r}$ values do not necessarily guarantee that the posterior space has been completely explored. The effective sample sizes are larger for the MH sampler, indicating a better estimation error and autocorrelation properties as compared to NUTS.

The primary difference between MH and NUTS was represented in the computational time required to solve the problem. The error in the mean posterior field as measured in the L^2-norm over the domain is depicted as a function of total samples and wall clock time in Figure 4-19. In our examination, we found that both the Metropolis-Hastings and the NUTS sampler deliver similar levels of accuracy in their results. However, despite their comparable accuracy, a stark contrast was noted in the runtime of these two methods. The MH algorithm was markedly faster, taking approximately 15 minutes to complete its run. This relatively swift operation time stands as a testament to the efficiency inherent in the MH sampler, which is designed to expedite the sampling process without compromising the accuracy of its results. On the other hand, the NUTS algorithm proved to be significantly slower. It recorded a runtime of approximately 27 hours, which, when compared to the 15-minute runtime of the MH, represents a significant delay. To put it in perspective, the runtimes of the MH and NUTS algorithms differ by a factor of approximately 122. In other

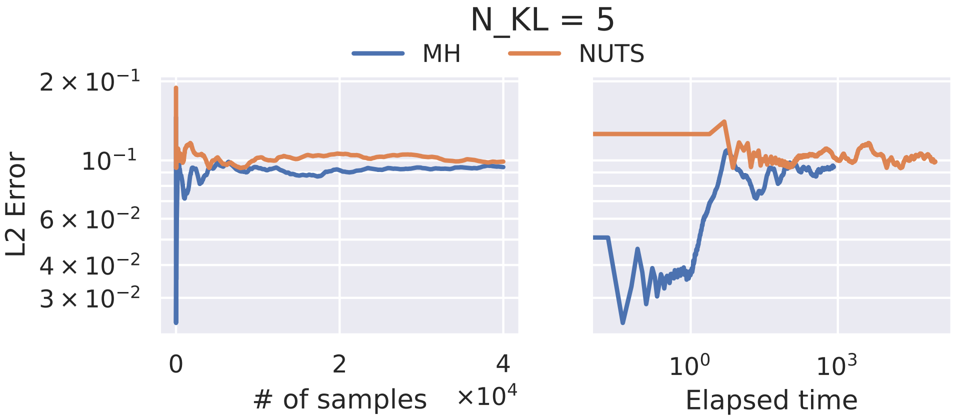


Figure 4-19: Error in mean posterior field as measured in the L^2-norm over the domain as a function of total number of MCMC samples completed (left) and elapsed wall clock time in seconds (right).

words, the NUTS algorithm took over two orders of magnitude longer to run than the MH algorithm. This disparity underscores the trade-offs that often arise when choosing between different sampling methods. While the NUTS sampler is praised for its detailed exploration of the target distribution and has certain benefits in complex statistical situations, it is important to consider the increased computational time when applying it to large scale problems. This juxtaposition between MH and NUTS serves as a tangible reminder of the need to balance accuracy with efficiency when choosing an appropriate sampling method.

4.3.3.3 10 KL modes

For the case where ten KL modes are employed, the time requirements imposed by the NUTS sampler restricted our comparison to 10^4 MCMC steps. Within these constraints, it became clear that only the MH sampler could solve the problem satisfactorily within the provided constraints. This finding suggests that in more complex settings, the exploratory efficiency of MH can surpass that of NUTS. A visual representation of this statement can be found in Figure 4-20, which displays the true diffusivity field κ and the inferred mean diffusivity field. Despite the complexities involved in exploring the higher-dimensional parameter space of ten KL modes, the diagram displays good qualitative agreement between the actual and the inferred mean diffusivity fields. Furthermore, it provides valuable insights into the uncertainty estimates associated with the process, which are corroborated by the mean fields results and reaffirms the robustness of the MH sampler.

Comparing the posterior credible intervals for MH and NUTS shown in Figure 4-21, we see that unlike the $N_{\text{KL}} = 5$ case, the posterior distributions returned by the two samplers show considerable disagreement. The true coefficients $\boldsymbol{z}_{\text{true}}$ are contained in all of the credible intervals returned by the MH sampler, and the mean estimates for most coefficients are quite good. The same can not be said about the estimates from the NUTS sampler, with estimates that often deviate significantly from the true coefficient value; more importantly, in some cases, the true value falls well outside the 95% credible interval.

The MCMC diagnostic statistics provide a clearer picture of what's going wrong with the NUTS sampler. Figure 4-26 shows the distribution of $\widehat{r}$ values for the samplers. The MH sampler has components with $\widehat{r} \leq 1.1$, indicating chains which are potentially approaching convergence. The NUTS sampler, on the other hand, is characterized by $\widehat{r} > 1.1$, indicating the the sampler is nowhere near convergence and that individual samples are still highly dependent, a finding substantiated by examination of both the trace and autocorrelation plots (omitted here).

Again, we consider the solution accuracy of the mean-field estimates as a function of total samples and wall clock time in Figure 4-23. We see that the accuracy of the MH posterior is far better than

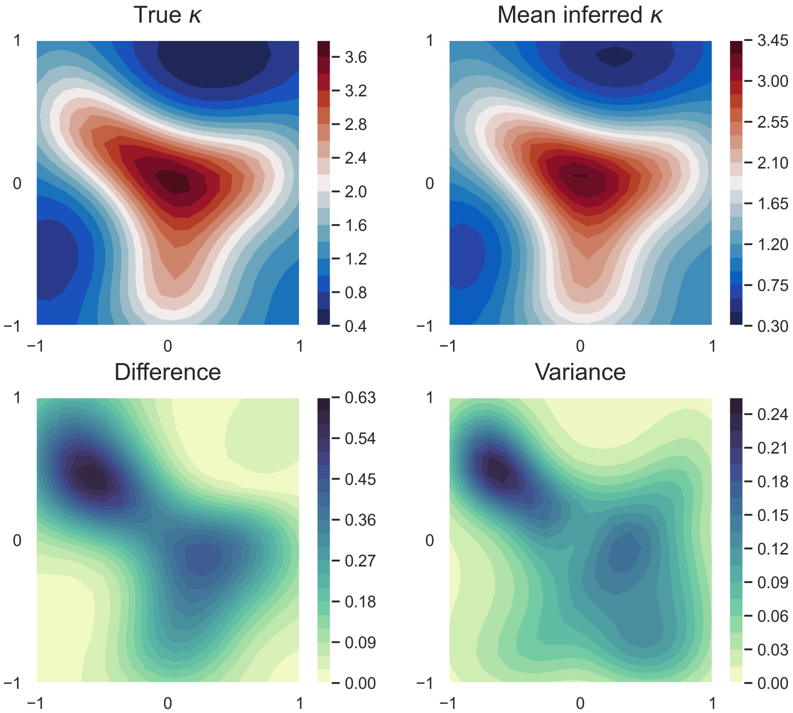


Figure 4-20: Mean-field inferred diffusivity from MH posterior.

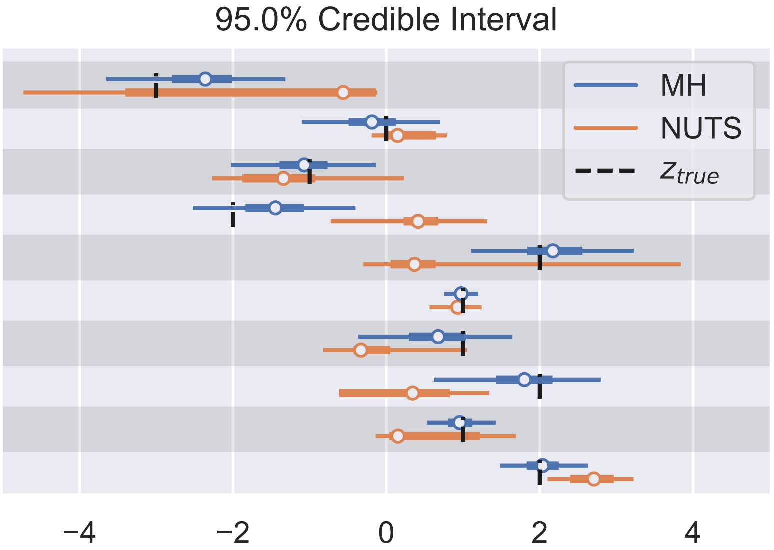


Figure 4-21: Posterior credible intervals over coefficients z_i, for MH and NUTS MCMC after 10^4 steps, along with generative coefficients z_{true} for the true diffusivity field.

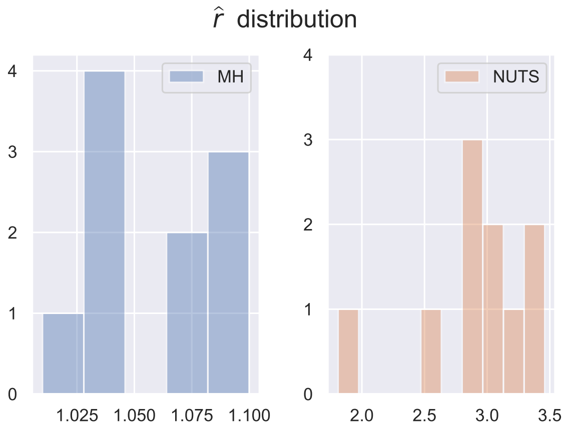


Figure 4-22: Distribution of the $N_{\mathrm{KL}} = 20$ values of $\widehat{r}_i$ for both MCMC samplers after 10^4 steps.

the that returned by the NUTS sampler. Comparison of required wall clock time reveals a similar discrepancy as the five KL mode case; roughly 30 minutes and 45 hours for the MH and NUTS samplers, respectively—roughly a factor of 85 difference between the two.

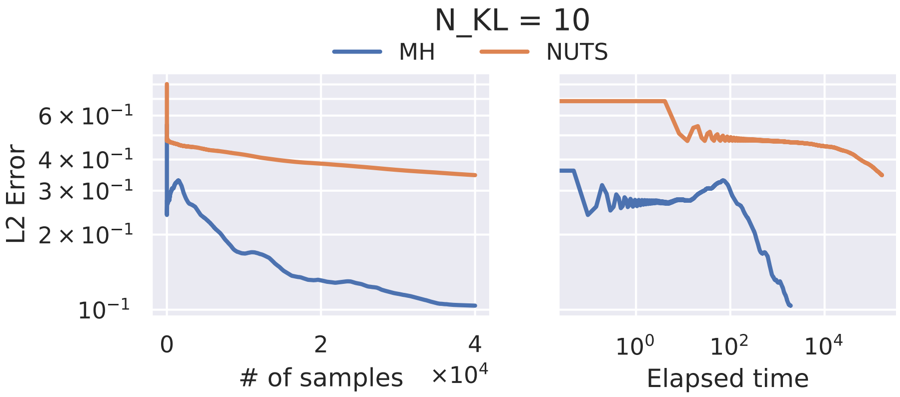


Figure 4-23: Error in mean posterior field as measured in the L^2-norm over the domain as a function of total number of MCMC samples completed (left) and elapsed wall clock time in seconds (right).

The discrepancy between the computational times is responsible for our decision to compare MH and NUTS using only 10^4 MCMC steps. While it's possible that NUTS could have achieved convergence with a small amount of additional steps, running the simulations for longer than a few days was simply not practically feasible.

4.3.3.4 20 KL modes

For the case of 20 KL modes, we once again found that only MH was able to solve the problem, and did so in a fraction of the run time of NUTS.

Figure 4-24 demonstrates the physical agreement between the mean field and the truth. We also see that once again, the MH sampler includes the true mean in the 95% credible interval for all inferred coefficients (Figure 4-25). NUTS, on the other hand, performs much more poorly, giving inaccurate estimates and uncertainties as compared to MH.

The MCMC diagnostics clearly show non-convergence for the NUTS sampler and dependence between samples. For MH, despite the good physical agreement and the capturing of the true values

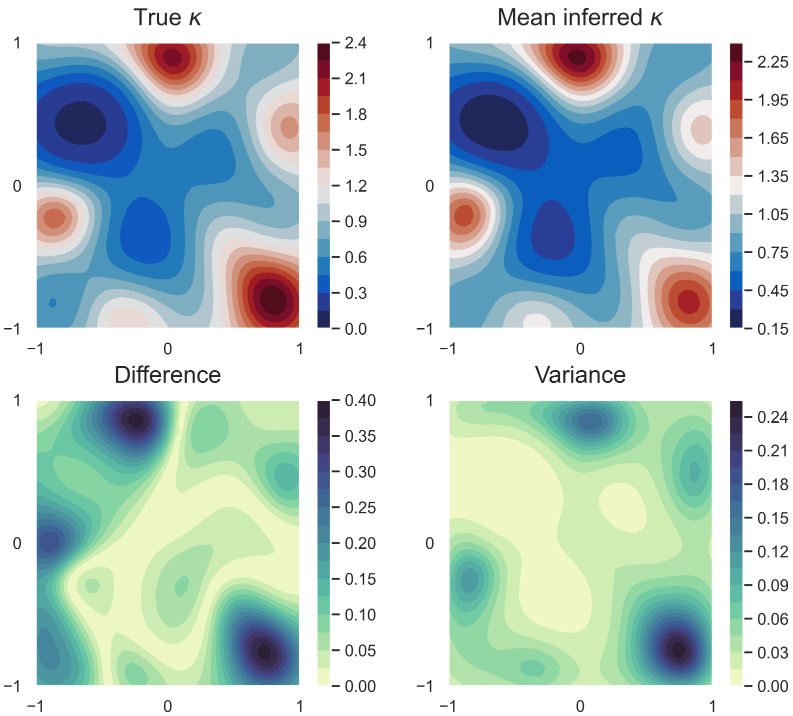


Figure 4-24: Mean-field inferred diffusivity from MH posterior, 10^4 MCMC steps.

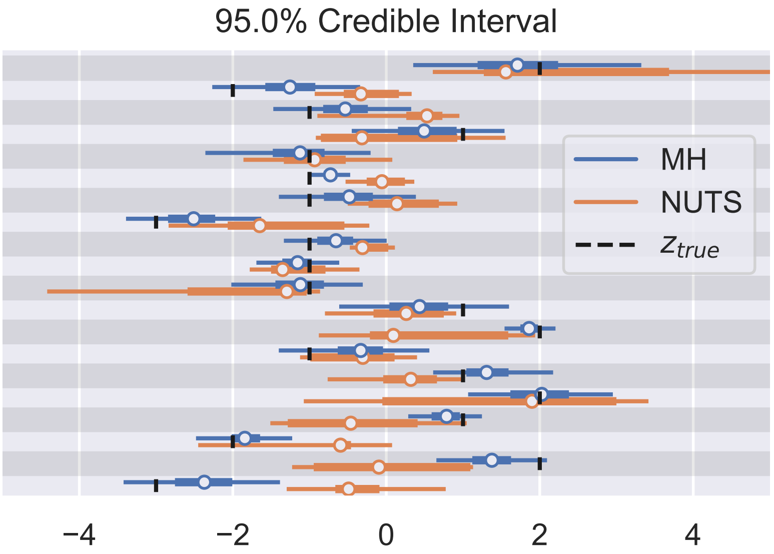


Figure 4-25: Posterior credible intervals over coefficients z_i, for MH and NUTS MCMC after 10^4 steps, along with generative coefficients z_{true} for the true diffusivity field.

within the 95% credible intervals, the values of $\widehat{r}$ shown in Figure 4-26 indicate non-convergence for some of the coefficients of $\boldsymbol{z}$, namely, the two with $\widehat{r} > 1.1$.

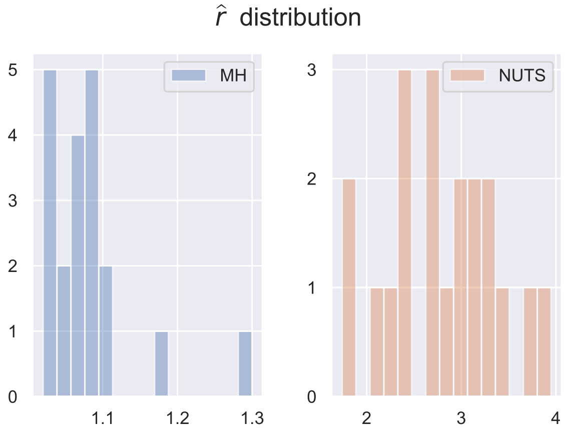


Figure 4-26: Distribution of the $N_{\mathrm{KL}} = 20$ values of $\widehat{r}_i$ for both MCMC samplers after 10^4 steps.

A potential lack of convergence is also corroborated by the error plots (Figure 4-27), where the dropping error in mean estimate seems to imply that the the mean estimate is still improving as we draw more samples. In the 20 KL mode experiment, NUTS performed miserably, showing complete lack of convergence and taking roughly 4 days to run as compared with the approximately 30 minutes required for MH, roughly a factor of 300 difference between the two samplers.

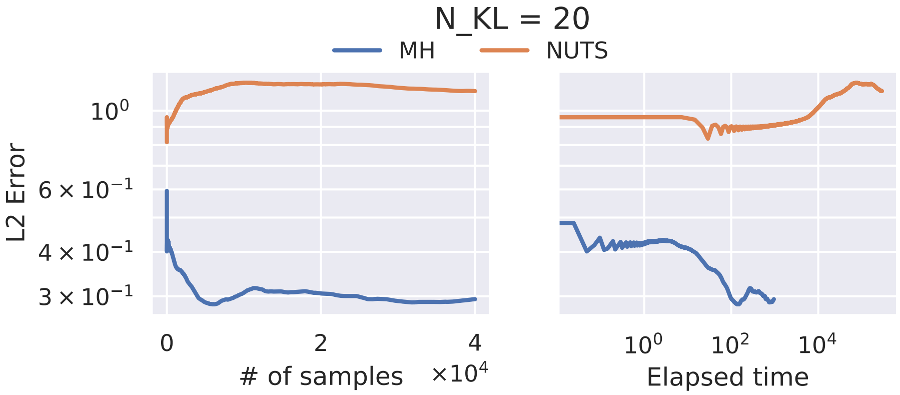


Figure 4-27: Error in mean posterior field as measured in the L^2-norm over the domain as a function of total number of MCMC samples completed (left) and elapsed wall clock time in seconds (right).

Since the MCMC diagnostics suggested that the MH sampler was not quite converged, we ran another longer test with the MH sampler, for 10^5 MCMC steps. The additional MCMC steps resulted in very similar posterior results as compared to the 10^4 MCMC step run, although with improved diagnostic $\widehat{r}$ values and effective sample size, as shown in Figure 4-28, demonstrating that the sampler had nearly converged at the end of the 10^4 steps.

4.3.4 Summary

We were able to successfully solve the proposed inverse problem. For the MCMC sampler, adaptive Metropolis-Hastings summarily outperformed the NUTS sampler, contrary to what one might expect. In all test cases, the adaptive MH sampler was orders of magnitude faster than NUTS and

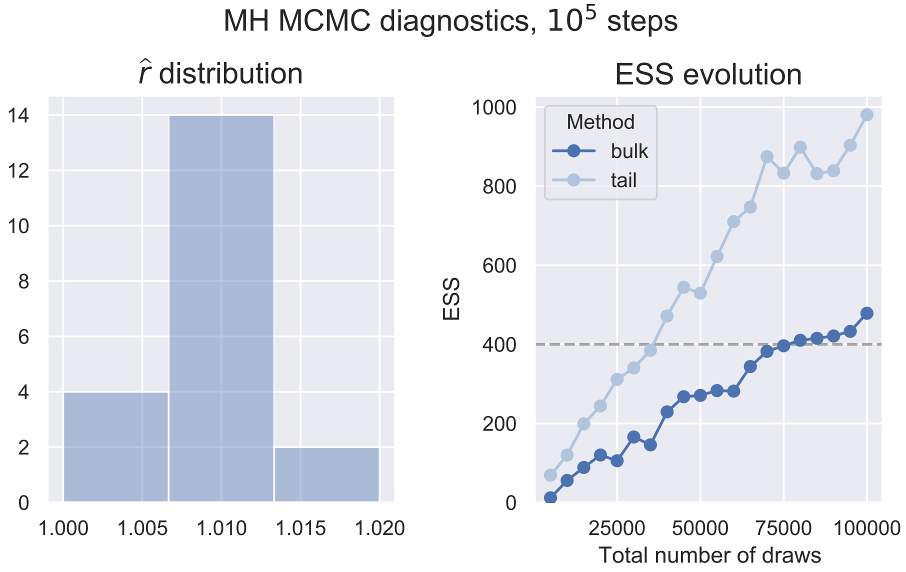


Figure 4-28: Distribution of $\hat{r}$ values and evolution of effective sample size (ESS) for MH after 10^5 MCMC steps.

provided better or equal results in terms of posterior predictive distribution. This is likely the result of the finite difference computations of the likelihood gradient with respect to the inference parameters, since the expensive numerical solution supplied by the forward model must be evaluated once for each dimension of the parameter space at every MCMC step.

It's possible that for much higher-dimensional parameter spaces, that a NUTS sampler would explore the reduced-order posterior space better than the Metropolis sampler, however, the wall-clock time-to-solution was simply too prohibitively expensive to explore a larger number of KL modes with the NUTS sampler. It may also be the case that if the gradient of the likelihood were evaluated with automatic differentiation rather than finite differences that NUTS would be competitive with or outperform the Metropolis sampler, due to the cheaper and more accurate gradient evaluations. However, we found that there were no automatic differentiation libraries that were capable of propagating the gradient through the complicated forward model code non-intrusively.

Our primary conclusion is that in the context of PDE-based inference where the evaluation of the likelihood and its gradient can be very expensive, the state-of-the-art MCMC technologies do not necessarily outperform simpler samplers when exact gradients of the likelihood model are not available. However, our experiments also shed light on the power of reduced-order modeling—inference over a naively discretized diffusivity field would have rendered the inverse problem completely intractable with MCMC. However, by leveraging the KL-expansion and the physical smoothness of the diffusivity field, we can obtain good estimates of the infinite-dimensional parameter fields responsible for physical observations, along with uncertainties—inferences which could prove invaluable during engineering analysis or design.

4.4 Conclusion

We completed a set of computational studies evaluating HDG-based implementations of standard methods in uncertainty quantification: ensemble Kalman filter (EnKF) for data assimilation and Markov Chain Monte Carlo sampling for Bayesian inversion of unknown PDE coefficients. The results of the computational experiments demonstrate that with the DG-FEM-targeted modifications to the standard methods investigated in this chapter, we are able to convincingly solve the uncertainty quantification problems, the high-order, discontinuous representation of the solution fields notwithstanding. These idealized investigations provide insight into managing the complexities introduced by the DG-FEM methods, thereby suggesting the applicability of these methods to real-ocean problems such as the nonhydrostatic model nesting approach developed in §3.

In this chapter, we've established the Ensemble Kalman Filter as a robust approach for data assimilation given an ensemble of realizations that capture the inherent uncertainty. This holds true regardless of whether the mesh is refined by element size or polynomial order. We also confirmed that EnKF updates can be effectively performed over discontinuous spaces, even when dealing with duplicated sensor data, and that the assimilation can be performed efficiently within the HDG framework. Overall, these insights suggest that the qualities that make Discontinuous Galerkin methods (DG-FEM) a stable and robust solver, capable of maintaining feature fidelity in advection-dominated regimes, also make it a desirable choice for data-assimilative models. Looking ahead, potential areas of exploration include introducing variance inflation to handle highly accurate sensors and tracking system behavior using an ensemble of solutions with varying resolution.

With regards to solving inverse problems, we were able to apply a dimensionality reduction technique based on polynomial chaos along with Markov Chain Monte Carlo (MCMC) methods to efficiently sample from the posterior distribution over an unknown coefficient field. Through a set of numerical test cases and benchmarks, the adaptive Metropolis-Hastings sampler markedly outperformed the No-U-Turn Sampler (NUTS). Contrary to initial expectations, the adaptive Metropolis-Hastings sampler proved to be significantly faster than NUTS in all test cases, delivering equal or superior results in terms of the posterior predictive distribution. This unexpected outcome is likely a result of sensitivity of the NUTS sampler to the finite difference computations of the likelihood gradient with respect to the inference parameters. As the expensive numerical solution offered by the forward model must be evaluated once for each MCMC sample, gradient-based updates with an insufficiently accurate gradient can struggle to adequately explore regions of high-probability in parameter space. Overall, we conclude that in the context of PDE-based inference, where evaluating the likelihood and its gradient can be quite costly, state-of-the-art MCMC technologies don't necessarily outperform simpler samplers when exact gradients of the likelihood model aren't available. But our experiments did highlight the power of reduced-order modeling. Inferring over a naively discretized diffusivity field would have made the inverse problem completely intractable with MCMC. However, by utilizing the KL-expansion and acknowledging the physical smoothness of the diffusivity field, we could obtain robust estimates of the infinite-dimensional parameter fields responsible for physical observations, along with their uncertainties. These insights could prove invaluable during engineering analysis or design, and future work could involve implementing HDG solvers in the framework of automatic differentiation libraries to improve the performance of the state-of-the-art samplers.

Chapter 5

Deep Reinforcement Learning for Adaptive Mesh Refinement

5.1 Introduction

In Chapter §3, we developed a high-order discontinuous Galerkin ocean model capable of capturing a wide variety of oceanic phenomena. We found that the scales of internal waves and Rayleigh-Taylor instabilities can be substantially smaller than their hydrostatic counterparts, creating an intricate and complex landscape to navigate and model. This reality presents a distinct opportunity when it comes to accurately modeling nonhydrostatic phenomena, in which the local nature of the nonhydrostatic phenomena could allow for reduction of computational cost. Their scale, coupled with their nonlinear behavior, necessitates novel and more sophisticated methodologies in order to accurately represent them in a model.

These observations motivate the exploration of adaptive mesh refinement techniques, with the aim of their application in realistic high-order nonhydrostatic ocean models. However, the inherent complexity of ocean models poses a barrier to the direct application of standard adaptive mesh refinement techniques. The unique dynamics of these models require solutions that are tailored to their specific dynamics.

Against this backdrop, machine learning approaches may prove fruitful—rather than applying a blanket technique, unsupervised methods could potentially learn effective cost-reduction strategies directly from simulation data. Techniques that allow strategies to be learned experientially from simulation would be even more advantageous, and bypass the necessity of a high-fidelity ground truth dataset. This self-learning approach could provide a far more nuanced and accurate representation of the physical behavior specific to the partial differential equation (PDE) in question. Taking into account these requirements, deep reinforcement learning constitutes an appealing solution to this challenge. This approach not only meets the criteria outlined but also offers the potential for significant advancement in the field. To that end, in this thesis, we have formulated a novel method to employ deep reinforcement learning to learn adaptive mesh refinement strategies directly from simulation. While the ultimate goal is to offer a new avenue for understanding and predicting the ocean's intricate and complex behaviors, the approach can be more generally applied to any problem involving the numerical solution of PDEs.

In recent decades, the finite element community has developed principled, efficient techniques for solving partial differential equations. Not only are these techniques extremely general methods that may be applied to almost any PDE, they also provide guarantees such as stability, consistency, and convergence [32, 110]. On the other hand, the machine learning community has developed a broad set of methods to learn latent patterns from large datasets in the absence of a model [24, 120]. However, for problems in computational physics, it is often the case that the PDE is an excellent model of the underlying physical phenomena, often down to the molecular level, where the continuum assumption begins to lose validity. Attempts to use machine learning to learn solutions to PDEs directly have shown promise for relatively trivial problems, but are as of yet subject to several failure modes in terms of generalization to even moderately more complicated problems [128]. Rather than ignoring the extensive body of work in either field, we propose combining techniques from numerical mathematics and machine learning in order to preserve mathematically hard-earned guarantees while improving accuracy and efficiency by applying machine learning to the peripheral aspects of numerical methods which lack a model and rely purely on heuristics. Adaptive mesh refinement is one such aspect.

Uniform meshes are often computationally inefficient for finite element simulations in that the mesh density required to resolve complex physical features such as steep gradients or small-scale processes is used everywhere over the computational domain, even in regions where the numerical solution is smooth and much coarser resolution could be used. In many problems, such features are dynamic: eddies, meandering jets, or nonlinearly evolving cracks constitute some examples. To optimize efficiency, adaptive mesh refinement (AMR) techniques are a class of methods that dynamically modify the computational mesh during simulation in an attempt to increase resolution specifically where it is needed [210].

Most AMR techniques follow an iterative procedure of numerically solving the PDE, estimating the error on each element, marking a subset of the mesh elements for refinement or de-refinement (coarsening), and executing the alterations to the mesh [31, 58]; this well-established paradigm is commonly referred to as the SOLVE→ESTIMATE→MARK→REFINE loop, terminology we will use in the present chapter. Following each numerical solve, the ESTIMATE process involves estimating the discretization error on each cell in the mesh. In this chapter, we draw the distinction between an *error estimator*, which provides objective measures of error in a specific norm, and an *error indicator*, which offers an empirical indication as to the local magnitude of numerical error but provides no theoretical guarantees. The MARK process selects cells for coarsening and refinement based on the estimates of the error. Two common approaches are bulk refinement and fixed-number refinement [20, 23, 31, 61, 111, 112, 185]. The former marks all the cells responsible for a specified percentage of the (estimated) total error for refinement and similarly for coarsening. The latter refines and coarsens a fixed percentage of the total number of cells.

While AMR methodologies have allowed computational scientists to solve problems which are completely intractable on a uniform mesh [125], the application of AMR strategies remains largely heuristic, and best practices are not universally agreed upon [210]. The process of estimating the error on each cell is often complex, dependent on both the PDE and the numerical method used to solve it, and constitutes an active subject of research [2, 19, 70]. In general, for nonlinear partial differential equations, it is often difficult, and in some cases, impossible to provide an upper bound on the error. Even in the situations where rigorous error estimation is possible, the bounds on the error may not be tight, or may apply to a limited subset of problems, rendering the estimator ineffectual. As a consequence, in practice, these estimators tend to be optimistically applied as *ad hoc* error indicators. A relevant example is the well-known Kelly error "estimator" [122], which is derived from analysis specifically for the Poisson equation, but is widely employed in AMR strategies for the spatial discretizations of many other PDEs [10, 29, 118, 250, 262] despite its lack of theoretical applicability. Equally important, but often overlooked in the literature, the MARK process is also fraught with challenges. In attempting to refine only the cells which constitute a certain percentage of the error, bulk refinement strategies can be expensive in cases with very few cells at singularities, miss important features of the solution, and present difficulties controlling the number of cells in the mesh [61, 185]. With fixed-number refinement, the number of cells in the mesh can be readily controlled, but the approach can wastefully refine too many cells [61]. Furthermore, the parameters in either of these methods (*i.e.*, the algorithmic realization of a refinement strategy with regard to the percentages of the estimated error to refine and coarsen in the case of bulk refinement or exactly how many cells to refine and coarsen in the case of fixed number refinement) are choices typically made *a priori* before a numerical simulation begins. In doing so, the balance between coarsening and refinement is implicitly assumed to be static in time, rather than dynamically driven by the behavior of the underlying solution and available computational resources.

As a result, even after decades of research, in practice, AMR strategies remain largely unprincipled and often require domain-specific knowledge, trial and error, or manual intervention. Selecting and combining an efficient set of AMR heuristics for a new problem is a challenging endeavor and an open research problem in general [61]—there exists a clear need for an automated, flexible, and principled approach to AMR, motivating the present research.

We propose the treatment of AMR as a partially observable Markov decision process (POMDP) that can be interpreted as a local Markov decision process (MDP) on each element and apply a deep reinforcement learning (RL) approach [13] in which we train an agent to increase or decrease mesh resolution, balancing improved accuracy against the computational cost associated with each mesh modification decision. Deep reinforcement learning replaces the expected reward function from the well-known classical Q-learning algorithm with a neural network as a function approximator [236], allowing for the discovery of arbitrarily sophisticated decision-making policies based on unstructured input data [113, 233] and obviating the need to manually engineer aspects of the strategy. In doing so, we replace the ESTIMATE and MARK processes in AMR with a trained RL policy learned from numerical simulation.

Most of the results presented in this chapter have been published in [81]. Our focus primarily on high-order discontinuous Galerkin finite element methods (DG-FEM), in part due to their incredible success in modeling a wide range of phenomena in computational physics (see [110, 133] and references therein) and due to their discontinuous representation of the numerical solution. It is this discontinuous representation that reliably links the smoothness of the local solution as measured by the interface jumps to local error [39] in a way that can be well-leveraged by a decision-making agent, as we develop in §5.2.2.2. The same property gives rise to the simple and effective non-conformity error estimator and its variants examined in the DG-FEM literature [91, 70, 129, 130, 188]. As demonstrated in what follows, the DG-FEM discretization plays an important role in the construction of the observation spaces for the reinforcement learning problem. However, in principle, our methodology could be applied with minor modifications to classical continuous Galerkin finite element methods or even other numerical methods such as finite difference or finite volume methods.

5.1.1 Related work and novel contributions

Optimal mesh refinement strategies have been shown to be theoretically learnable in a recurrent neural network setting [28]. The application of deep RL specifically to problems in computational physics is in its infancy, and recent work at the intersection between the disciplines can be found in [75, 255, 256]. A deep RL approach for the related problem of mesh generation is explored in [201]. The work in [261] constitutes the first attempt to formulate AMR as an RL problem and demonstrates the feasibility of the approach. The present work was developed concurrently and independently, and we formulate the deep RL problem differently. To avoid growing and shrinking action and observation spaces, our decision-making problem is inherently local rather than defined over the entire mesh; to that end, our observation space includes measures of local non-conformity of the solution. We elect not to impose a max refinement depth or provide a hard limit on the refinement budget in the action space; rather, we impose these restrictions implicitly through our reward function. Accordingly, we do not aim to maximize total error reduction; our reward function seeks to strike a tunable balance between the accuracy of the numerical solution and available computational resources.

In this work, we present novel theory and schemes that use a POMDP representation to formulate AMR as a deep RL problem under incomplete information and obtain a trained RL policy that can be thought of as a custom error indicator discovered through trial and error. To the best of our knowledge, this is the first work that includes both refinement and de-refinement actions as part of the resultant policy. We provide a very general framework for choosing an observation space that can incorporate physically relevant features of the PDE to be solved. The methodology is non-reliant on an exact solution or ground truth during model training or deployment. The agent learns a cost-effective refinement and coarsening strategy that balances improved solution accuracy with computational cost, as opposed to a hard threshold.

DG-FEM methods have been shown to be competitive in the *under-resolved* regimes of fluid flow simulation, both in the sense of stability and robustness of the schemes [77], and in their ability to transport high-frequency features over long time-integration horizons without altering the shape of the features or losing amplitude (dispersion and dissipation) [244, 78]. Therefore, practically, the ability of DG-FEM schemes to capture and preserve physical features is a more important criterion of merit than the norm-measured errors often presented in academic convergence studies, (see [133], pp.35-48). This is crucial, as it suggests that any performant AMR technique should be measured against its ability to resolve dynamical features while making efficient use of problem degrees of freedom. This motivates the specific form of our reward function, as well as informs our primary objective of this work; to obtain an AMR policy that accurately resolves features of the numerical solution and efficiently uses computational resources.

This chapter is organized as follows. In §5.2, we introduce our RL formulation of the adaptive mesh refinement problem. The partially observable Markov decision process is described in §5.2.2, with the action and observation spaces detailed in §5.2.2.1-§5.2.2.2. The design and implementation of a new reward function that allows for model training without the need for exact or ground-truth solutions to the underlying problem are provided in §5.2.3. Procedures for training and deployment are given in §5.2.4, and a brief discussion of the deep learning policy architectures employed therein is specified in §5.2.5. The discontinuous Galerkin spatial discretizations for the numerical forward models are provided in §5.3 for different PDEs. In §5.4, we demonstrate the efficacy of the methodology on a set of numerical test cases spanning a variety of PDEs and numerical methods. We show that the resultant RL policies can outperform widely-used AMR heuristics in terms of accuracy per degree of freedom and that the policy corresponding to a single trained model generalizes well to different boundary conditions, forcing functions, and problem sizes. We offer concluding remarks in §5.5.

5.2 Deep reinforcement learning framework

Deep reinforcement learning combines reinforcement learning with deep learning using a neural network to represent the value function, policy, or model considered in a classical RL setting [165]. Similar to other deep learning approaches, the network is typically trained by optimization of a loss function over many episodes in which a strategy that maximizes the expected reward (5.1) is determined through trial and error. This approach is attractive, as it does not require domain-specific knowledge, nor does it require a large training data set; training data is generated experientially by "self-play" as the agent interacts with its environment.

Formulating an RL approach (§5.2.1) to a problem involves describing the underlying decision process and representation of the agent and the state (§5.2.2). We must specify a reward function (§5.2.3) which encodes desirable behaviors of the resultant learned policy. Lastly, we must specify the neural network architectures which are to represent the different parts of the RL problem (§5.2.5). The subsections that follow present our novel formulation of adaptive mesh refinement as a deep RL problem.

5.2.1 Notation

Reinforcement learning is characterized by a decision-making agent that interacts with an environment described by a state S. We consider the discrete-time case: at time step $t \in \{0, 1, 2, \dots\}$, the observable state of the environment is S_t and the action selected by the agent is A_t, the latter of which gives rise to the reward R_{t+1} and modified state S_{t+1}.

A partially observable Markov decision process (POMDP) is a formalism for describing a sequential decision-making process under incomplete information [216]. A POMDP is characterized by ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, T, $\mathcal{O}$, O, γ), where the sets $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ are the sets of possible states, actions, and rewards, respectively. The state transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ encodes the distribution of transition probabilities from a state $S_t = s$ to $S_{t+1} = s'$ given the action a; that is, $T(s, a, s')$ returns the probability of transitioning to state $S_{t+1} = s'$ under the action $A_t = a$ executed in state $S_t = s$. The set $\mathcal{O}$ is the set of possible observations, and the probability distribution $O(o|s, a)$ describes the probabilities over the set of possible new observations $O_{t+1} = o \in \mathcal{O}$ upon taking action $A_t = a$ in state $S_t = s$. The scalar $\gamma \in [0, 1]$ is a time discount factor—a lower discount factor motivates the agent to favor taking actions early. A POMDP differs from a standard Markov decision process (MDP) in that the agent does not have access to the complete state S_t, but rather, indirect observations of it.

The goal of RL is to find a stochastic policy function $\pi : \mathcal{O} \to \mathcal{A}$ that maps the observation space, which we take as a subset of the entire space $\mathcal{S}$, to the set of actions that maximizes the expected reward

$$Q_t(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s,\ A_t = a \right] \tag{5.1}$$

over an infinite time horizon [232].

5.2.1.1 Mesh description

We now provide the notation for the computational mesh and finite-element operations (§5.3). We let $\mathcal{T}_h = \cup_i K_i$ be a finite collection of non-overlapping elements K_i that discretizes the entire problem domain $\Omega \subset \mathbb{R}^d$. We refer to the boundary of the problem domain as Γ. The set $\partial\mathcal{T}_h = \{\partial K : K \in \mathcal{T}_h\}$ refers to all boundary edges and interfaces of the elements, where ∂K is the boundary of element K. For two elements K^+ and K^- sharing an edge, we define $e = \partial K^+ \cap \partial K^-$ as the edge between elements K^+ and K^-. Each edge can be classified as belonging to either ε° or ε^∂, the set of interior and boundary edges, respectively, with $\varepsilon = \varepsilon^\circ \cup \varepsilon^\partial$.

The elements K^+ and K^- have outward pointing unit normals $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$, respectively. The quantities $a^\pm$ denote the traces of a on the edge e from the interior of $K^\pm$. When relevant for element-wise operations, we take as convention that the element K^- refers to the local element, and K^+ to the neighboring element. The jump $[\![\cdot]\!]$ operator for scalar quantities are then defined as $[\![a]\!] = a^- - a^+$ on the interior faces $e \in \varepsilon^\circ$. On the edges described by $\partial\mathcal{T}_h$, we can uniquely define the normal vector $\boldsymbol{n}$ as outward for a given cell and inward for its neighbor.

5.2.2 Elemental refinement as a local Markov decision process

As per the approach described in [10, 20] and references therein, the computational mesh is composed of linear, quadrilateral, or hexahedral elements, and represented by a tree data structure. Upon refinement of an element, 2^d child elements are generated by bisection and marked as active, while the parent element is marked as inactive. The active cells of the computational mesh at any given time are the leaves of the tree. This process is schematically illustrated in Figure 5-1. For the methods described in this chapter, we refer to elements (active or inactive) which share the same parent element as "sibling" elements.

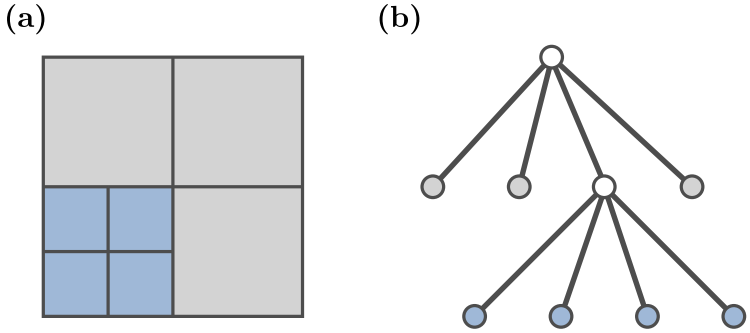


Figure 5-1: (a) Adaptively refined elements in a computational mesh $\mathcal{T}_h$ generated from a single element. (b) Underlying data structure representation as a tree, with inactive parent elements and active elements as the leaves of the tree.

The state S_t is characterized by the current computational mesh $\mathcal{T}_h$, its numerical solution, u_h, and any data known to the PDE (boundary conditions, forcing functions, *etc.*), which we refer to as

$\mathcal{D}$. Although the POMDP would be well-defined if the observation space $\mathcal{O}$ is the set of all possible mesh configurations and numerical solutions, and even some nonlinear functions of these variables, such choices of the global state and observation space would be computationally intractable, as the agent could encounter a combinatorially growing action space to explore.

In light of this fact, we consider a local description of the state as the POMDP observation space used to train the agent. During training, the agent observes a single cell locally, takes action on the current cell alone, and receives a reward which is the effect of its local action on the entire computational domain. Defining the POMDP observation space in this way, such that $\mathcal{O} \subseteq \mathcal{S}$, allows for a fixed action space size as the size of the observation space is known and fixed. We can then formally define the current state S_t as the union of the observation space on every element of the mesh; $S_t = \cup_{K \in \mathcal{T}_h} O_K(t)$.

Following every action, the agent is subsequently moved to another cell in the mesh randomly, according to a distribution $B_t(O_t, S_{t+1})$ which encodes the POMDP belief in the state of the environment and which cell should be visited next. In this work, for simplicity, we take the distribution to be uniform over all cells in the mesh. However, a Bayesian treatment of the belief state is also possible within our RL formulation.

Although it may seem counter-intuitive to train the agent on small subsets of the entire state space, formulating the learning problem locally in this way has several advantages. Considering the entire state of the mesh as an MDP leads to a growing and shrinking action space as the mesh changes, with a combinatorially growing number of action possibilities; this can be addressed [261]—however, the complicated action space may present training and generalization difficulties. Second, many problems in computational physics are governed by PDEs with strong spatial locality, which is why the linear systems arising from the discretization of PDEs are typically sparse. Therefore, viewing the RL problem on a single element rather than on the entire mesh makes sense, as the numerical solution is typically more sensitive to local phenomena. This is not always the case; *e.g.*, in elliptic PDEs, errors affect the numerical solution globally [234, 211]. To address this, we specifically include elliptic PDE test cases in our numerical experiments in §5.4.

Lastly, the resultant trained neural network constituting the RL policy has the intuitive interpretation of a custom cell-wise error indicator learned during training. With this interpretation, it can be seen that the same trained model can be deployed repeatedly over the entire mesh, in parallel if desired.

5.2.2.1 Action space

The action space of the agent is the discrete set $\mathcal{A} = \{\texttt{coarsen}, \texttt{do nothing}, \texttt{refine}\}$, with respect to the current cell. The goal of the RL problem can be stated succinctly as: train the agent such that it refines in areas that are locally under-resolved, coarsens in areas that are locally over-resolved, and takes no action otherwise. Furthermore, the agent should take these actions flexibly and efficiently with respect to the available computational resources.

While it is always possible to refine, the mesh may be in a state such that it is not topologically possible to coarsen the current cell. This is because the current cell could be at the coarsest possible level of the mesh, *e.g.*, a computational mesh comprised of a single cell, or because the siblings of the current element could have arbitrarily many children, and coarsening is only possible when all the sibling cells are at the highest possible refinement level (*i.e.*, leaves on the tree). For example, in the mesh depicted in Figure 5-1, all of the elements marked as blue can be coarsened, whereas those marked as gray can not. Our implementation is such that if coarsening is not possible, the agent defaults to doing nothing.

5.2.2.2 Observation space

The observation space $\mathcal{O}$ is the observable portion of the total state $\mathcal{S}$. The observation space on an element $K \in \mathcal{T}_h$ of the computational mesh consists of:

A. The averaged absolute jump in the numerical solution over the boundary of the cell, denoted

$$\Xi_K = \frac{1}{|\partial K|} \int_{\partial K} |[\![u_h]\!]| \, d\partial K$$

as well as $\Xi_{K'}$ for the cell neighbors K'

B. The average integrated jump across all mesh elements $\sum_K \Xi_K / N_K$

C. The current usage of available computational resources p

D. Any physically relevant features of the numerical solution or data $\phi(u_h, \mathcal{D})$ local to the cell, where $\phi(\cdot)$ refers to a feature and $\mathcal{D}$ refers to available data

The rationale for the inclusion of item (A.) is that discontinuities present at the element interfaces measure the non-conformity of the numerical solution. This is due to the assumption that, in the absence of a physical discontinuity, the exact solution is continuous across element interfaces. This observation is borne out by straightforward analysis, which shows a strong relationship between the residuals of the numerical solution u_h over the element interior and the jumps across the inter-element boundaries [40], leading to the development of so-called "non-conformity" error estimators [188] specific to discontinuous Galerkin finite element methods.

Although formulating the observation space of the POMDP as a completely local problem is computationally attractive, it is clear that in the absence of global information, the agent should always perform the `refine` action. Items (B.) and (C.) both serve the purpose of communicating global information regarding the relative utility of local refinement to the local decision process.

The inclusion of the average integrated jump over all mesh elements provides information about the relative estimate of the error on the current cell as compared to other cells in the mesh. During training, this element of the observation space allows the agent to decide when to forgo the opportunity to refine the current cell in order to spend computational resources elsewhere in the mesh, where the numerical errors may be greater than those observed locally.

The scalar $p \in [0, 1]$ indicates the current usage of available computational resources. In this work, we take p to be the fraction of active mesh elements out of a user-specified maximum number of elements. However, p need not have this representation: in the case of an HPC application, the value of p could be measured directly by monitoring CPU or memory usage in real time. Alternatively, p could be defined with respect to a maximum allowable wall-clock time per solution time step or could be computed from some other *a priori* allocation of computational resources such as RAM or arithmetic throughput. Regardless of how the computational usage p is defined, its inclusion in the observation space serves to indicate to the agent the availability of computational resources; the reward function §5.2.3 specifies the cost-benefit relationship of making use of them.

Due to item (D.), this observation space is very flexible and general. This is by design, as it allows the user to include physically or computationally relevant information which may help the agent learn a more effective strategy. As an example, the advection problem as described in §5.3.2 may include the local convective velocity in the observation space, as it determines the flow of information.

Remark. For the RL formulations provided in this chapter, we emphasize discontinuous Galerkin methods due to the simplicity and effectiveness of the non-conformity of the local solution as a proxy for the local error. In the case of classical continuous Galerkin finite element schemes, the jump of the solution along the element boundary is identically zero, due to the continuous nature of the approximation spaces in which the numerical solution is sought [33]. However, item (A.) can be readily replaced with other indicators of non-conformity in order to generalize the methodology to other finite element discretizations. For example, the jump of the solution gradient along the element boundary can be applied as a simple surrogate in the continuous Galerkin case—the core principle is to select a numerical quantity that vanishes as the numerical solution approaches the exact solution of the PDE. However, more sophisticated indicators of non-conformity exist [19].

5.2.3 Designing the reward function

The reward function is an environment-provided reinforcement signal that determines the immediate reward or penalty due to each decision on the part of the agent [236]. In that sense, the reward function is central to the ultimate behavior of the agent after training and should encode all the information pertaining to what we wish our agent to accomplish. For AMR, the reward function used to train the agent should strike a balance between improvements in accuracy and incurring additional computational costs. With these conditions, our proposed reward function takes the general form of

$$[\text{accuracy}] - \gamma_c[\text{cost}]B(p). \tag{5.2}$$

The coefficient γ_c is a scaling factor expressing the relative importance of the accuracy of the solution versus increasing computational cost in the RL agent's policy; it can either be tuned as a hyperparameter or empirically computed for different computational regimes. The function $B(p) : [0, 1) \to [0, \infty)$ acts as an asymptotic barrier discouraging the agent from exceeding the limit of its computational resources. Two choices are shown in in Figure 5-2. The barrier function can be modified to encourage the agent to use a certain percentage of resources; for example, to incentivize aggressive refinement in the under-resolved case, which we refer to as a barrier function "with hortation". In what follows, unless otherwise specified, we use the non-hortative barrier function $B(p) = \sqrt{p}/(1 - p)$.

It remains to define a metric for improvement in solution and change in computational cost for each of the terms in equation (5.2).

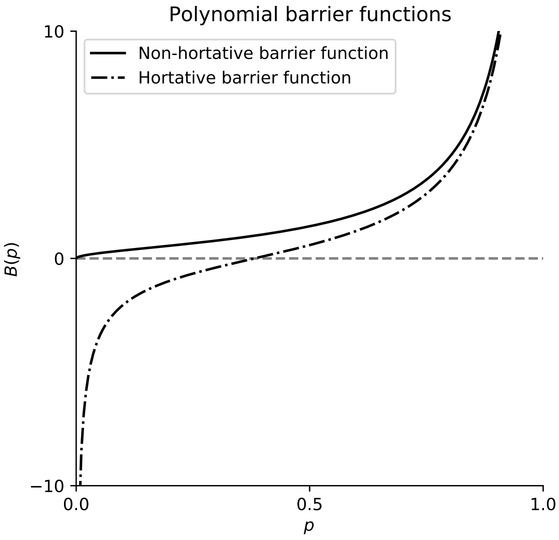


Figure 5-2: Example of the non-hortative polynomial barrier functions $B(p) = \sqrt{p}/(1 - p)$ and hortative barrier function $B(p) = p/(1 - p) - [1/\sqrt{p} - 1]$.

Accuracy. The optimality properties of the numerical solution arising from continuous and discontinuous Galerkin finite element discretizations ensure that the error of the solution (measured in an appropriate norm) decreases or remains the same upon any refinement of the computational mesh [33, 39, 110]. These properties are crucial, as they guarantee that any change in the error of the numerical solution, $e(u_h) = \|u_h - u_{\text{exact}}\|_{L^2(\Omega)}$, that arise as a result of refinement will be monotonically non-increasing. Therefore the fundamental idea is to reward any change in the numerical solution upon refinement, and to penalize any change to the numerical solution upon coarsening. We define the quantity

$$\Delta u_h = \sum_{K \in \mathcal{T}_h} \int_K |u_h^{t_{\text{RL}}+1} - u_h^{t_{\text{RL}}}| \, dK \tag{5.3}$$

to represent the change in the solution, upon the action A_t, as is schematically illustrated in Figure 5-3. From here onward, we will use the integers t_{RL} and $t_{\mathrm{RL}}+1$ to refer to a RL time step corresponding to the POMDP notation in §5.2.1 and use t to refer to time in the case of a time-dependent PDE. The quantity Δu_h represents the global difference in the numerical solution as a result of different successive levels of local refinement. The computation of Δu_h is always performed by first interpolating the coarser of the two solutions onto the finer of the two grids, then performing the integration. This way, the change in solution Δu_h upon refining and coarsening the same element are identical in magnitude, but with opposite signs, making every decision by the agent to coarsen or refine reversible.

Naively applying the reward $+\Delta u_h$ upon refinement and $-\Delta u_h$ upon coarsening is logical; however, this choice often results in an inconsistent reward signal due to scaling. One of the attractive features of high-order finite element methods is that the error in the numerical solution decreases quickly with respect to the mesh element size for problems with smooth solutions, typically as $1/h^{p_{\mathrm{order}}}$ where h is a representative size of a mesh element, and p_{order} is the polynomial order of the scheme [33, 110]. As a result, the error in numerical solutions often spans many orders of magnitude over a small number of mesh refinement cycles. To account for these large differences in the scale of the error, and to prevent the initial refinements from dominating the total achievable reward during training, we scale the rewards logarithmically, awarding the agent

$$R_{\Delta u} = \pm\left[\log(\Delta u_h + \epsilon_{\mathrm{machine}}) - \log \epsilon_{\mathrm{machine}}\right],$$

where the positive and negative signs apply to the cases of refinement and coarsening, respectively. Here, $\epsilon_{\mathrm{machine}}$ is a representation of machine precision, which we take to be 10^{-16}, and is included for the edge cases in which refinement or coarsening does not change the numerical solution. Of course, it can also be a desired accuracy for the numerical solution. Lastly, the additive factor is chosen to center the positive and negative rewards around zero, rather than $\log(\epsilon_{\mathrm{machine}})$ for interpretability, and follows from our specific choice of machine precision.

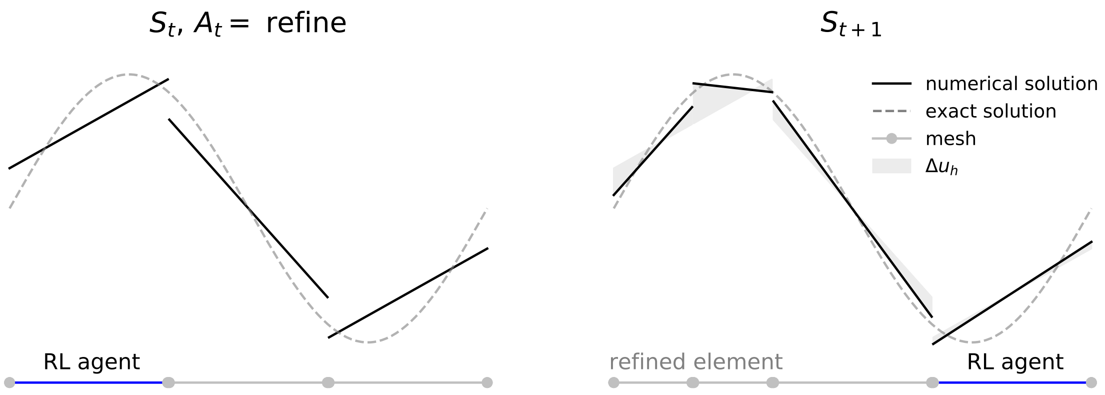


Figure 5-3: Schematic of Δu_h upon refinement. The RL agent's new location is sampled from the belief distribution $B_{t_{\mathrm{RL}}}(O_{t_{\mathrm{RL}}}, S_{t_{\mathrm{RL}}+1})$.

Cost. We can specify the reward due to the change in computational cost upon action $a_{t_{\mathrm{RL}}}$ as the quantity

$$R_{\Delta C} = B\left(p^{t_{\mathrm{RL}}+1}\right) - B\left(p^{t_{\mathrm{RL}}}\right). \tag{5.4}$$

The sign of $R_{\Delta C}$ is not explicitly dependent on the action $a_{t_{\mathrm{RL}}}$, as it purely relates to the increase or decrease in computational cost as measured by p before and after the action occurs.

The reward function. Combining the constituent components, the final explicit form of the new reward function that we employ is

$$R(s_{t_{\mathrm{RL}}}, a_{t_{\mathrm{RL}}}) = \begin{cases} +\left[\log(\Delta u_h + \epsilon_{\mathrm{machine}}) - \log(\epsilon_{\mathrm{machine}})\right] - \gamma_c R_{\Delta C}, & \text{if } a_{t_{\mathrm{RL}}} = \texttt{refine} \\ -\left[\log(\Delta u_h + \epsilon_{\mathrm{machine}}) - \log(\epsilon_{\mathrm{machine}})\right] - \gamma_c R_{\Delta C}, & \text{if } a_{t_{\mathrm{RL}}} = \texttt{coarsen} \\ 0, & \text{if } a_{t_{\mathrm{RL}}} = \texttt{do nothing}. \end{cases} \tag{5.5}$$

As a convention, we take $R_{t_{\mathrm{RL}}+1} = R(a_{t_{\mathrm{RL}}},\, s_{t_{\mathrm{RL}}})$ to be the reward returned by the environment, given the input of action and state at time t_{RL}.

We remark that equation (5.3) can be naively applied to vector-valued problems as well, with u_h comprising a vector-valued state rather than a scalar-valued one. However, care should be taken in selecting this state and performing differencing between refinement levels. For example, in the case of the incompressible Navier–Stokes equations, the difference can be computed with the velocity field alone or involve the pressure. In this case, relative scaling between the fields becomes important in preserving the reward signal.

5.2.4 Training and model deployment

To train the model on a static problem, we begin each episode with a very coarse mesh. The agent is placed randomly on a cell in the mesh; after collecting the reward for each action, the agent is moved to a different element randomly as described in §5.2.2. After a large given number of actions have occurred, the episode ends, with early termination possible in the case of repeated `do nothing` actions. If at any point the agent exceeds the allotted computational budget, it receives a large negative reward and the episode terminates (see Appendix C). For time-dependent problems, the training is similar. The typical RL action/reward cycles depicted in Figure 5-3 are performed within a single time step, using the previous time solution state u_h^{t-1} as the starting condition. After a random number of iterations, the current RL-iterated solution is advanced to the next time step and the process is repeated. We sample from a uniform distribution $\mathrm{Uniform}(1, \text{max episode iterations})$ to determine the number of iterations before the solution is advanced in time. In order for experience signals to be propagated to the policy network during training, all that matters is that there are areas of under- and over-resolution in the numerical solution; as long as the time advancement allows this to occur, other valid heuristics for advancing the solution are possible.

For model deployment, the trained model considers every cell in the mesh sequentially, in the order specified by the non-conformity estimator, and makes a decision based on the local observation on that particular element. The percentage p of resources in use is updated in between elements so that the model may update its recommendation. This constitutes the marking process—once every element has been visited, the recommendations of the RL policy can be executed, concluding a single AMR "cycle" for the RL model. The number of cells allowed as a budget during model deployment can be significantly larger than that used in training. The trained policy network is agnostic to the actual maximum number of cells allocated; rather, it sees only the percentage of the available cells currently in use through the parameter p.

5.2.5 Policy and deep learning architectures

We now describe the particular architectures we used for our AMR agent. Our deep Q-network [183] is the simplest policy. It consists of an input layer, the size of which corresponds to the size of the observation space, two hidden-layers, each with 64 neurons, and an output layer with one neuron for each possible action. Rectified linear units (ReLUs) are chosen as our activation functions. Taking the argmax of the output yields the action we take (assuming we are using the optimal policy deterministically).

The Advantage Actor-Critic (A2C) policy [182] uses the same network architecture as the deep Q-network, but it trains two networks, one to estimate the value function (the critic) and one to determine the optimal policy (the actor). At each step, the actor chooses an action that the critic evaluates to estimate the state-value and improve the policy of the actor. Finally, the Proximal Policy Optimization (PPO) policy [231] uses the same network architecture as A2C; the main difference is that PPO uses clipping in the objective function so that policy updates remain relatively small. We compare the performance of these algorithms in §5.4.1.1.

5.2.6 Summary: reinforcement learning framework

The design of the reward function is pivotal in RL. The idea behind the reward signal (5.5) is illustrated in Figure 5-3. When the agent uses resources to refine the solution, it accumulates a reward corresponding to how much the numerical solution changes as a result of the refinement ($R_{\Delta u}$) and is penalized according to the increased usage of its resource budget ($R_{\Delta C}$). In the limiting case, in which the solution is perfectly resolved, the agent accrues only penalization for additional refinement. Conversely, when the agent elects to coarsen the resolution, it is rewarded according to the resources it saves but pays for any change in the numerical solution. In the limiting case where the solution is perfectly resolved with more elements than necessary, the agent receives a reward only for decreasing the number of elements, as the numerical solution does not change upon coarsening. In essence, the reward function provides a signal which communicates the trade-off between accuracy and computational cost. To preserve the reward signal, the reward function logarithmically scales and machine-precision-normalizes the quantities $R_{\Delta u}$ and $R_{\Delta C}$, which can span many orders of magnitude. The hyperparameter γ_c is a tunable setting that reflects the degree of displeasure incurred upon using additional resources, effectively weighting the user priority of accuracy versus cost.

By formulating the sequential decision process locally as a POMDP, the observation space provides the link between the local solution data and its conformity to the cost-benefit trade-off of refinement in that region. Informally speaking, the observation $O_{t_{\text{RL}}}$ provides a signal relating local solution conformity to that of the global solution as well as the remaining computational resources and other potentially relevant data.

```
training_step($A_{t_{\text{RL}}}$, $O_{t_{\text{RL}}}$):
$K_{t_{\text{RL}}}$ ← cell corresponding to $O_{t_{\text{RL}}}$
if $A_{t_{\text{RL}}}$ = refine
  $S_{t_{\text{RL}}+1}$ ← refine $K_{t_{\text{RL}}}$
if $A_{t_{\text{RL}}}$ = coarsen
  $S_{t_{\text{RL}}+1}$ ← coarsen $K_{t_{\text{RL}}}$
update $u_h^{t_{\text{RL}}+1} = \mathcal{M}(S_{t_{\text{RL}}+1}),\quad p_{t_{\text{RL}}+1}$
compute $\Delta u_h$ (on finer mesh) via (5.3), $R_{\Delta C}$ via (5.4)
compute $R_{t_{\text{RL}}+1}$ via (5.5)
sample cell $K_{t_{\text{RL}}+1} \leftarrow B_{t_{\text{RL}}}(O_{t_{\text{RL}}}, S_{t_{\text{RL}}+1})$
$O_{t_{\text{RL}}+1}$ ← compute observation space for $K_{t_{\text{RL}}+1}$
if $p_{t_{\text{RL}}+1} > 1$
  $R_{t_{\text{RL}}+1}$ ← large, negative reward
  done ← true
if iteration ≥ episode iterations
  done ← true
return $R_{t_{\text{RL}}+1}$, done, $O_{t_{\text{RL}}+1}$
```

```
model_deployment($S_0$, $\pi_h$):
$S' \leftarrow S_0$
$\{T\}$ ←sort $K \in \mathcal{T}_h$ by $\Xi_K = \int_{\partial K} [\![u_h]\!] d\,\partial K$
For $K' \in \{T\}$
    $O_{K'}$ ← compute observation space for $K'$
    $A_{K'} \leftarrow \pi_h(O_{K'})$ query policy network
    execute action $A_{K'}$, update $S'$, $p$
compute new solution $u_h' \leftarrow \mathcal{M}(S')$
return $u_h'$, $S'$
```

Figure 5-4: Deep RL-AMR. Algorithms for a single RL training time step (left) and for the deployment of the trained policy over a single refinement cycle (right). The training step takes as input an action $A_{t_{\text{RL}}}$ and an observation $O_{t_{\text{RL}}}$, and returns the reward $R_{t_{\text{RL}}+1}$, the Boolean episode termination condition `done` (default false), and the new observation $O_{t_{\text{RL}}+1}$. The model deployment procedure takes as input a trained policy network π_h and starting state S_0; it returns the proposed next mesh state S' and the updated numerical solution u_h'.

We summarize the algorithm for a training time step and model deployment in Figure 5-4. The notation $u_h^{t_{\text{RL}}} = \mathcal{M}(S_{t_{\text{RL}}})$ describes running the forward model on the mesh state $S_{t_{\text{RL}}}$ to compute the corresponding solution $u_h^{t_{\text{RL}}}$ by numerically solving the PDE. The trained policy network is denoted

π_h (§5.2.2), episode iterations denotes a chosen number of maximum iterations during training, and we use the Boolean `done` as a sentinel for episode termination. Lastly, we refer to the belief distribution over the mesh cells $K \in \mathcal{T}_h$ at state $S_{t_{\mathrm{RL}}}$ as $B_{t_{\mathrm{RL}}}(O_{t_{\mathrm{RL}}}, S_{t_{\mathrm{RL}}})$. For the architectures we employ (§5.2.5), we refer to [212] for the complete specification as to how the weights of the policy network are updated as a result of the training steps taken.

5.3 Finite element spatial discretization and problem physics

5.3.1 Notation and approximation spaces

We define the inner products over a set $D \subset \mathbb{R}^d$ and its boundary $\partial D \subset \mathbb{R}^{d-1}$ using typical discontinuous Galerkin finite element notation as

$$(c, d)_D = \int_D cd \, \mathrm{d}D, \qquad \langle c, d \rangle_{\partial D} = \int_{\partial D} cd \, \mathrm{d}\partial D.$$

Let $\mathcal{P}^{p_{\mathrm{order}}}(D)$ denote the set of polynomials of degree p_{order} on a domain D. We consider the discontinuous finite element spaces

$$\begin{aligned} W_h^{p_{\mathrm{order}}} &= \left\{ w \in L^2(\Omega) \colon w\big|_K \in \mathcal{P}^{p_{\mathrm{order}}}(K) \, \forall K \in \mathcal{T}_h \right\}, \\ V_h^{p_{\mathrm{order}}} &= \left\{ \boldsymbol{v} \in \left[L^2(\Omega) \right]^d : \boldsymbol{v}\big|_K \in [\mathcal{P}^{p_{\mathrm{order}}}(K)]^d \, \forall K \in \mathcal{T}_h \right\}, \\ M_h^{p_{\mathrm{order}}} &= \left\{ \mu \in L^2(\varepsilon_h) : \mu\big|_e \in \mathcal{P}^{p_{\mathrm{order}}}(e) \, \forall e \in \varepsilon_h \right\}, \end{aligned}$$

where $L^2(\Omega)$ is the space of square-integrable functions on the domain Ω. Informally, $W_h^{p_{\mathrm{order}}}$ represents the space of piecewise discontinuous polynomials of degree at most p_{order} on every element in the mesh.

5.3.2 Linear advection equation

We will consider the linear advection equation of a tracer u

$$\begin{aligned} \frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{c} u) &= f \qquad && \text{in } \Omega, \\ u &= g_D && \text{on } \Gamma_{\mathrm{in}}, \end{aligned} \tag{5.6}$$

on the domain Ω with boundary Γ separated into inflow and outflow regions $\Gamma = \Gamma_{\mathrm{in}} \cup \Gamma_{\mathrm{out}}$, that are defined according to the continuous, divergence-free advection velocity field $\boldsymbol{c}$ and outward normal $\boldsymbol{n}$,

$$\begin{aligned} \Gamma_{\mathrm{in}} &= \{ x \in \Gamma \colon \boldsymbol{c} \cdot \boldsymbol{n} \le 0 \}, \\ \Gamma_{\mathrm{out}} &= \{ x \in \Gamma \colon \boldsymbol{c} \cdot \boldsymbol{n} > 0 \}. \end{aligned} \tag{5.7}$$

The linear advection problem defined in equation (5.6) admits the following semi-discrete discontinuous Galerkin discretization [39, 133]: we seek $u_h \in W_h^{p_{\mathrm{order}}}$ such that

$$\left(w, \frac{\partial u_h}{\partial t} \right)_{\mathcal{T}_h} - (\nabla w, \boldsymbol{c} u_h)_{\mathcal{T}_h} + \langle [\![w]\!], u^* (\boldsymbol{c} \cdot \boldsymbol{n}) \rangle_{\varepsilon^\circ} + \langle w, u_h(\boldsymbol{c} \cdot \boldsymbol{n}) \rangle_{\Gamma_{\mathrm{out}}} = (w, f)_{\mathcal{T}_h} - \langle w, g_D(\boldsymbol{c} \cdot \boldsymbol{n}) \rangle_{\Gamma_{\mathrm{in}}},$$

for all $w \in W_h^{p_{\mathrm{order}}}$. There are many choices for the single-valued, inter-element numerical flux u^*. To mimic the physics of the problem, we use the common upwinded flux

$$u^* = \begin{cases} u_h^+, & \boldsymbol{c} \cdot \boldsymbol{n} < 0 \\ u_h^-, & \boldsymbol{c} \cdot \boldsymbol{n} \ge 0. \end{cases}$$

In the steady case, the time derivative term is set to zero. In the unsteady case, a standard explicit time-integration scheme can be used for temporal discretization using a method of lines approach [110]; we use LSERK-45 [37].

5.3.3 Advection-diffusion equation

We will show that the methodology generalizes well across both different problem physics and different finite element discretizations. To do so, we will consider advection-diffusion PDEs, which include, as a subset, second-order Poisson-type problems. Namely, we consider the set of problems described by the PDEs

$$\begin{aligned} \frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{c} u) - \nabla \cdot (\kappa \nabla u) = f, \qquad & \text{in } \Omega \\ (-\kappa \nabla u + \boldsymbol{c} u) \cdot \boldsymbol{n} = g_N, \qquad & \text{on } \Gamma_N, \\ u = g_D, \qquad & \text{on } \Gamma_D, \end{aligned} \tag{5.8}$$

where Γ_D and Γ_N denote the Dirichlet and Neumann segments of the boundary, respectively. The problem in equation (5.8) admits a hybridizable discontinuous Galerkin (HDG) formulation, developed in [191]: we seek $(\boldsymbol{q}_h, u_h, \hat{u}_h) \in V_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}}$ such that

$$\begin{aligned} \left(\boldsymbol{v},\, \kappa^{-1} \boldsymbol{q}_h\right)_{\mathcal{T}_h} - (\nabla \cdot \boldsymbol{v},\, u_h)_{\mathcal{T}_h} + \langle \boldsymbol{v} \cdot \boldsymbol{n},\, \hat{u}_h \rangle_{\partial \mathcal{T}_h} &= 0 \\ \left(w,\, \frac{\partial u_h}{\partial t}\right)_{\mathcal{T}_h} - (\nabla w,\, \boldsymbol{c} u_h)_{\mathcal{T}_h} + (w,\, \nabla \cdot \boldsymbol{q}_h)_{\mathcal{T}_h} + (w,\, (\boldsymbol{c} \cdot \boldsymbol{n}) \hat{u}_h)_{\partial \mathcal{T}_h} + \langle w,\, \tau (u_h - \hat{u}_h) \rangle_{\partial \mathcal{T}_h} &= (w,\, f)_{\mathcal{T}_h} \\ \langle \mu,\, \boldsymbol{q}_h \cdot \boldsymbol{n} + (\boldsymbol{c} \cdot \boldsymbol{n}) \hat{u}_h + \tau (u_h - \hat{u}_h) \rangle_{\partial \mathcal{T}_h} &= \langle \mu,\, g_N \rangle_{\partial \mathcal{T}_h} \end{aligned} \tag{5.9}$$

for all $(\boldsymbol{v}, w, \mu) \in V_h^{p_{\text{order}}} \times W_h^{p_{\text{order}}} \times M_h^{p_{\text{order}}}$. We take the diffusion coefficient $\kappa = 1$ and choose the stabilization parameter $\tau = \kappa/\ell + |\boldsymbol{c} \cdot \boldsymbol{n}|$, where we use as a diffusion length scale $\ell = 1/5$. In order to solve Poisson-like problems, we set the convective velocity field $\boldsymbol{c}$ to zero. In the unsteady case, an implicit time-integration scheme can be applied for the temporal discretization; we use the third-order backward difference formulae (BDF3). We refer the reader to [191] for a thorough discussion of these choices.

5.3.4 Error indicators and AMR heuristics

For the test cases in §5.4, we benchmark the RL policy against two common AMR heuristics, the Kelly Error indicator and a gradient indicator. The indicators are shown in Table 5.1.

Error indicator	faces $F = K \cap K'$	boundary face Γ_N
Kelly	$\displaystyle\sum_{F \in \partial K} c_F \int_F \left[\!\left[\frac{\partial u_h}{\partial \boldsymbol{n}} \right]\!\right]^2 dF$	$n_F \int_F \left| g_N - \frac{\partial u_h}{\partial \boldsymbol{n}} \right|^2 dF$
Gradient-based	$\displaystyle Y^{-1} \sum_{K'} \frac{y_{K'}}{\lVert y_{K'} \rVert} \frac{u_h(x_{K'}) - u_h(x_K)}{\lVert y_{K'} \rVert}$	-

Table 5.1: Cell-wise error indicators, contributions by face.

The Kelly indicator uses parameters $c_F = n_F = h_K/24$ for element width h_K. The approximate gradient is formed using the distance vectors $y'_K = x'_K - x_K$ between the cell centers of element K and its neighbors K', as well as the matrix

$$Y = \sum_{K'} \left(\frac{y_{K'}}{\lVert y_{K'} \rVert} \frac{y_{K'}^T}{\lVert y_{K'} \rVert} \right).$$

We scale the approximate gradient in Table 5.1 by a power of the mesh width, using $h_K^{1+d/2}||\tilde{\nabla} u_h||$, where the integer d represents the spatial dimension of the problem. Apart from being widely used, the Kelly indicator is a natural choice of estimator to compare with the RL-agent's policy network, as it also gauges error by measuring local nonconformity—specifically, in the jump of the gradient of the solution across element faces. We avoid using the non-conformity error indicator itself, as it is known to exhibit undue dependence on previous refinement history and is recommended to be used in conjunction with other estimators [188]. To execute an AMR strategy based on either of these estimators, we use either a bulk or fixed-fraction rule. As described in §5.1, a bulk strategy refines and coarsens the cells in which the top and bottom percentages of the estimated error occurs; therefore the actual number of cells coarsened or refined is entirely dependent on the estimate. On the other hand, a fixed-fraction refinement strategy sorts the cells according to their error estimates and refines and coarsens a top and bottom percentage of the total number of cells. We denote these strategies as `bulk`(refine percentage, coarsen percentage) and, similarly, `fixed`$(\cdot, \cdot)$; *e.g.*, `fixed`(0.4, 0.6) refers to an AMR strategy where the top 40 percent of cells in terms of their estimated error are refined and the bottom 60 percent are coarsened. For additional details on both these indicators and refinement strategies, see [10]. Budget constraints can be applied to AMR heuristics as well. Imposition of a maximum depth on the tree data structure representing the mesh effectively limits the number of cells as well as the minimum cell width. Alternatively, regardless of the depth of the tree, a maximum number of cells can be imposed directly, after which no refinement is allowed to occur. A disadvantage to both these constraints is that they are simple and arithmetic in nature and make no use of the current state of the solution or percentage of resources used.

5.3.5 Numerical implementation

The finite element forward models were implemented in `C++` and make use of the finite element library `deal.II` [10]. The POMDP characterizing the RL environment was implemented using the OpenAI Gym framework [35], and the custom deep RL architectures were implemented in the open-source RL framework Stable-Baselines 3 (SB3) [212].

Unless otherwise specified, the policy architecture used is A2C, and we take the hyperparameter $\gamma_c = 25$ (see §5.2.3, §5.4.1), which we empirically found to give good performance over a wide range of test cases. The main DRL-AMR parameters for each numerical experiment are listed in Table 5.2. Additional details on RL training are given in Appendix C. As is typical in deep RL, the particular number of training time steps used is not particularly meaningful as compared to the order of magnitude of the total number of steps (see discussion in §5.4.1.1), as long as the policy network is given sufficient time to increase its mean episodic reward significantly from its starting performance. For all experiments, we used on the order of 10^5 RL training time steps, consisting of 100-200 time step episodes, and training took on the order of 1-3 hours on a desktop computer without GPU acceleration. More detailed benchmarking would be extraneous, as performance in terms of training time is dominated by the cost of running the numerical solver rather than updating the policy network, for all but trivial problems. Similarly, we benchmark the resultant heuristics and policy networks using L^2-error per degree of freedom as a figure of merit. Model deployment requires only a forward pass through the policy network, whereas AMR heuristics require computation of the relevant estimators; however, these two operations happen in different programming languages, making CPU-time or wall-clock time performance comparisons between the two unclear. However, as problem size grows, the cost of running the numerical solver dominates the time-to-solution. Since the underlying PDE solvers are the same for both the DRL-AMR and heuristic approaches, we avoid comparisons using CPU time or wall-clock times and show L^2-error per degree of freedom directly, as this metric succinctly illustrates the cost of obtaining a given accuracy using each approach.

All linear systems arising from the finite element discretizations are solved using direct solvers (UMFPACK) to avoid the complicating factors of iterative solver tolerances and stopping criteria. All surface and volume integral operators are discretized with Gaussian quadrature using $p_{\text{order}} + 1$ one-dimensional (1D) quadrature points in each spatial direction, where p_{order} is the polynomial order of the finite element space $W_h^{p_{\text{order}}}$.

L^2-errors, when shown, are computed as a post-processing step using a known exact solution and numerically integrated using $p_{\text{order}} + 3$ Gaussian quadrature points in each spatial direction to ensure that the calculation of errors is not adversely affected by integration error. We re-emphasize that no exact solutions are used at any point during training or deployment of the RL model itself.

5.4 Numerical experiments

The numerical experiments are designed to demonstrate the features and performance of the deep RL AMR (DRL-AMR) approach. We start with a 1D, steady linear advection problem in §5.4.1 as an illustrative example to exhibit the feasibility of the approach and to demonstratively explore the characteristics of the RL policy. We discuss details related to training and model evaluation, compare the performance of different RL algorithms, and easily interpret results due to the simplicity of the test case.

Subsequent experiments focus on the performance and generalizability of the DRL-AMR method. In §5.4.2, we use the same trained RL model as in §5.4.1, but apply it to a different linear advection problem to demonstrate generalization of the policy to different boundary conditions and forcing functions. In §5.4.3, to show the generalization of the method to unsteady dynamics, we train a model on a 1D time-dependent advection problem. Lastly, to support the claim that the approach is not specific to a particular PDE or finite element scheme, we apply the DRL-AMR framework to the Poisson equation discretized with an HDG scheme (§5.3.3). As the Poisson equation is a second-order, elliptic PDE, its solutions are characterized by different physics than the hyperbolic, first-order advection problems solved in §(5.4.1-5.4.3). Similarly, as HDG schemes are mixed methods, their formulation (5.9) involves vector-valued finite element spaces and use of a traced finite element space [191] on the mesh skeleton; that is, the underlying numerical method is significantly different than those used for the solution of the advection equation (*cf.* (5.3.2)).

To show that the methods are not relegated to small or one-dimensional problems and that the DRL-AMR method generalizes to higher spatial dimensions and larger, more complex problems, we examine the performance of deep RL policies trained on 2D problems in §(5.4.5-5.4.6). The steady advection problem in §5.4.5 allows evaluation of DRL-AMR in the context of the two-dimensional generalization of the problem in §5.4.1. The steady advection-diffusion problem in §5.4.6 examines the effectiveness of the deep RL policy when both advection and diffusion processes occur, and highlights its ability to detect and resolve non-trivial features in an automated way by using solution smoothness. The unsteady advection problem in §5.4.7 shows the ability of DRL-AMR to preserve salient physical features of a numerical solution over long-time integration horizons. Throughout the entire set of numerical experiments, we focus on the goal stated in the introduction of providing a numerical solution that captures all physically relevant features efficiently in terms of problem degrees of freedom, avoiding spurious diffusion and dispersion effects due to under-resolution.

The parameters used in training the RL models for all numerical experiments are shown in Table 5.2, along with the AMR heuristics to which we compared the models.

§	p_{order}	γ_c	training episodes	training budget	AMR heuristic	indicator
5.4.1	3	25	$2 \cdot 10^4$	25 cells	`bulk`(0.5, 0.5)	gradient-based
5.4.2	3	25	$2 \cdot 10^4$	25 cells	`fixed`(0.5, 0.1)	Kelly
5.4.3	3	100	$5 \cdot 10^4$	25 cells	`bulk`(0.5, 0.1)	gradient-based
5.4.4	3	25	$2 \cdot 10^5$	20 cells	`fixed`(0.5, 0.5)	Kelly
5.4.5	2	25	$2 \cdot 10^5$	110 cells	`bulk`(0.5, 0.5)	gradient-based
5.4.6	4	25	$3 \cdot 10^5$	200 cells	`bulk`(0.5, 0.5)	Kelly
5.4.7	3	25	$3 \cdot 10^5$	200 cells	`bulk`(0.6, 0.4)	Kelly

Table 5.2: Numerical experiments of §5.4: Training parameters of the DRL-AMR models, AMR heuristics, and indicators.

5.4.1 Steady 1D linear advection

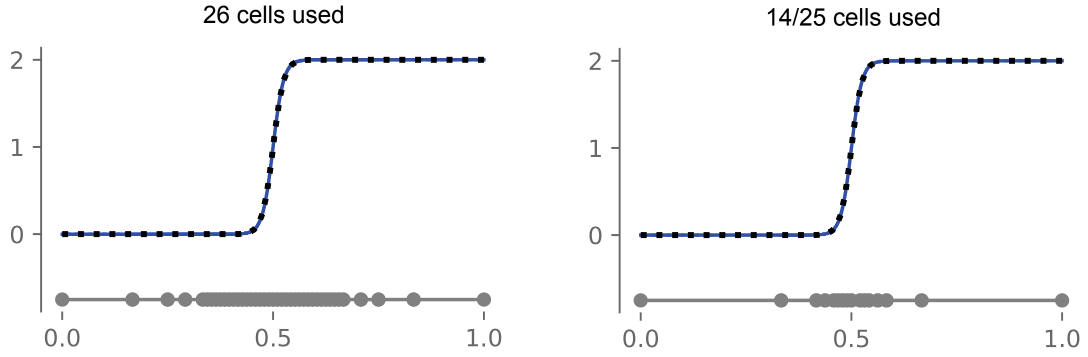
For proof-of-concept, we consider the linear advection equation described in §5.3.2 on the 1D spatial domain $\Omega = [0, 1]$. We choose the boundary conditions g_D at the inlet and forcing function f such that the exact solution takes the form

$$u(x) = 1 - \tanh\left[\alpha(1 - 4(x - 1/4))\right],$$

where we take as the steepness parameter $\alpha = 10$. As the exact solution has the form of a smooth step function, the resultant meshes and numerical solutions are easily interpretable—resolution should be concentrated in the steep gradient region around the "step". In light of this, we will use this illustrative example to demonstrate the features of the trained deep RL policy, as well as the details of training and model deployment. Subsequent sections will focus on generalizability.

Using a numerical solution of polynomial order $p_{\text{order}} = 3$, we train the RL-agent for $2{\cdot}10^4$ episodes on a computational "budget" of 25 cells (§5.2.3). However, at the time of model deployment, we are free to give the trained RL policy whatever budget we wish; we emphasize that this is because in general, we would like to train our model on much cheaper problems than those we intend to solve.

During deployment, starting with a very coarse mesh consisting of 4 elements, we perform 6 AMR cycles comparing the DRL-AMR model to an AMR heuristic that attempts to refine the elements responsible for the top 50 percent of the total error and attempts to coarsen the elements responsible for the bottom 50 percent of total error as measured by the approximate gradient error indicator (Table 5.1). That is, a `bulk`(0.5, 0.5) strategy (see §5.3.4). We expect this heuristic indicator to perform well, as this test case has only one feature, the steep gradient in the center of the domain.



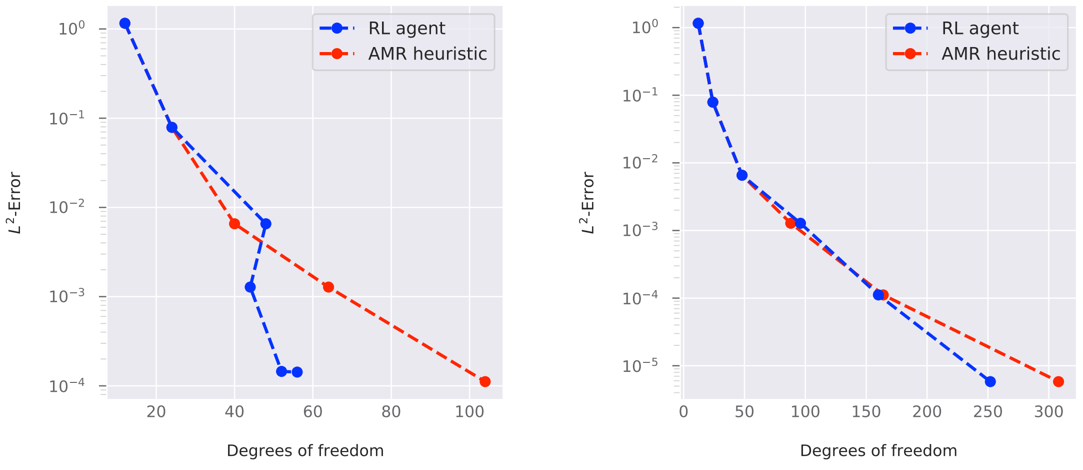
(a) AMR heuristic

(b) RL Agent

Figure 5-5: Steady 1D linear advection (§5.4.1). Numerical solution with the mesh resulting from 6 cycles of refinement, using the approximate gradient error indicator as an AMR heuristic and the deep RL policy resulting from training. The exact solution is overlaid (dashed) in both cases for comparison.

Figure 5-5 shows the numerical solution and meshes proposed by the two approaches. We see that the deep RL agent deployed with a 25-cell budget is able to find a high-quality solution with fewer elements than that of the AMR heuristic recommendation. This is corroborated by the more precise comparison in Figure 5-6, which shows that at model deployments with budgets of 25 and 500 cells, the RL agent outperforms the AMR heuristic over six refinement cycles in terms of L^2-accuracy per degree of freedom. That is, the RL policy provides a numerical solution of approximately the same accuracy as that of the AMR heuristic, but does so with fewer degrees of freedom after the refinement cycles, both on the problem size it was trained on, as well as at a much larger problem size.

Although for 1D problems, differences in problem degrees of freedom are small, this example illustrates the central difference between the learned policy and the classical AMR approaches. AMR algorithms mark cells for refinement either in terms of the estimated volume fraction of total



(a) 25 cell budget

(b) 500 cell budget

Figure 5-6: Steady 1D linear advection (§5.4.1). L^2-error plotted against problem degrees of freedom over 6 cycles of refinement, using the gradient-based error indicator as an AMR heuristic and the deep RL policy: (5-6a) deployed with a computational budget of 25 elements; (5-6b) with a computational budget of 500 elements.

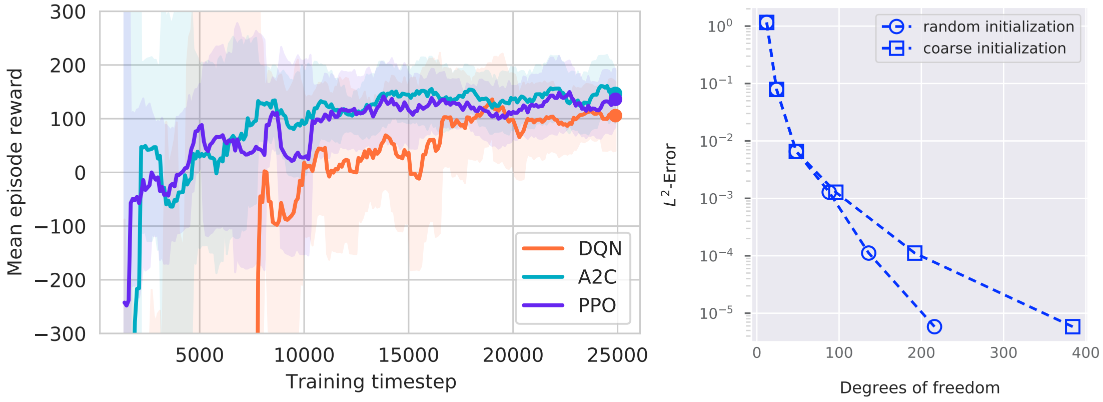
error in the case of bulk refinement or by a certain number of cells in the case of fixed-number refinement. In either case, the AMR algorithms rely on estimators purely to decide which cells to refine, but the actual refinement behavior is in some sense, specified *a priori*. Similarly, the commonly-used way to prevent AMR heuristics from continuing to refine beyond a computational limit is to manually specify a max refinement depth, again an a priori choice uninformed by the particulars of the solver or PDE, and specified everywhere. The DRL-AMR approach, in contrast, not only estimates which cells are likely responsible for a disproportionately large or small share of the total error but also estimates a stopping point, beyond which the solution smoothness indicates that additional decreases in error are likely to be marginal from the perspective of adding new unresolved features to the solution. This is because, during training, the RL agent is given global information related to available computational resources, which it balances against the expectation of change in the solution upon local refinement with respect to computational cost.

Where the stopping point occurs, in terms of the solution error, is controllable through the hyperparameter γ_c. A larger value of γ_c means a cheaper numerical solution with a greater aversion to incurring a cost. On the other hand, a small value of γ_c will result in a numerical solution with a lower error and less tolerance for interface jumps. Informally, γ_c is a user-specified setting indicating the trade-off between cost and accuracy, and is a way to inform the agent "how accurate" versus "how cheap" to make the solution. In Appendix A, we provide an approach to remove γ_c as a training hyperparameter entirely, and instead make it a user-chosen value during deployment by modifying the policy architecture; however, this discussion is beyond the scope of the main contributions of this thesis, and we train using a single value of γ_c for all of the numerical experiments in this section. Next, we explore the training and deployment behavior of the RL policy for this simple case to better understand the model.

5.4.1.1 Training and deployment considerations

Initialization. At the start of each training episode, we can either allow the RL agent to begin on the coarsest possible mesh (coarse initialization) or initialize the mesh to a random state by performing a random number of refinements.

Random initialization and coarse initialization perform similarly for the small problem sizes on which they are trained; however, per unit training time, we find that random initialization often



(a) DRL-AMR algorithm performance over training

(b) Mesh initialization

Figure 5-7: For the steady linear advection test in §5.4.1: (a) Performance achieved during training for different deep RL training algorithms, as measured by average episode reward (solid lines). Shaded bands indicate the rolling sample standard deviation of the episodic reward over a 10 episode window. (b) Performance of DRL-AMR models trained using random state initialization and coarse initialization, as measured by L^2-error per degree of freedom over 6 refinement cycles (on a 25-cell budget example and deployed with a 500-cell budget).

yields better policies when deployed on substantially larger problems. Figure 5-7b compares the performance of two DRL-AMR models trained for 10^5 training time steps, one with the mesh in a coarse state at the beginning of each episode, and the other with a mesh in a random state at the beginning of each episode. Both models were trained on the linear advection problem in §5.4.1 with a maximum budget of 25 cells but were deployed with a budget of 500 cells over six refinement cycles. Comparing the accuracy with the problem degrees of freedom, it is clear that the model trained with random initialization significantly outperforms the model with coarse initialization. Both models begin the refinement cycle on the same coarse mesh during deployment. The model trained with random initialization finds a solution of the same accuracy but uses only half the elements as the model trained with coarse initialization. We obtained similar results for other test cases we ran.

We hypothesize that random initialization during training leads to a more aggressive exploration of the action space and produces better results because initializing the mesh to a random state produces regions that are over-refined and under-refined. On the other hand, initialization on a coarse mesh tends to produce an under-resolved solution everywhere, leading the agent to become biased towards refinement during deployment. As might be expected, in the limit of long training times, performance becomes comparable between the two initialization strategies.

In light of these performance advantages, all results shown correspond to models trained using random initialization.

Learning algorithms. The performance—as measured by mean episodic reward—for the three RL policy architectures (DQN, A2C, and PPO, see §5.2.5) are shown in Figure 5-7a. The sample variance bands are found to tighten non-monotonically over the training duration, indicating more consistent performance. All three algorithms achieve similar mean reward and variance after roughly 25,000 training time steps, successfully solving the problem. However, as is common in RL problems, the model performance during training is non-monotonic, and it is advantageous to periodically save the most performant model state for deployment, rather than deploying the policy occurring at the end of the training window. The most performant model at any time during training is defined as the model with the highest mean episodic reward over a given lookback window–we use the SB3 library default of 500 training time steps. In general, we have not found any particular architecture to be consistently better in terms of performance over the set of test cases considered in this chapter. We note however that we have not yet attempted to accelerate the training process through optimization of training parameters such as batch size and episode length.

Model deployment. Unlike the AMR heuristic, the deployed DRL-AMR model finds solutions by apportioning computational resources over the mesh in a way that handles the smoothness/efficiency trade-off, then changes the mesh topology over the subsequent refinement cycles in order to increase accuracy. The DRL-AMR algorithm thus considers a richer set of strategies than a purely greedy strategy.

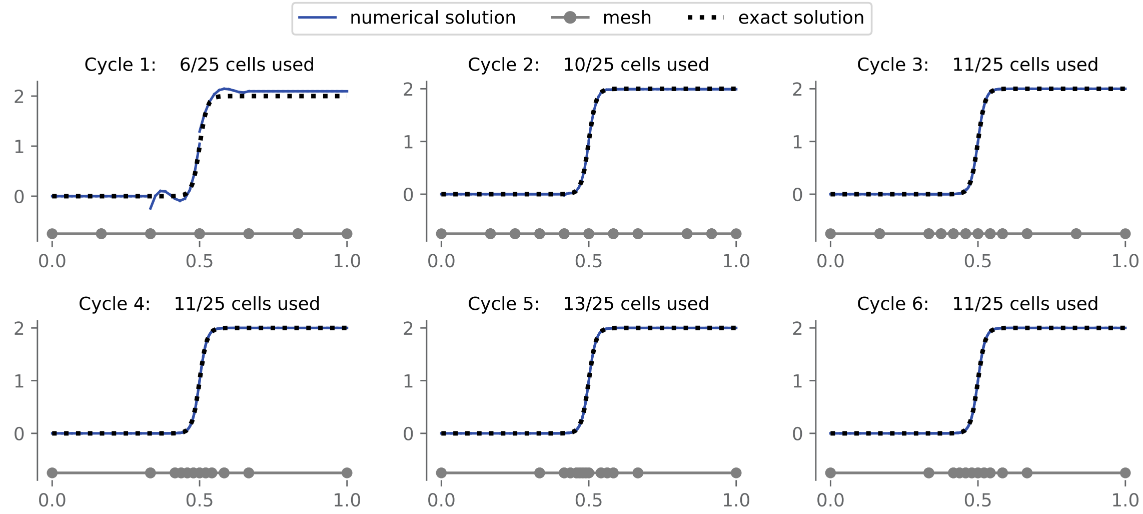


Figure 5-8: Steady 1D linear advection (§5.4.1). State of the numerical solution and mesh during the deployment of the deep RL agent AMR policy, over six refinement cycles.

To illustrate this, Figure 5-8 shows the mesh and resultant numerical solution over a set of six refinement cycles for an RL-agent trained in the same manner as in §5.4.1, but with a hyperparameter value of $\gamma_c = 100$, chosen to provide coarser solutions for the ease of visualization. Although cycles 2-6 all contain the order of 10 elements, the topology of their allocation over the domain changes to concentrate around the steep gradient region. This showcases a particular strength of the method: namely, that the decision of the number of elements to use during the exploratory cycles while preserving solution smoothness is delegated to the machine learning model, rather than manually needing to be specified. This allows the number of elements to increase and decrease exploratively, as opposed to AMR heuristics, which must increase or decrease the number of elements according to the error estimator or by a fixed number.

In practice, the cost of each cycle depends on the problem size but we find that the trained RL policies reach a converged mesh state in a few cycles, typically around 5-10, independent of problem size. This is in part because the number of cells can potentially double or halve every cycle, so locally under-resolved regions either become resolved very quickly, or the computational budget is approached and refinement recommendations become more conservative.

Introspection of the neural network model. As a deep RL policy is ultimately a neural network, we can query the trained neural network in order to visualize the policy suggestions for different inputs. Here, we are interested in visualizing the DRL recommendations (refine, no-change, or coarsen) as a function of generic solution properties. The input space to a policy network is the observation space and, in our case, it includes the numerical solution over the element. To sample this network, we could for example use the solutions on all elements and marginalize over these element solutions to create a map (decision versus solution properties). However, this sampling may not be sufficiently complete (limited by the element solutions used) and solution properties would need to be defined. To avoid to these issues, as discussed in §5.2.2.2, we could instead change the observation space and retrain a new DRL model for which we can easily sample the entire observation space and easily interpret results. Therefore, we train a new model using a reduced input observation space consisting only of the cell boundary jump, the mean boundary jump over all cells, and the current use of computational resources p. The purpose of this visualization is not to evaluate the performance of this simpler model but to show that such DRL-AMR networks are able to learn a sensible mapping between the observation space and the refinement recommendation.

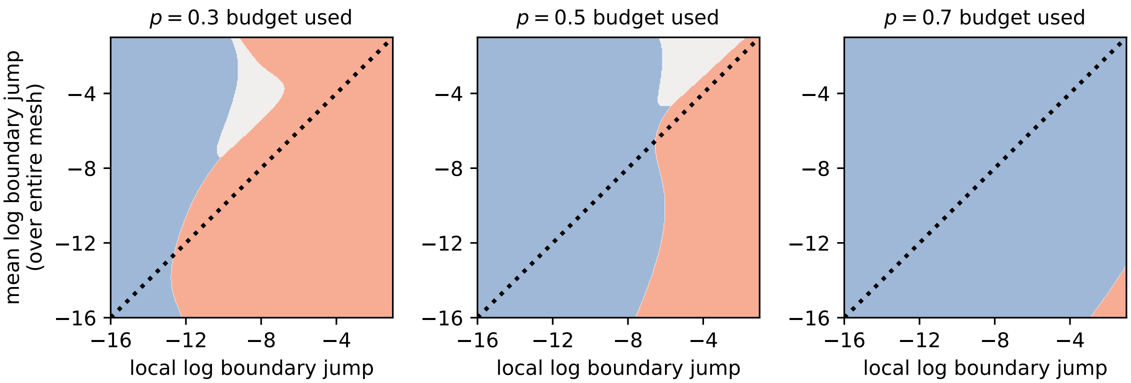


Figure 5-9: Samples of the policy recommendation of a newly trained DRL-AMR network for steady 1D linear advection (§5.4.1), as a function of a simplified observation space consisting of the cell boundary jump, mean jump, and current use of computational resources. Decision boundaries are shown between network recommendations to coarsen (blue), do nothing (grey), and refine (salmon). The dashed line indicates where the local boundary jump is the same as the mean cell-wise boundary jump over the entire mesh.

Figure 5-9 shows the recommendations of the simplified model trained on the problem in §5.4.1 using a value $\gamma_c = 25$. We consider a range of [-16, -1] in log space, as these are the range of jump values encountered by the network for this test case; recommendations outside of this region are uninformed by data encountered during training. Each sub-figure shows the decision boundary regions corresponding to the action recommended by the policy. At low use of computational resources ($p = 0.3$), the model suggests mostly refinement, even in regions where the observed boundary jump is significantly lower than the average over the entire mesh. We hypothesize that this corresponds to exploratory refinement in some sense. At moderate use of computational resources ($p = 0.5$), the model provides more conservative recommendations, suggesting refinement where the local boundary jump exceeds the mesh mean and is not less than 10^{-8} in terms of magnitude. When computational resources become scarce ($p = 0.7$), the model suggests coarsening except in regions where the local log boundary jump is much greater than the mean. The decision boundaries are highly nonlinear, even for this simplified observation space. This demonstrates the potential power and flexibility of the DRL-AMR approach, as it learns complex relationships between any physical relevant information included in the observation space and the utility of coarsening or refining.

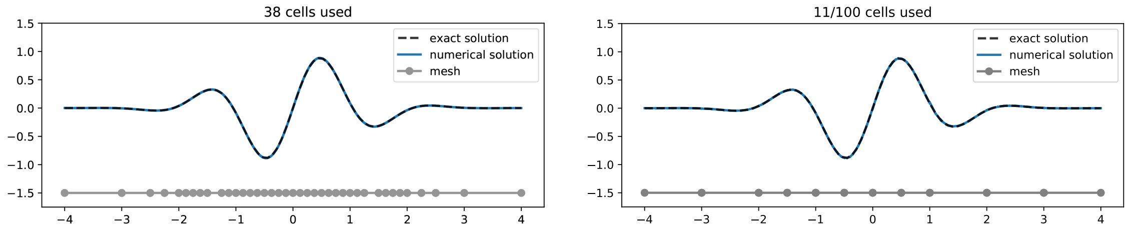
5.4.2 Generalization to different boundary conditions and forcing functions

We deploy the same trained DRL-AMR model in §5.4.1 to the same PDE, but on a different domain $\Omega = (-4, 4)$, with a different set of boundary conditions and forcing. We choose the boundary conditions g_D at the inlet and forcing function f such that the exact solution takes the form

$$u(x) = \sin(nx) \exp\left(-\frac{1}{2}x^2\right).$$

At the time of model deployment, we give the trained RL policy a budget of 100 elements. We evaluate the model over 6 AMR cycles this time using an AMR heuristic which makes use of the Kelly error indicator (Table 5.1, [122]) and implements a `fixed`(0.5, 0.1) strategy, as described in §5.3.4.

The final mesh and numerical solution for both is shown in Figure 5-10. Our findings are similar



(a) AMR heuristic

(b) RL Agent

Figure 5-10: DRL-AMR model trained on the test case in §5.4.1 but deployed on the problem in equation (5.4.2). Numerical solution with the mesh resulting from 6 cycles of refinement, using the Kelly error indicator as an AMR heuristic and the RL policy resulting from training. The exact solution is overlaid in both cases for comparison.

to the test case in the previous section. Both methods find a mesh and corresponding numerical solution that very closely matches the exact solution, however, the RL policy finds a coarse mesh on which the solution is well represented and stops refinement past the fourth refinement cycle. Similar to Figure 5-6, after the six refinement cycles, the RL policy returns a mesh that provides strictly better accuracy per problem degree of freedom (not shown).

This highlights the generality of the method; the RL policy did not suggest refining close to the center of the domain as it did for the example in §5.4.1. This is to be expected, as the RL agent is never given global location information during training, only local cell information along with the surrounding interface jumps; therefore it's impossible to over-fit the RL agent to a specific training example during training. However, we remark that if the agent were trained against a pathological example such as the Weierstrass function, where the correct action could always be to refine, this will be reflected in the trained model, although this can hardly be considered overfitting. Because the agent learns to relate features of the PDE and local jumps to the local smoothness of the solution, we find that the agent generalizes well across different test cases from the same PDE. This is an important characteristic of the approach, as we wish to deploy the trained model on problems of interest other than the subset on which the model was trained.

5.4.3 Unsteady 1D linear advection

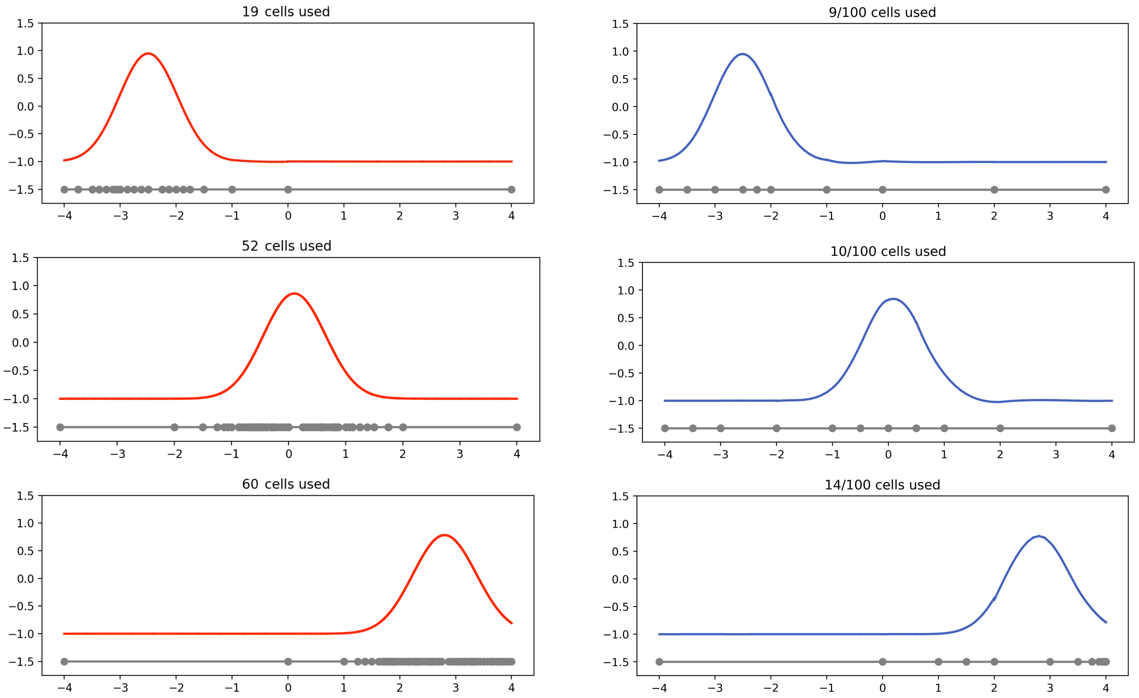
As an extension to the steady experiments, we consider the time-dependent Sommerfeld wave equation

$$\begin{aligned} \frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} &= 0 \qquad \text{in } \Omega \times [0,T], \\ u(x,t) &= g_D(t) \qquad \text{on } \Gamma_{\text{in}}, \\ u(x,0) &= u_0 \ , \end{aligned}$$

on the computational domain $\Omega = (-4, 4)$. The Dirichlet boundary condition $g_D(t)$ at the inlet is chosen according to the exact solution to the time-dependent wave equation $u_{\text{exact}}(x,t) = u_0(x-t)$. We consider numerical solutions at a polynomial order $p_{\text{order}} = 3$.

Gaussian pulse: training and deployment. In the first example, we use a Gaussian initial condition $u_0 = \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$. The scalar constants are $\mu = -4$ and $\sigma^2 = 0.25$, and the background velocity field is taken to be $c = 1$. The outlet boundary of the domain is taken to be an outflow condition.

We train the DRL-AMR model for $5 \cdot 10^4$ RL time steps using the time-dependent solver and training procedure as detailed in §5.2.4 with a computational budget of 25 elements and using a scaling factor value $\gamma_c = 100$ to render a relatively coarse numerical solution (5.2), as the computational cost is more heavily penalized. All other training parameters assume the default values (§5.3.5). For the AMR heuristic, we employ a gradient indicator (Table 5.1), that approximates the gradient of the numerical solution to estimate the error. Given the fast-decaying tails of the Gaussian pulse



(a) AMR heuristic

(b) RL policy

Figure 5-11: Unsteady 1D linear advection: Gaussian pulse. Numerical solution with time-dependent AMR policies using a gradient-based heuristic (left column) and the trained RL policy (right column), at times $T = 1.6$ (top), $T = 4.2$ (middle), and $T = 6.9$ (bottom).

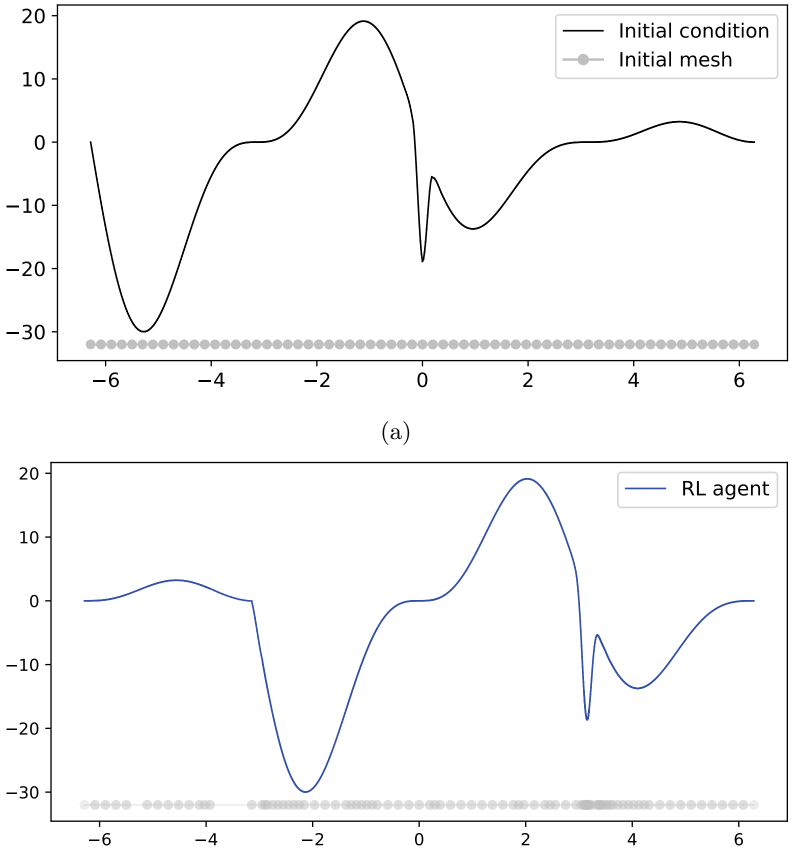
being advected, the gradient-based refinement indicator can be expected to accurately recommend coarsening outside of the pulse.

During deployment, we allow the AMR heuristic and the RL policy to perform one cycle of refinement/coarsening at every time step before advancing the numerical solution in time. We consider an increased budget of 100 elements for the RL-agent, and for the AMR heuristic we perform a bulk-refinement approach where we refine the cells responsible for the top 50 percent of the numerical error as measured by the gradient indicator, and coarsen the bottom 50 percent of the numerical error as measured by the same, *i.e.*, a `bulk`(0.5, 0.5) strategy. Using a time step $\Delta t = 0.01$, we simulate to a final time of $T = 7$. The results are shown in Figure 5-11. Similar to the other experiments, we find that both algorithms are able to capture the advection of the initial condition; the RL policy does so using a computational mesh with far fewer elements than the AMR heuristic. Namely, the DRL-AMR model is able to preserve the features of the solution while using fewer than 25 percent of the computational cost incurred by the AMR heuristic. Due to the increased value of γ_c as compared to the other numerical experiments considered thus far, the RL agent is more tolerant of a lack of smoothness in the solution. This is reflected in the small discontinuities of the numerical solution at the location $x = 0$ at $t = 1.6$ and at $x = 2$ at $t = 6.9$.

Deployment on multi-feature wave advection. We showed in §5.4.2 that the trained policy was able to generalize beyond its training setup in terms of boundary conditions and forcing functions for the steady advection problem. To highlight the same in this time-dependent case, we use the policy network trained on the time-dependent wave equation example (Figure 5-11), but deploy it on a more feature-rich initial condition

$$u_0 = \left(\alpha_1 e^{-\frac{1}{2\sigma^2}}(x - \mu) + \alpha_2 \sin(\alpha_2 x)\right)(x - 2\pi) \tag{5.10}$$

with parameters $\alpha_1 = 3$, $\alpha_2 = 2$, $\mu = 0$, and $\sigma = 2/25$, a different background velocity $c = \pi$, and a larger budget of 350 cells.



(a)

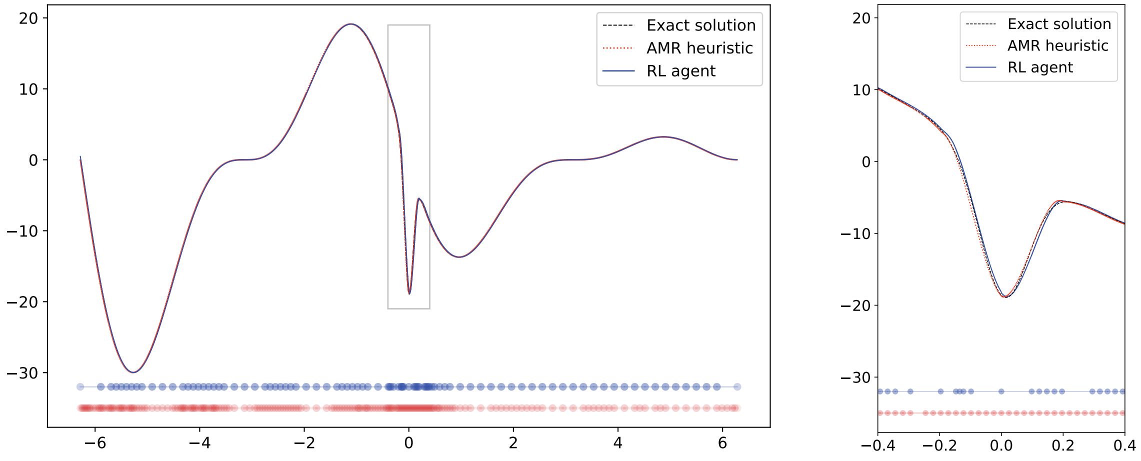
(b)

Figure 5-12: Unsteady 1D linear advection: multi-feature wave. (Left) Initial condition (5.10) and initial mesh of 64 elements. (Right) Numerical solution at time $t = 1$, integrated using RL-agent (94 active cells) trained on the Gaussian pulse. The mesh is shown with transparency for ease of visualization.

The domain and background velocity field are chosen as $\Omega = [-2\pi, 2\pi]$ and $c = \pi$, respectively, such that the exact solution is advected to its initial profile at the final time $T = 4$. To ensure the numerical error is primarily due to spatial discretization rather than temporal error, we use for time-marching a fourth-order explicit Runge Kutta (LSRK4) scheme and a small time step $\Delta t = 1 \cdot 10^{-3}$. For spatial discretization, we use the discontinuous Galerkin scheme of §5.3.2 with a polynomial order $p_{\text{order}} = 3$. We compare the performance of the RL-agent to an AMR heuristic using the Kelly error estimator and a `bulk`(0.5, 0.3) strategy.

Numerical results are shown in Figure 5-12 and Figure 5-13. Figure 5-12a shows the initial condition and the mesh of 64 elements on which the AMR heuristic and RL agent are initialized. Figure 5-12b shows the numerical solution and mesh given by the RL policy network at $t = 1$. Qualitatively, the features of the initial condition have been well-preserved, despite the use of only 27% of the active cell budget; the L^2-errors at the final time were $3.13 \cdot 10^{-2}$ and $3.21 \cdot 10^{-2}$ for the AMR heuristic and the RL-agent solution, respectively. To provide a more thorough comparison, Figure 5-13 compares the numerical solutions provided by the AMR heuristic and the RL agent at the final time $T = 4$; both demonstrate agreement with the exact solution–however, the RL-agent does so using far fewer elements. The RL-agent uses only 92, fewer than half of the 210 cells used by the AMR heuristic.

Although both approaches appear to prioritize refinement in roughly the same regions, by ex-



(a)

(b)

Figure 5-13: Unsteady 1D linear advection: multi-feature wave. (Left) Numerical solutions and corresponding meshes at the final time $T = 4$ for the RL-agent trained on the Gaussian pulse and for the AMR heuristic, along with the exact solution. (Right) Zoomed-in center region corresponding to the box in (a).

amining the fast-varying feature in the center region, depicted in Figure 5-13b, we see that the RL agent is more selective about refinement, even in local regions with sharp features. To explain this, we remark that the reward function is formulated such that the agent learns a map between a local observation and the expected utility of refinement; therefore, even if the agent is in an area of "interesting" activity, the numerical solution on a particular element may suggest that there's diminishing local utility to performing refinement. Moreover, the use of neighbor cell information, as well as the convective velocity in the observation space can result in anticipatory behavior in the recommendations, whereas the AMR heuristic will not.

We emphasize that due to the local nature of the observation space, over-fitting to a particular feature encountered during the training example(s) is not something to be concerned with, as the agent never has access to the solution as a whole, and the same feature will be covered with different numbers of cells over coarsening and refinement. Ultimately, the RL agent is primarily concerned with learning the relationship between a local signature of the numerical solution and the conformity of the solution; this is why the agent performs well at preserving solution features. In that sense, the DRL-AMR approach is advantageous in that applying it should not be particular to a specific PDE or numerical scheme, a claim we investigate in the next section.

5.4.4 Generalization to different PDEs

To demonstrate that the DRL-AMR approach is not specific to the advection equation considered above, we apply it to the second-order Poisson equation, a subset of advection-diffusion PDEs (§5.3.3).

$$-\frac{\partial^2 u}{\partial x^2} = f \qquad \text{in } \Omega = (-1, 1), \tag{5.11}$$

with mixed boundary conditions consisting of a Dirichlet condition $u = g_D$ on boundary $\Gamma_D = \{-1\}$ and Neumann condition $\nabla u \cdot \boldsymbol{n} = g_N$ on boundary $\Gamma_N = \{1\}$. The boundary conditions and forcing function are deduced from the exact solution, chosen to be a superposition of Gaussian functions

$$u = \sum_{i=1}^{3} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - r_i)^2}{\sigma^2}\right) \tag{5.12}$$

with parameters $r_1 = -1/3,\ r_2 = 0,\ r_3 = 1/3$ and $\sigma = 1/10$. We discretize the problem using HDG finite elements as described in §5.3.3, with polynomial order $p_{\text{order}} = 3$. We set the diffusivity coefficient $\kappa = 1$ and the convective velocity $c = 0$.

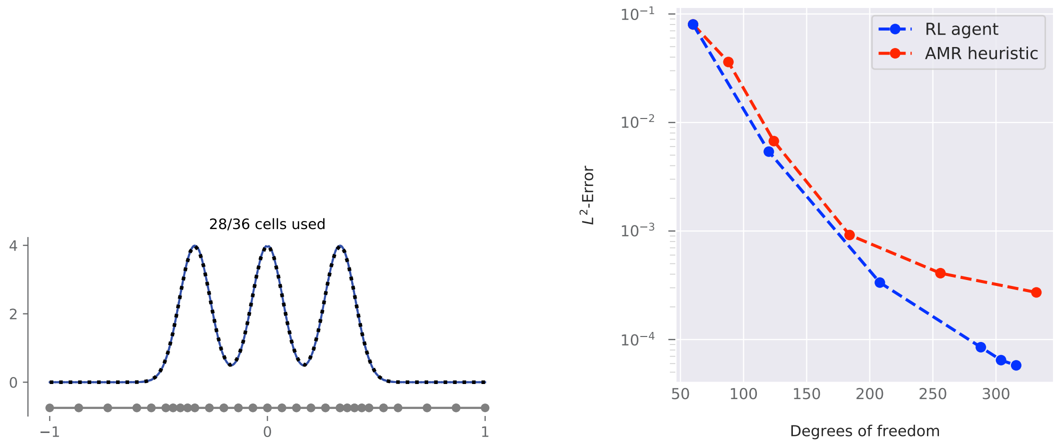


Figure 5-14: Steady 1D Poisson problem (5.11). (Left) Mesh distribution and corresponding numerical solution resulting from 6 cycles of refinement using the trained RL policy with a budget of 36 cells; the exact solution (5.12) (black, dashed line) is overlaid for comparison. (Right) L^2-Error of the numerical solution vs problem degrees of freedom over 6 mesh refinement cycles for the RL agent and the Kelly error estimator as AMR heuristic.

We train an RL model for $2 \cdot 10^5$ training time steps, and all non-specified training parameters assume the default values (§5.3.5). Results of the trained RL policy are shown in Figure 5-14. We see that the features of the solution are well resolved, and match the exact solution.

Furthermore, the trained policy identifies regions on the outer edges of the Gaussian mixture as in need of refinement; we hypothesize that this region is specifically sensitive to Gibbs phenomena caused by the change from a near-zero solution to a non-trivial one. The more precise comparison between L^2-error and problem degrees of freedom in Figure 5-14 shows that the DRL-AMR model outperforms the AMR heuristic. As the Kelly error indicator was specifically designed for this type of Poisson problem and can be referred to as an error estimator in this context, the fact that the RL policy is competitive is an encouraging result.

Lastly, the PDE in (5.11) is elliptic. The Green's function for the Laplacian operator decays as $1/r$, where r is the euclidean distance from the point in question, in contrast to the advection equation, where errors are advected along with the solution along characteristic paths. Therefore the numerical solutions as well as effective adaptive refinement strategies can be expected to be of a different character than those for purely hyperbolic problems [234]. The fact that the RL methodology was nonetheless able to satisfactorily solve the problem lends credence to the POMDP formulation and use of a local observation space during training.

5.4.5 Steady 2D linear advection

To examine the generalizability and performance in the context of larger, higher-dimensional problems, we solve the steady version of the linear advection equation (§5.3.2) on the two-dimensional (2D) spatial domain $\Omega = (0,1)^2$ with a circular, counter-clockwise convective velocity field $\boldsymbol{c}(x) = 1/\left\|x\right\|_2 (-x_2, x_1)$. On the inflow boundary Γ_{in}, we specify the boundary condition g_D according to the chosen solution,

$$u(r) = 1 - \tanh\left(\alpha(1 - 4(r - x_0))\right), \tag{5.13}$$

where r denotes the radius in cylindrical coordinates. We use the parameters $\alpha = 10$, $x_0 = 1/4$ and forcing function $f = 0$. The exact solution is a two-dimensional, cylindrical coordinate generalization of the test case considered in §5.4.1; that is, the solution contains a smooth, steep gradient. However, unlike the test case in §5.4.1, due to the higher problem dimensionality, the steep gradient is present both in the domain interior as well as at the domain boundary.

To train the network, we solve the linear advection problem

$$\begin{aligned} \nabla \cdot (\boldsymbol{c}u) &= 0, && \text{in } \Omega, \\ u &= u(r), && \text{in } \Gamma_{\text{in}} \end{aligned}$$

using the DG discretization given in §5.3.2 with elements of polynomial order $p_{\text{order}} = 2$, and deploy the model on the same problem. However, we train the RL policy network with a resource budget of only 110 cells, limiting the learning process to a relatively small number of elements compared to the deployment budgets; all other training parameters are the default values (§5.3.5). After $2 \cdot 10^5$ training, time steps, we deploy the trained RL policy over a series of 6 refinement cycles, starting with a coarse mesh of 25 elements with 5 elements in each direction.

We compare the performance of the RL agent to the gradient-based error indicator (Table 5.1) that uses a `bulk`(0.5, 0.5) refinement strategy. The AMR heuristic is given an effective budget of 3200 cells by limiting the maximum refinement depth; we compare this approach to RL policies with both larger and smaller budgets. Similar to the test case considered in §5.4.1, this AMR heuristic can be expected to perform well, given the steep gradient as the dominant feature of the exact solution.

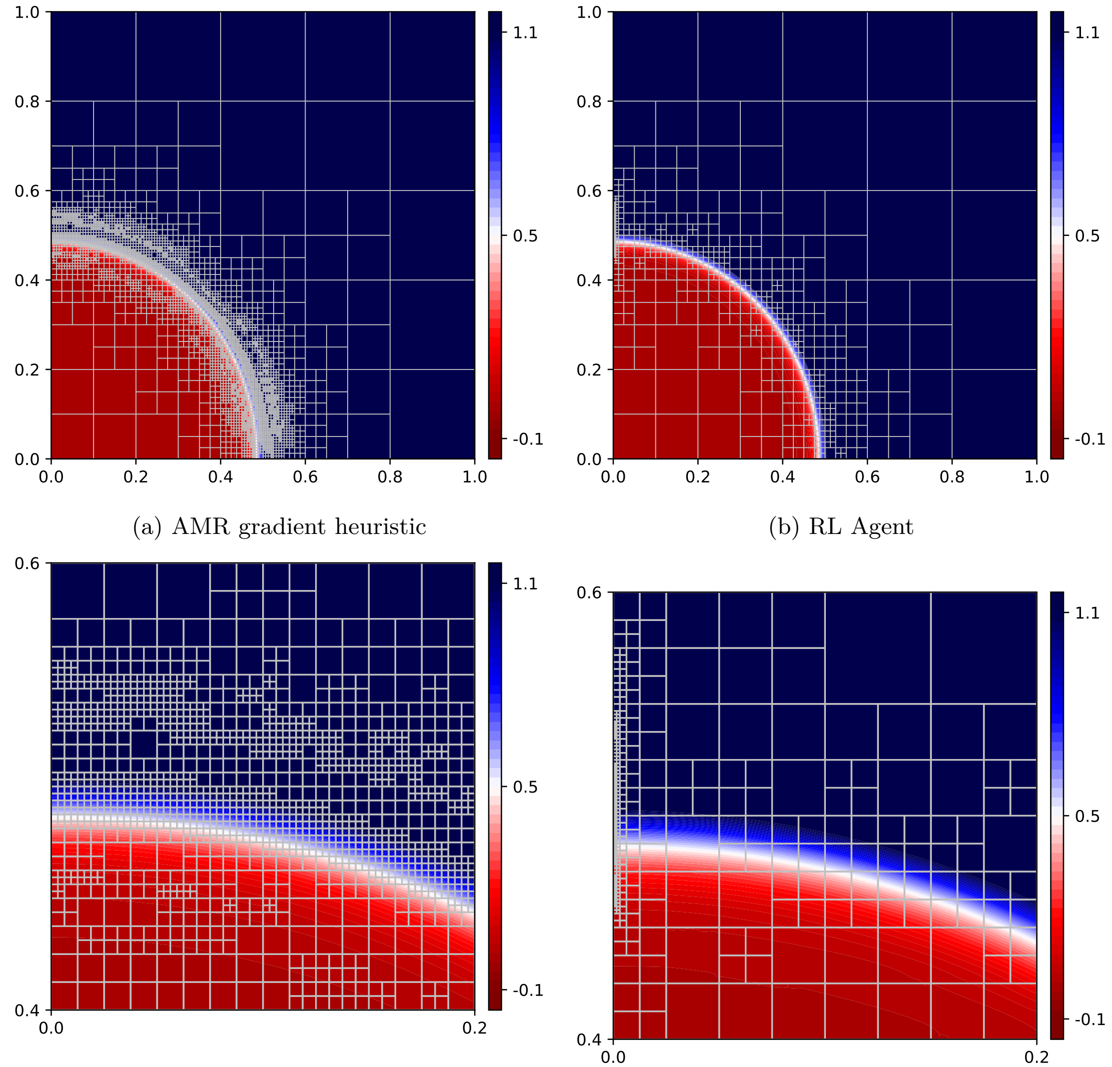
In Figure 5-15, we show the two numerical solutions overlaid with the final meshes and find that the RL policy is able to outperform the heuristic. The RL policy was deployed with a budget of 1000 cells; it makes use of roughly 2/3 of its budget allocation by the final refinement cycle. Qualitatively, both approaches are able to resolve the steep gradient; however, compared to the AMR heuristic, the RL agent has much more conservatively refined the mesh around the slope. Instead, the RL agent refines preferentially in the region where the steep gradient encounters the outflow boundary (*cf.* Figure 5-15c and Figure 5-15d). In training, the RL agent learned that the numerical solution is sensitive to the discontinuities arising where the gradient meets the outflow boundary. Although the agent has no information on where cells are located, the cells in this problem region exhibited behaviors in their observation space that indicated the numerical solution would improve by refining them. That is to say, the RL policy learned a non-trivial, spatially-heterogeneous strategy for regional refinement. In contrast, the gradient-based indicator detected the steep discontinuity but was not able to exploit the sensitivity of the error to the resolution close to the outflow boundary.

In Figure 5-16, we show the L^2-norm of the solution errors as a function of the number of degrees of freedom in the numerical discretization at each level of the refinement cycle, showing results for the RL policy deployed at different cell budgets, specifically 200, 1500, and 5000 elements. By the final refinement cycle, the RL policy finds a solution of comparable or better accuracy than that of the AMR heuristic for the same number of degrees of freedom. With a budget of 5000 cells, the RL agent remains well under budget but substantially outperforms the AMR heuristic. This is due to the aggressive refinement of the heuristic around the gradient region; after five refinement cycles, nearly all the mesh cells are located in this region, and an additional refinement cycle results in performance similar to a uniform refinement *i.e.*, nearly quadrupling the degrees of freedom. This poor performance of the AMR heuristic can be somewhat attributed to the nature of the problem, as discontinuous Galerkin methods for transport equations can be plagued by reduced convergence rates on arbitrary meshes, especially meshes which are not flow-aligned [41], as is the case in this example.

5.4.6 Steady 2D advection-diffusion equation

Section §5.4.5 involves a relatively simple steady numerical solution, with a single steep gradient region. It is natural to ask whether the methodology can capture solutions with more complicated features. In this section, we thus consider the steady advection-diffusion problem (5.8) with mixed boundary conditions, and the less straightforward hybridizable finite element discretization (5.9) of the same.

We choose a circular velocity field $\boldsymbol{c} = (x_2, -x_1)$ in the domain $\Omega = (-1, 1)^2$. The top and



(a) AMR gradient heuristic

(b) RL Agent

(c) AMR gradient heuristic (zoom, outflow boundary)

(d) RL Agent (zoom, outflow boundary)

Figure 5-15: Steady 2D linear advection. Numerical solution overlaid with the mesh refined using a gradient-based error indicator and the RL policy (1000 cell budget). (Top row) Entire problem domain. (Bottom row) Zoomed-in portion of the domain where the steep gradient meets the left outflow boundary.

bottom boundary conditions are Dirichlet Γ_D, whereas the left and right conditions are Neumann Γ_N. The boundary values and forcing function are deduced from the exact solution, a superposition of Gaussian functions chosen to create non-trivial features. The details of the exact solution are given in Appendix B. The problem is discretized at polynomial order $p_{\text{order}} = 4$ and makes use of element-local post processing, resulting in an observed optimal convergence order of $p_{\text{order}} + 2$ upon uniform mesh refinement (see [191]).

We train the RL policy for $3 \cdot 10^5$ training time steps, using a budget of 200 cells; all other training parameters are the default values (§5.3.5). As in the other numerical experiments, the training budget was chosen to be significantly cheaper than the deployment budget. For comparison, we consider an AMR heuristic that makes use of the Kelly error indicator with a `bulk`(50, 50) refinement strategy, refining and coarsening cells responsible for the top 50 percent and bottom 50 percent of the total estimated error, respectively. The AMR heuristic is given an effective budget of 3200 cells

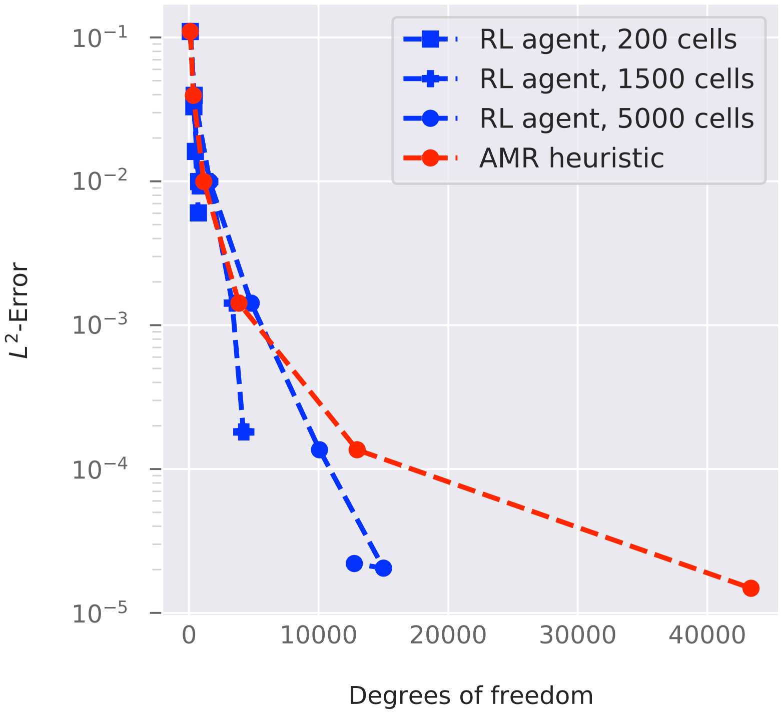
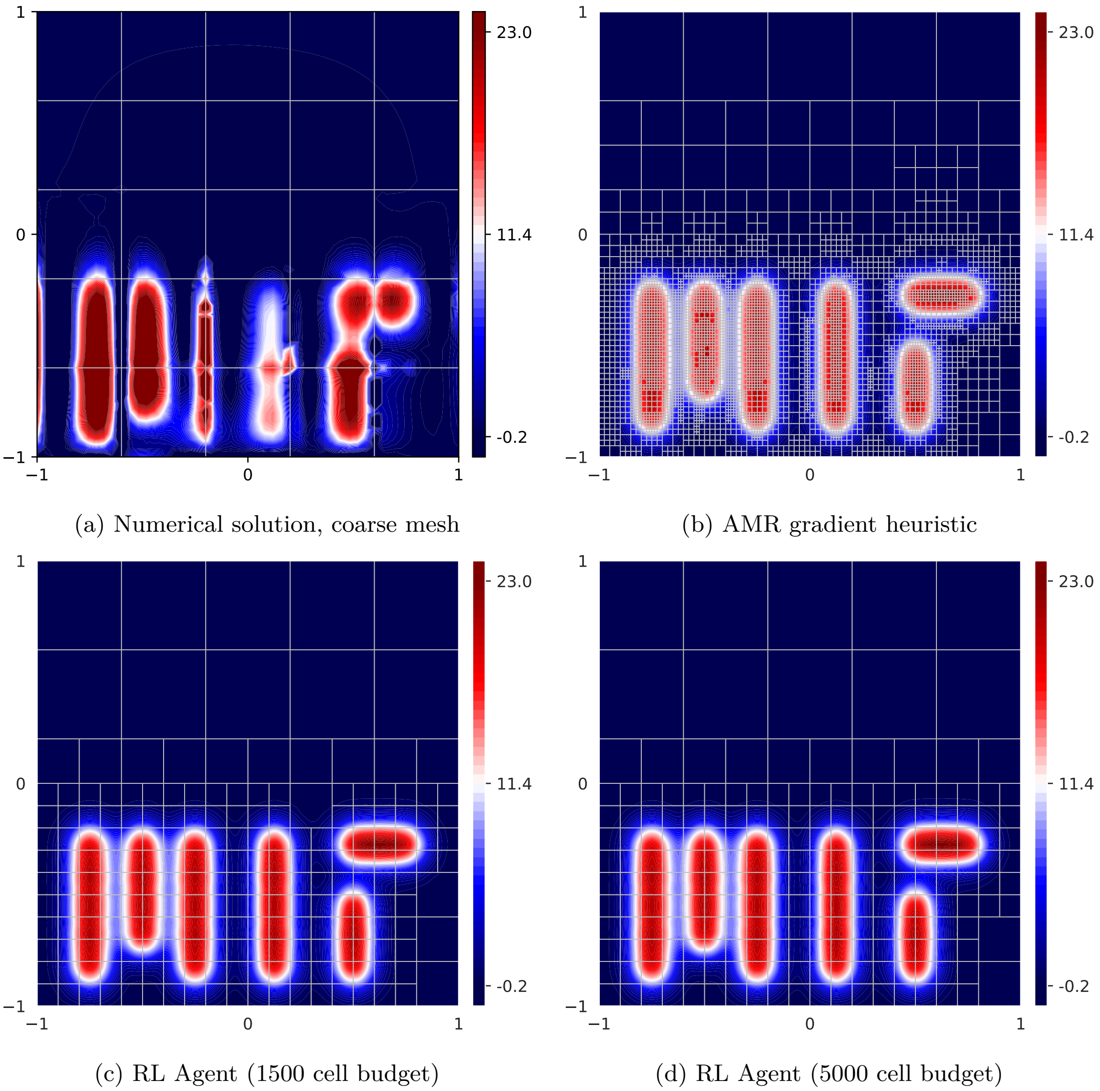


Figure 5-16: Steady 2D linear advection. L^2-error as a function of the number of degrees of freedom for the refinement cycles resulting from a gradient-based error indicator and the RL policy.

by limiting the maximum refinement depth, applying a cell limit as discussed in §5.3.4. We compare this approach to RL policies with both larger and smaller budgets; we choose larger and smaller budget as a comparison because, as is shown in the following, the RL-policy network recommends substantially fewer cells than the AMR heuristic in either case. Both strategies are employed over 5 refinement cycles, beginning from the coarse mesh shown in Figure 5-17a.

The results of the comparison between the two algorithms are given in Figure 5-17. Examination of the L^2-error per degree of freedom (not shown) again demonstrates that the RL agent is competitive with any of the AMR heuristics considered in this chapter. However, in this experiment, we focus on the relative ability of the heuristic and RL AMR strategies to resolve relevant features of the solution, pursuant to the discussion in §5.1.1. We see from Figure 5-17a that on the coarse starting mesh, the numerical solution is acutely under-resolved. Both the AMR heuristic (Figure 5-17b) and the trained RL agent (Figures 5-17c, 5-17d) are able to satisfactorily resolve the features of the numerical solution, but the RL agent does so with a more parsimonious allocation of elements over the mesh. Even when the computational budget is increased from 1500 cells to 5000 cells, the RL agent makes only minor adjustments to the mesh.

The explanation for this is that the trained DRL-AMR model makes decisions based on the learned relationship between the features of the observation space local to each element, and the change in the numerical solution upon refinement. The HDG finite element schemes (5.9) enjoy a faster convergence rate and, typically, better overall accuracy than the advection DG scheme (5.3.2) due to its dual formulation and element-wise post-processing, as discussed above. Additionally, we use a relatively high-order (sixth-order effective convergence rate) finite element scheme as compared to the smooth step example in §5.4.5, which exhibited second-order convergence. The RL policy is able to detect the fast convergence to the exact solution (according to the local numerical solution as well as the jump discontinuities over the boundary of each element) and elects to use only a small portion of its computational budget in order to resolve the solution, approximately 8 percent and 2 percent for budgets of 1500 cells and 5000 cells, respectively. We hypothesize that the inclusion of



(a) Numerical solution, coarse mesh

(b) AMR gradient heuristic

(c) RL Agent (1500 cell budget)

(d) RL Agent (5000 cell budget)

Figure 5-17: Steady 2D advection-diffusion. Numerical solutions overlaid with the corresponding final mesh.

additional HDG-specific features into the observation space, such as the numerical gradient $\boldsymbol{q}_h$ or the approximate numerical flux on the mesh skeleton $\hat{u}_h$ (5.9) could additionally improve performance.

In summary, the RL policy is able to judiciously allocate resources in order to capture non-trivial features of a numerical solution. Furthermore, the generality of the RL methodology extends to more complicated finite element schemes and is able to take advantage of their properties—in this case, high-order convergence due to a post-processed solution.

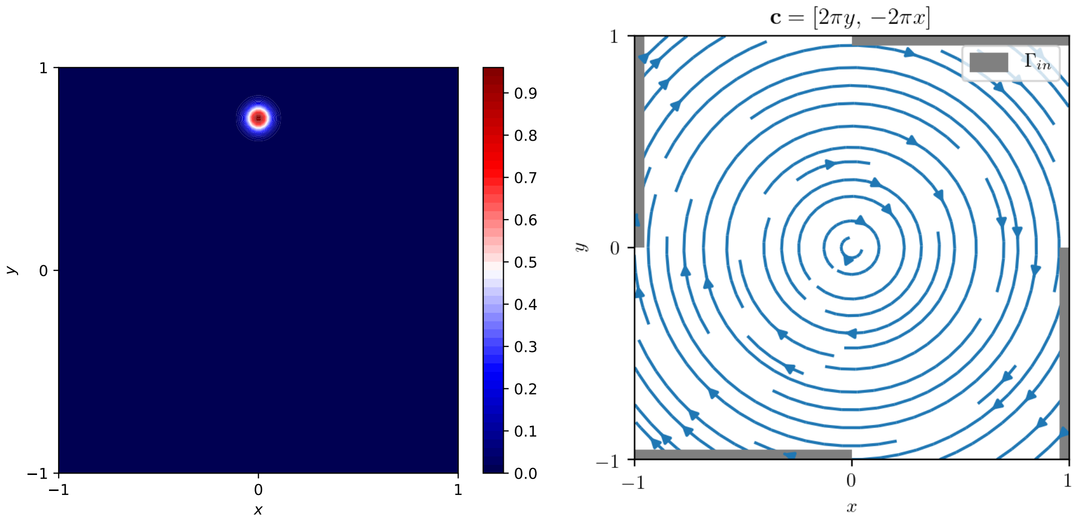
5.4.7 Unsteady 2D linear advection

In section §5.4.3, we showed that the RL policy, trained on a simple unsteady example, was able to perform well when deployed on a much more complex flow, with a different setup than that seen in training. We also showed that the complex features of the solution were preserved over many time integration steps. In this section, we extend these results to two dimensions: specifically, unsteady 2D linear advection problems (§5.3.2).

Unsteady 2D Gaussian pulse advection: training. For DRL training, we employ the advection of a simple Gaussian as illustrated in Figure 5-18, extending the Gaussian pulse of §5.4.3 to 2D. The initial tracer condition (Figure 5-18a) is

$$u_0 = \exp\left(-\frac{1}{2\sigma^2}((x-\mu_x)^2 + (y-\mu_y)^2)\right)$$

where $\mu_x = 0$, $\mu_y = 0.75$, and $\sigma = 1/25$. The convective velocity is $\boldsymbol{c} = (2\pi y, -2\pi x)$ over the region $\Omega = (-1,1)^2$, with numerical fluxes and upwind boundary conditions as specified in §5.3.2, specifically inflow boundary conditions $u_h = 0$ on Γ_{in} shown in (Figure 5-18b) and no boundary conditions imposed on the outflow boundaries Γ_{out} (§5.3.2). We train the DRL agent for $3 \cdot 10^5$ training time steps on a budget of 200 cells; all other training parameters assume the default values (§5.3.5 and Appendix C).



(a) Initial condition u_0

(b) Convective velocity $\boldsymbol{c}$

Figure 5-18: Unsteady 2D Gaussian Pulse advection: Training test case setup. The locations of inflow boundary conditions are highlighted in grey.

Unsteady reversible 2D ring advection: deployment. Once trained, we deploy and benchmark the DRL agent's performance against a more complex unsteady linear advection (§5.3.2). Inspired by the novel advection-diffusion test case in [136], we consider the advection of a thin ring by a reversible swirl flow (Figure 5-19). The initial tracer condition is given by

$$u_0 = \exp\left(-\frac{1}{2\sigma^2}\left(\sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0\right)^2\right), \tag{5.14}$$

specifying a ring with inner radius r_0 and approximate thickness 3σ centered at the point (x_0, y_0). For this test case, we choose the parameters $r_0 = 0.2$, $(x_0, y_0) = (0.25, 0.25)$, and $3\sigma = 0.05$, describing a thin ring centered in the bottom right-hand corner of the domain, depicted in Figure 5-19a. The velocity field is given by the reversible swirl flow

$$\boldsymbol{c}(x,y,t) = \left(\frac{3}{2}a(t)\sin^2(\pi x)\sin(2\pi y),\ -\frac{3}{2}a(t)\sin^2(\pi y)\sin(2\pi x)\right) \tag{5.15}$$

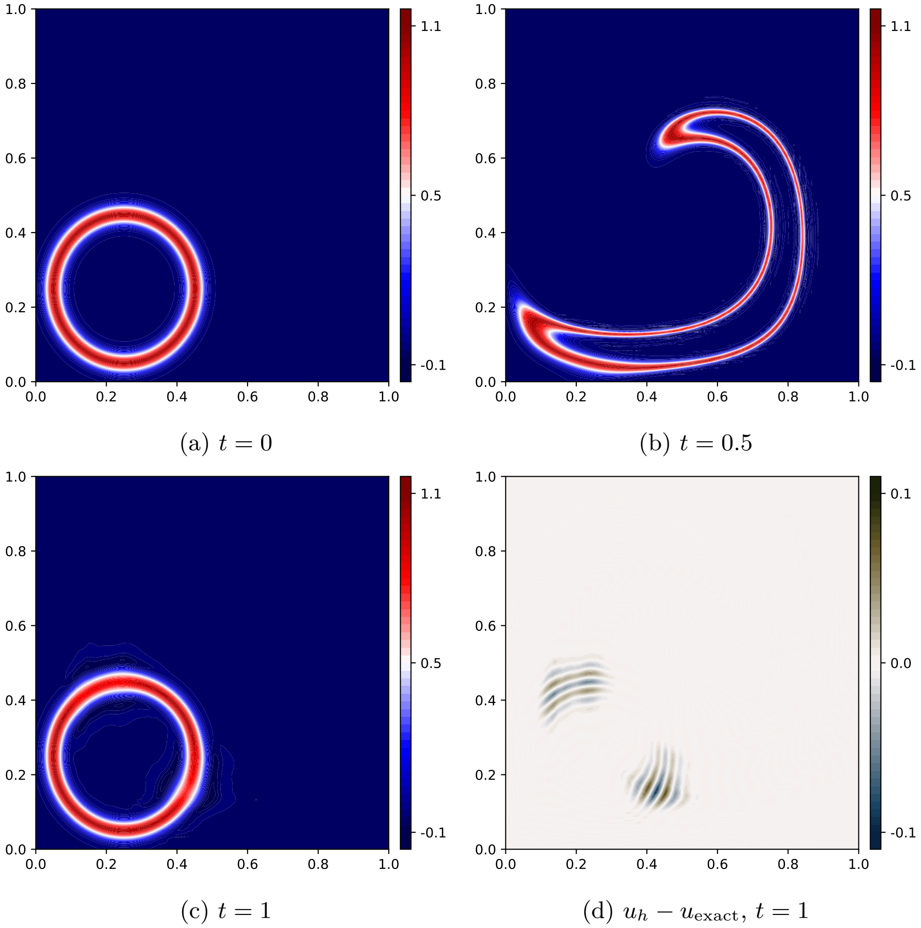
over the square domain $\Omega = (0,1)^2$. Here $0 \le t \le 1$, $a(t) = 1$ if $t < 0.5$, and $a(t) = -1$ if $t \ge 0.5$, which reverses the direction of the flow field at $t = 0.5$. At the time $T = 1$, the analytical initial tracer distribution returns to its initial location due to the flow symmetry and can be compared to the initial condition to measure the error in the numerical solution. The inflow boundary conditions and forcing function are taken to be zero over the duration of the simulation; however, the locations of the inflow and outflow boundaries switch over the course of the time integration. The presence of steep gradients in the solution along the inner and outer circumferences of the ring during both the forward and backward swirl advection process makes this problem challenging, as well as a suitable test case to assess the ability of a numerical scheme to avoid spurious diffusion and dispersion, and to preserve features of the analytical solution [136].

All numerical simulations are run at polynomial order $p_{\text{order}} = 3$ and use a small time step $\Delta t = 10^{-3}$ over the LSERK45 time-marching so that the total error is dominated by the spatial discretization error.

During deployment, we compare the trained DRL-AMR model to an AMR heuristic which uses the Kelly error indicator with a conservative `bulk`(60, 40) refinement strategy. Both strategies start on a fine, uniform mesh of 64×64 elements and are allowed 6 refinement cycles during initialization, after which, they are allowed one refinement cycle per time step for the duration of the simulation.

In order to prevent the number of degrees of freedom from growing unbounded over time integration for the AMR heuristic, we apply an effective resolution requirement by specifying a maximum refinement depth of 4 as discussed in §5.3.4, corresponding to the finest possible grid of 64×64 cells. To provide a fair comparison, the RL policy is deployed with an equal maximum cell budget of $64^2 = 4096$ elements.

For baseline comparison for the AMR heuristic and the RL policy, we show first in Figure 5-19 the numerical solution for a uniform grid of 64×64 elements, corresponding to the resolution limit for the AMR heuristic. Qualitatively, the uniform grid simulation is able to preserve the features of the ring over the forward and reverse advection, up to final integration time $t = 1$ (Figure 5-19c).



(a) $t = 0$

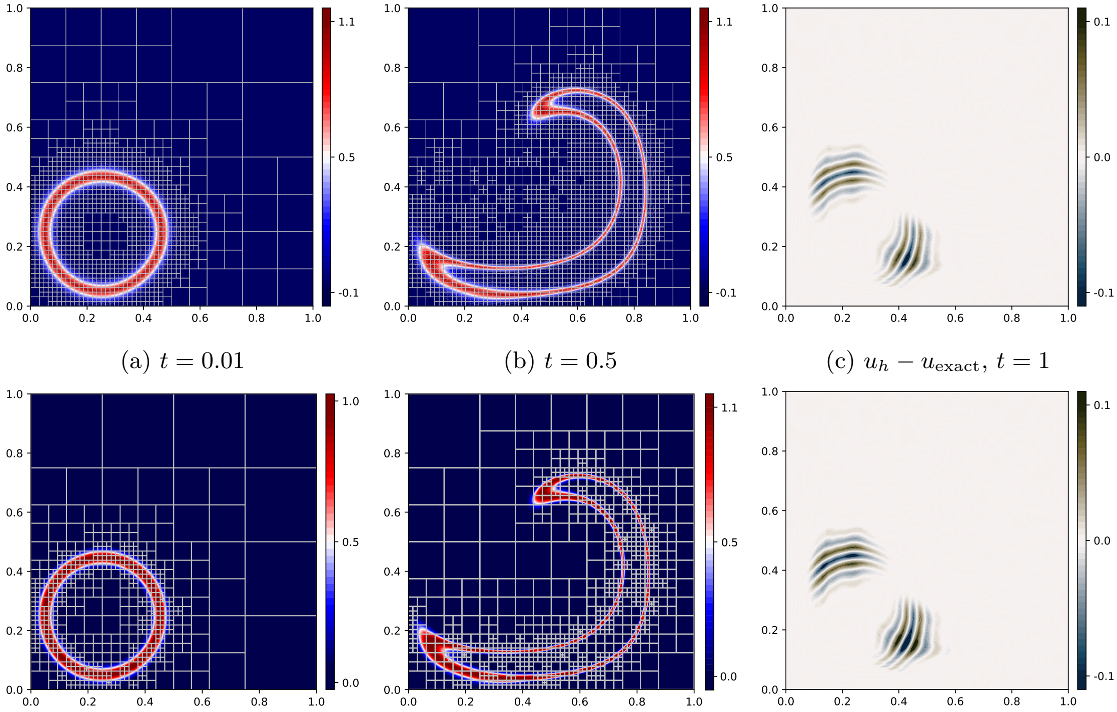
(b) $t = 0.5$

(c) $t = 1$

(d) $u_h - u_{\text{exact}}$, $t = 1$

Figure 5-19: Unsteady reversible 2D ring advection. Numerical solution on a 64×64 uniform grid.

The proposed meshes and resultant numerical solutions for the AMR heuristic and RL policy are shown in Figure 5-20. As can be seen in panels 5-20a, 5-20b, 5-20d, and 5-20e, the qualitative features of the numerical solution resulting from both schemes are very similar to that of the uniform grid simulation (Figure 5-19); that is, both approaches are able to successfully resolve the features of the solution. Similar to the other experiments, the RL agent is able to achieve comparable accuracy to the AMR heuristic, but with a comparatively coarser mesh, as is corroborated in Figures 5-20c and 5-20f. The numerical errors for each approach are comparable and within 10 percent of the numerical error resulting from the uniform mesh. However, in this particular test case, at each time step, the RL-agent used far fewer elements.



(a) $t = 0.01$

(b) $t = 0.5$

(c) $u_h - u_{\text{exact}}$, $t = 1$

(d) $t = 0.01$

(e) $t = 0.5$

(f) $u_h - u_{\text{exact}}$, $t = 1$

Figure 5-20: Unsteady reversible 2D ring advection. AMR heuristic (top row) vs DRL agent (bottom row).

The L^2-norm of the errors for the three numerical solutions at final time $T = 1$ are provided in Table 5.3 and the time series of active cells for both the AMR heuristic and the RL agent are shown in Figure 5-21. They confirm that the DRL-AMR agent can achieve the same accuracy for a much smaller cell budget.

Refinement method	L^2-error	Percent change
Uniformally refined mesh (4096 cells), ground truth	$1.8316 \cdot 10^{-2}$	-
AMR Heuristic (Kelly)	$1.9553 \cdot 10^{-2}$	6.8%
DRL Agent	$1.9630 \cdot 10^{-2}$	7.2%

Table 5.3: Unsteady reversible 2D ring advection. Comparison of numerical errors at final time $T = 1$. The evolution in time of the number of active cells is provided in Figure 5-21.

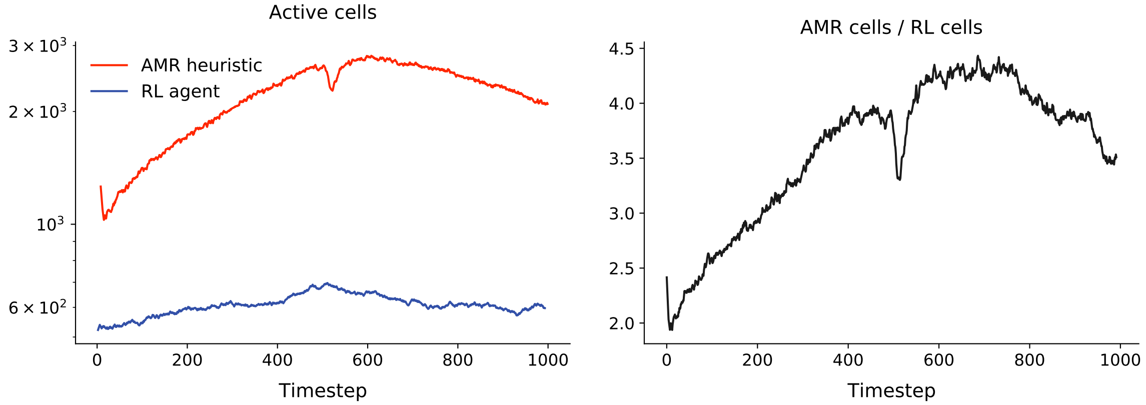


Figure 5-21: Unsteady reversible 2D ring advection. Number of active cells versus time during deployment (post-initialization).

5.4.8 Discussion of numerical experiments

The numerical experiments in §5.4.1-§5.4.2 demonstrated a proof of concept for the DRL-AMR approach; we were able to train a model on a small budget and apply it effectively to larger, different problems governed by the same PDE. Our experiments indicated that random initialization led to a better exploration of the decision space and that all learning algorithms considered were able to solve the problem. We saw that for a simplified observation space, the neural network representing the trained RL policy was interpretable in terms of local solution conformity. The numerical experiments in §5.4.3-§5.4.4 showed that the method generalizes to time-dependent problems, as well as to different PDEs with mixed boundary conditions and significantly more complicated numerical schemes.

The test case in §5.4.5 demonstrated that the RL agent was able to learn a spatially heterogeneous, non-trivial strategy to increase accuracy per cost by preferentially refining around a problem region on the boundary, a strategy on which the AMR heuristic was not able to capitalize. The advection-diffusion problem in §5.4.6 extended the findings in §5.4.4 to higher dimensions and exhibited the ability of the RL policy to take advantage of the fast convergence of a post-processed solution as well as to resolve non-trivial features of a numerical solution cheaply. Lastly, the unsteady 2D advection problem in §5.4.7 displayed the ability of the trained RL policy to preserve similar non-trivial features over the course of time integration in problems with dynamic, sharp gradients.

Overall, our DRL-AMR methodology is flexible and effective at delivering efficient, high-quality solutions for both static and time-dependent problems over a wide range of different PDEs, boundary conditions, dimensions, and problem sizes.

5.5 Conclusions and future work

In this chapter, we introduced a novel deep reinforcement learning formulation for adaptive mesh refinement based on a partially observable Markov decision process designed to balance numerical accuracy with computational cost, with the goal of providing a learned, high-quality strategy for resolving solution features efficiently. The underlying idea is that rather than hand-designing a heuristic error indicator *a priori*, the reinforcement learning agent will instead learn one through trial and error during training by solving many inexpensive problems. This is advantageous because the learned policy network can be arbitrarily complex and make use of any feature included in the observation space, including information about computational cost. Furthermore, training the RL policy requires no domain-specific knowledge, as the relevant information is encoded in the numerical solver, and, more abstractly, in the underlying PDE. Conversely, specific features of a numerical scheme such as polynomial degree, convergence order, and problem dimension are implicitly considered in training through use of the numerical solver, as opposed to having to be analyzed and explicitly specified in the case of a manually-defined error indicator.

Our implementation shows that the resultant trained policies are able to execute adaptive refinement strategies which are competitive with, and in many cases, better than common AMR heuristics in terms of accuracy of the final solution per problem degree of freedom. Our methodology is not specific to any particular PDE, spatial dimension, or numerical scheme, and can flexibly incorporate physical or temporal history data into the observation space. The local nature of the RL problems allows for training the RL agent on small problems and deploying the policy on much larger problems, ensuring scalability. Lastly, at no point during training nor model deployment do we ever make use of an exact solution or a "ground truth."

Future work could incorporate p-refinement into the learning process to allow the RL agent access to a richer set of finite element representations of the numerical solution, incorporate a more sophisticated belief distribution as to the regions of under- or over-resolution in the numerical solution, or integrate transfer learning using existing error estimators to accelerate training. Finally, the application of the new DRL-AMR schemes to vector-valued problems such as the Navier–Stokes equations, ocean equations, and atmospheric [127] applications, as well as studies of related numerical topics such as slope-limiting, preconditioning, and adaptive time integration, are the subject of ongoing research.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we derived and developed new numerical schemes to solve the hydrostatic and nonhydrostatic ocean equations, using high-order hybridizable discontinuous Galerkin (DG) spatial discretizations and advances in stability and robustness properties for projection-method temporal integration. We implemented the ocean models in a high-performance `C++` framework, and the model nesting within the MSEAS-PE hydrostatic ocean modeling and forecasting system. We verified the correctness and efficiency of the finite element models and conducted preliminary 2D and 3D model-nested simulations using realistic ocean data in the Alboran sea.

Using the new schemes, models, and software, we conducted exploratory investigations into use of data assimilation and Bayesian inverse problems using ensemble Kalman filtering and Markov chain Monte Carlo methods, respectively. We demonstrated that these approaches are viable for use with high-order discontinuous finite element solvers.

With the long-term goal of adaptive mesh refinement for ocean processes in mind, we developed a general and flexible methodology to learn new adaptive mesh refinement strategies using deep reinforcement learning. This approach couples state-of-the-art machine learning schemes and libraries with our custom numerical solvers, enabling the discovery of tailor-made AMR strategies for the numerical solution of PDEs using high-order DG methods. Altogether, these advances are aimed at the creation of next-generation nonhydrostatic ocean models.

6.2 Future work and outlook

The chapters of this thesis constitute advances in the field of high-order finite element methods, ocean modeling, and machine learning. However, the primary logical extension of this work is the combination of all the constituent advances into a single, integrated ocean modeling framework.

Incorporation of the data assimilation techniques and uncertainty quantification techniques explored in §4 into the nonhydrostatic ocean model developed in §3 is entirely straightforward, once a particular stochastic quantity of interest has been identified. The inclusion of ensemble Kalman filtering into ocean models is well-understood and requires little explanation. In the case of the Markov chain Monte Carlo methods developed in §4.3, we sought to invert an unknown spatially-variable diffusivity field using synthetic sparse observation data at certain sensor locations. The application to ocean modeling applications is immediate, as one of the most important parameters is the choice of the diffusivity in the ocean equations. Since all ocean models are limited in their spatial resolution, they are incapable of capturing physical dynamics occurring on length scales smaller than the element size. The choice of a numerical diffusivity in the ocean equations amounts to the replacement of the actual molecular viscosity with a much larger eddy viscosity, an approach referred to as *subgrid-scale parameterization* [56]. Even for nonhydrostatic ocean processes, the horizontal scales of the modeling domains tend to be much larger than those of the vertical, and we have treated the subgrid-scale parameterizations anisotropically, choosing both a vertical and horizontal diffusivity coefficient. Therefore, any simulation using the nonhydrostatic model in §3 involves the modeling choice of four diffusivity parameters: a horizontal and vertical eddy viscosity for both the momentum and tracer equations. Furthermore, as the eddy viscosities ought to depend on the grid dimensions, the inclusion of adaptive mesh refinement also induces a natural spatio-temporal dependence for these parameters. Areas of fine resolution should apply a different eddy viscosity than those with coarse resolution, especially when the capture of highly-localized phenomena such as internal solitary waves is of modeling interest. In general, more complex ocean turbulence and mixing models could be devised and implemented in the present HDG framework. Returning to the application of MCMC methods for Bayesian inversion of the diffusivity field, the most simple and natural application of these methods is a principled inference of these four parameters from observational data. Introduction of adaptive mesh refinement and unstructured meshes allows for the inference of spatially-variable or even dynamic diffusivity fields, extending §4.3.

Application of the deep reinforcement learning methods in §5 into an ocean modeling framework is less straightforward and constitutes the subject of ongoing research. The key question of interest is how to appropriately treat vector-valued finite element spaces. The jumps in the different solution variables could all be included in the observation space, but there is a question as to how to treat the relative scales and dynamic vector gradients of the solution. Additionally, this question affects the reporting of the global average of the jumps in the vector-valued solution. Neural networks are known to be sensitive to inputs that have a large difference in scales, as would be the case for most ocean applications. Many different heuristics to normalize or rescale the solution variables are possible, but it is the author's opinion that careful consideration is required to arrive at a principled, general method which allows all solution variables to be treated in a way that allows for the actual desired goal: to find locally under- and over-resolved regions. The free-surface seiche test case in §3, for example, has a constant density, allowing for a simultaneously over-resolved density perturbation field but an under-resolved velocity or free-surface. Therefore, a future research direction is to devise a method which allows for the detection of high-priority areas of refinement based on the underlying dynamics, while simultaneously handling large differences in the scales of the state variables.

The constituent advances in ocean modeling and machine learning made in this thesis also have their own possible independent extensions.

6.2.0.1 Ocean modeling

In the future, the present work will complement the suite of multidisciplinary simulation, estimation, and assimilation systems (MSEAS) methods and software [186], used for fundamental research and for realistic simulations and predictions in varied regions of the world's ocean [105, 104, 107, 150, 155, 164, 200, 215] Applications include monitoring [152], real-time acoustic predictions and data assimilation [68, 142, 148, 259, 3, 4], probabilistic forecasting [160, 158, 69], adaptive sampling and path planning [147, 153, 155, 151, 67], and ecosystem predictions and environmental management [55, 106]. Namely, the present results can be applied and developed pursuant to research in high-order multi-dynamics regional ocean modeling [243, 244, 246, 228]. These models can be employed for targeted non-hydrostatic and biogeochemical process studies.

Preliminary work has been done to extend HDG methods to a distributed computing environment [73, 79, 134, 224, 225] and a logical step for future work involves adapting the matrix-free approaches and implementations of the present results to a GPU environment. The fast application of the sparse HDG matrix-vector product in particular may be amenable to GPU acceleration, perhaps in conjunction with an HDG-specific graph-coloring approach similar to that discussed in [225] to provide further vectorization. Numerical experiments conducted in tandem with the work in this thesis conclusively demonstrate that the dominant computational cost of HDG-FEM methods for the incompressible Navier–Stokes equations and ocean equations discretized with projection methods is the solution of the elliptic pressure correction equation, a result corroborated by similar experiments in the ocean modeling literature [178, 251]. Geometric multigrid (GMG) methods are known to be solvers with optimal complexity properties in the number of total unknowns [38, 100]. However, the same element-to-skeleton parameterization that makes HDG-FEM advantageous presents a distinct weakness in terms of devising efficient GMG methods. As of the time of this writing, optimal-complexity GMG methods for HDG schemes remain under development [134]. Should this line of research prove fruitful, the application of high-performance GMG methods to the linear systems arising from the discretization of these typically ill-conditioned pressure correction systems would confer significant computational advantages. In turn, both numerical and computing improvements to HDG-FEM scalability could be employed for multi-scale multi-dynamics ocean and fluid modeling [62, 161].

While not used in this thesis, a common technique applied to advection-dominated problems in fluid physics is the use of slope limiting to provide additional numerical stability for DG finite element methods [110]. However, this approach is largely heuristic, as it involves filtering out high-frequency modes from the nodal solution. As these high-frequency modes may be either spurious or physical, slope limiting can lead to substantial losses in accuracy. Competing techniques inspired by the stabilization approaches used for advection-dominated flows modeling using classical continuous Galerkin finite element methods have recently been applied with success to DG schemes for the incompressible Navier–Stokes equations [77, 78]. These methods involve the use of mixed polynomial spaces for the discrete velocity and pressure, as well as the inclusion of penalization terms derived from the particular choices of numerical flux to discourage the growth of numerical energy over long time integration. A sensible future direction would involve a comparison between slope limiting methods and these more modern techniques in the interest of achieving the best balance of accuracy and stability in under-resolved regimes, and the conclusions could be of interest to the DG-FEM modeling community more broadly.

Aside from the underlying numerical methods, the work completed in §3 has additional application to the validation of hydrostatic models and identification of dynamical regimes in which nonhydrostasy plays a crucial role. Conversely, the model will enable investigations into accurate nonhydrostatic parameterizations natively employable in hydrostatic models such as the MSEAS-PE forecasting system. The software developed over the course of this thesis has already been extended and applied to idealized hydrostatic-nonhydrostatic multi-domain modeling [228]. These developments offer the future possibility of a two-way nested hydrostatic-nonhydrostatic model, where the nonhydrostatic HDG finite element model is nested inside of a larger DG-FEM hydrostatic model, with boundary conditions to the DG-FEM hydrostatic model provided by the MSEAS-PE model. The advantage of such a set up over that considered in this thesis is a fully two-way nesting, whereby the nonhydrostatic and hydrostatic models communicate with each other dynamically. Such realistic multi-dynamics simulation capabilities with Bayesian data assimilation and learning [98, 170] constitute the subject of ongoing research.

6.2.0.2 Machine learning

In the author's opinion, machine learning methods are better-suited to improving heuristic aspects of numerical PDE solvers than to the task of replacing them directly. As alluded to in §5.1, this is because the convergence, consistency, and stability guarantees provided by numerical methods are essential to research confidence in the output of any numerical model. In contrast, attempting to directly learn the nonlinear mapping between boundary conditions and forcing functions and the solution to a partial differential equation using the machinery of deep learning requires solving a highly non-convex optimization problem in which convergence is not guaranteed. Preliminary work has started in this area [213] but much remains to be done. On the other hand, modern machine learning techniques have shown themselves to be effective at learning effective heuristics and patterns from data alone, in the absence of a model. In §5, this idea was explored at length for adaptive mesh refinement; however, there are many such heuristics auxiliary to the direct application of numerical schemes which could conceivably provide substantial benefits to the accuracy and efficiency of the numerical solution of PDEs. Even relatively mature technologies such as matrix multiplication have proven to be amenable to acceleration by using deep reinforcement learning to find faster, more efficient computational strategies [76]. To that end, fruitful areas of future research involve the application of deep reinforcement learning to the problems of preconditioning, adaptive time stepping, slope limiting, and avoidance of CFL limitations. The replacement of human tinkering and adjustment with machine-learning-discovered policies for any one of these applications would constitute a substantial advance in the field of numerical modeling.

Chapter 7

Appendices

7.1 Useful identities

The traces $u^{\pm}$ are defined on $K^{\pm}$ as

$$u^{\pm} \equiv \lim_{\epsilon \to 0} \left(\boldsymbol{x} - \epsilon \boldsymbol{n}^{\pm}\right).$$

Since $\boldsymbol{n}^{\pm}$ points outward, the limit of $\boldsymbol{x} - \epsilon \boldsymbol{n}^{\pm}$ has the interpretation of the value of $u(\boldsymbol{x})$, approaching the boundary of K from its *interior*.

7.1.0.1 Jump relations

$$\frac{1}{2}[\![u_h^2]\!] = \frac{1}{2}\left((u_h^-)^2 - (u_h^+)^2\right) = \frac{1}{2}(u_h^- + u_h^+)(u_h^- - u_h^+) = \{\{u_h\}\}\,[\![u_h]\!] \tag{7.1}$$

$$\{\{u_h\}\} + \frac{1}{2}[\![u_h]\!] = \frac{1}{2}(u_h^+ + u_h^-) + \frac{1}{2}(u_h^- - u_h^+) = u_h^- \tag{7.2}$$

$$[\![uv]\!] = \{\{u\}\}\,[\![v]\!] + [\![u]\!]\,\{\{v\}\} \tag{7.3}$$

The following relation is useful computationally since it avoids negation on each edge to distinguish between the normal vectors $\boldsymbol{n}^+$ and $\boldsymbol{n}^-$, using instead that $\boldsymbol{n}^+ = -\boldsymbol{n}^-$:

$$[\![u_h \boldsymbol{n}]\!] = u_h^+ \boldsymbol{n}^+ + u_h^- \boldsymbol{n}^- = \left(u_h^+ - u_h^-\right) \boldsymbol{n}^+. \tag{7.4}$$

7.1.0.2 Flux relations

Upwind fluxes are defined as

$$\widehat{\boldsymbol{c}u_h} \cdot \boldsymbol{n}^+ = \begin{cases} (\boldsymbol{c} \cdot \boldsymbol{n}^+) u_h^+, & \boldsymbol{c} \cdot \boldsymbol{n}^+ \geq 0 \\ (\boldsymbol{c} \cdot \boldsymbol{n}^+) u_h^-, & \boldsymbol{c} \cdot \boldsymbol{n}^+ < 0 \end{cases} \tag{7.5}$$

loosely "take the value from where $\boldsymbol{c}$ is blowing". This would require logic to compute; but is equivalent to the computationally more efficient (for vectorization)

$$\frac{1}{2}\left(\boldsymbol{c} \cdot \boldsymbol{n}^+ + |\boldsymbol{c} \cdot \boldsymbol{n}^+|\right) u_h^+ + \frac{1}{2}\left(\boldsymbol{c} \cdot \boldsymbol{n}^+ - |\boldsymbol{c} \cdot \boldsymbol{n}^+|\right) u_h^- \tag{7.6}$$

which can be easily verified as equivalent by considering the positive and negative cases of $\boldsymbol{c} \cdot \boldsymbol{n}^+$. Another common way of writing the upwind flux is

$$\widehat{\boldsymbol{c}u} = \boldsymbol{c}\,\{\{u\}\} + \frac{1}{2}|\boldsymbol{c} \cdot \boldsymbol{n}|[\![u\boldsymbol{n}]\!] \tag{7.7}$$

which is actually correct regardless of whether $\boldsymbol{n}^+$ or $\boldsymbol{n}^-$ is chosen for $\boldsymbol{n}$. To see this, we make use of the vector triple product identity $\boldsymbol{a} \times \boldsymbol{b} \times \boldsymbol{c} = \boldsymbol{b}(\boldsymbol{a} \cdot \boldsymbol{c}) - \boldsymbol{c}(\boldsymbol{a} \cdot \boldsymbol{b})$, which implies

$$\begin{aligned} (\boldsymbol{c} \cdot \boldsymbol{n})\,\boldsymbol{n}^{\pm} &= \boldsymbol{n} \times \boldsymbol{n}^{\pm} \times \boldsymbol{c} + \boldsymbol{c}\left(\boldsymbol{n} \cdot \boldsymbol{n}^{\pm}\right) \\ &= \boldsymbol{c}\left(\boldsymbol{n} \cdot \boldsymbol{n}^{\pm}\right). \end{aligned}$$

Expanding equation (7.7),

$$\widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- + \frac{1}{2}|\boldsymbol{c} \cdot \boldsymbol{n}|u^+\boldsymbol{n}^+ + \frac{1}{2}|\boldsymbol{c} \cdot \boldsymbol{n}|u^-\boldsymbol{n}^-,$$

yielding the two cases

$$\begin{aligned}
\boldsymbol{c}\cdot\boldsymbol{n}\geq 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- + \frac{1}{2}(\boldsymbol{c}\cdot\boldsymbol{n})\boldsymbol{n}^+u^+ + \frac{1}{2}(\boldsymbol{c}\cdot\boldsymbol{n})\boldsymbol{n}^-u^-,\\
\boldsymbol{c}\cdot\boldsymbol{n}< 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- - \frac{1}{2}(\boldsymbol{c}\cdot\boldsymbol{n})\boldsymbol{n}^+u^+ - \frac{1}{2}(\boldsymbol{c}\cdot\boldsymbol{n})\boldsymbol{n}^-u^-,
\end{aligned}$$

If $\boldsymbol{n} = \boldsymbol{n}^+$, then

$$\begin{aligned}
\boldsymbol{c}\cdot\boldsymbol{n}^+\geq 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- + \frac{1}{2}\boldsymbol{c}u^+ - \frac{1}{2}\boldsymbol{c}u^- = \boldsymbol{c}u^+\\
\boldsymbol{c}\cdot\boldsymbol{n}^+< 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- - \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- = \boldsymbol{c}u^-
\end{aligned}$$

on the other hand, if $\boldsymbol{n} = \boldsymbol{n}^-$, then

$$\begin{aligned}
\boldsymbol{c}\cdot\boldsymbol{n}^-\geq 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- - \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- = \boldsymbol{c}u^-\\
\boldsymbol{c}\cdot\boldsymbol{n}^-< 0: \qquad & \widehat{\boldsymbol{c}u} = \frac{1}{2}\boldsymbol{c}u^+ + \frac{1}{2}\boldsymbol{c}u^- + \frac{1}{2}\boldsymbol{c}u^+ - \frac{1}{2}\boldsymbol{c}u^- = \boldsymbol{c}u^+
\end{aligned}$$

and we have that the flux is indeed upwinded.

7.1.0.3 Vector calculus

Construction of divergence-free velocity fields. It can be of numerical interest to test the discretization of a divergence-free velocity field. The vector calculus identity

$$\nabla\cdot(F\times G) = G\cdot(\nabla\times F) - F\cdot(\nabla\times G) \tag{7.8}$$

implies that if we choose two continuous, twice-differentiable vector fields f and g, and choose $\boldsymbol{C} = \nabla f\times\nabla g$, we have that the divergence $\nabla\cdot\boldsymbol{C} = 0$, since $\nabla\times(\nabla\phi) = 0$. This approach is useful for generating three-dimensional divergence-free vector fields, but to generate two-dimensional divergence-free vector fields, we must be careful to avoid a z-dependence in C_z; if we have that the dependence $C_z(x,y)$, then the partial derivative $\partial/\partial z(C_z)$ vanishes and

$$\nabla\cdot\boldsymbol{C} = \frac{\partial C_x}{\partial x} + \frac{\partial C_y}{\partial y} = 0.$$

To avoid a trivial vector field in the xy plane, it suffices to choose the functions f and g of the form $z + f(x,y)$.

General Relations.

$$\int_\Omega \boldsymbol{c}v\cdot\nabla v\,d\Omega = \frac{1}{2}\int_\Omega \boldsymbol{c}\cdot\nabla v^2\,d\Omega = -\frac{1}{2}\int_\Omega v^2\nabla\cdot\boldsymbol{c}\,d\Omega + \frac{1}{2}\int_{\partial\Omega} v^2\boldsymbol{c}\cdot\boldsymbol{n}\,d\partial\Omega \tag{7.9}$$

7.2 Consistency and stability of DG-FEM for second-order problems

We use the so-called energy method to derive a set of numerical fluxes such that the numerical solution is stable [39]. Our point of departure is the element-local problem

$$\begin{aligned} (\boldsymbol{q}_h,\, \boldsymbol{\mathsf{v}})_K + (u_h,\, \nabla\cdot\boldsymbol{\mathsf{v}})_K - \langle \widehat{u}_h,\, \boldsymbol{\mathsf{v}}\cdot\boldsymbol{n}\rangle_{\partial K} &= 0 \\ (\boldsymbol{q}_h,\, \nabla\mathsf{w})_K - \langle \widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n},\, \mathsf{w}\rangle_{\partial K} &= (f,\, \mathsf{w})_K\,. \end{aligned}$$

Taking $\boldsymbol{\mathsf{v}} = \boldsymbol{q}_h$ and $\mathsf{w} = u_h$ and summing over all mesh elements for each equation, we find

$$\begin{aligned} \int_\Omega |\boldsymbol{q}_h|^2\, d\Omega - \sum_{K\in\mathcal{T}_h}\left(-\int_K u_h(\nabla\cdot\boldsymbol{q}_h)\, dK + \int_{\partial K}\widehat{u}_h(\boldsymbol{q}_h\cdot\boldsymbol{n})\, d\partial K\right) &= 0 \\ \sum_{K\in\mathcal{T}_h}\left(\int_K \boldsymbol{q}_h\cdot\nabla u_h\, dK - \int_{\partial K} u_h(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\, d\partial K\right) &= \int_\Omega f u_h\, d\Omega, \end{aligned}$$

and summing the two yields

$$\int_\Omega |\boldsymbol{q}_h|^2\, d\Omega + \Theta_h = \int_\Omega f u_h\, d\Omega,$$

with

$$\Theta_h = -\sum_{K\in\mathcal{T}_h}\left(-\int_K u_h(\nabla\cdot\boldsymbol{q}_h)\, dK - \int_K \boldsymbol{q}_h\cdot\nabla u_h\, dK + \int_{\partial K}\widehat{u}_h(\boldsymbol{q}_h\cdot\boldsymbol{n})\, d\partial K + \int_{\partial K} u_h(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\, d\partial K\right).$$

In order for the numerical solutions $(u_h,\, \boldsymbol{q}_h)$ to be bounded and hence stable, we must choose the numerical fluxes such that $\Theta_h \geq 0$. Doing so ensures that with right-hand side data $f = 0$, the only possible solution is $(u_h,\, \boldsymbol{q}_h) = (0,\, \boldsymbol{0})$; in which case, we have that there exists a unique solution for every nontrivial f and the DG method is well-posed. Using first the product rule $\nabla\cdot(u_h\boldsymbol{q}_h) = u_h(\nabla\cdot\boldsymbol{q}_h) + \boldsymbol{q}_h\cdot\nabla u_h$, followed by the divergence theorem $\int_K \nabla\cdot(u_h\boldsymbol{q}_h)\, dK = \int_{\partial K} u_h\boldsymbol{q}_h\cdot\boldsymbol{n}\, d\partial K$,k we can simplify our expression:

$$\begin{aligned} \Theta_h &= \sum_{K\in\mathcal{T}_h}\int_{\partial K}\left(u_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - \widehat{u}_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - u_h(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\right)\, d\partial K \\ &= \sum_{e\in\varepsilon}\int_e \llbracket (u_h\boldsymbol{q}_h - \widehat{u}_h\boldsymbol{q}_h - u_h\widehat{q}_h)\,\boldsymbol{n}\rrbracket\, de \end{aligned}$$

Where we have reorganized the sum over each mesh edge $e \in \varepsilon$ rather than over the element boundaries ∂K. Applying the respective definitions of the jump operators on the interior edges ε° and boundary edges ε^∂, and using the fact that the numerical traces $\widehat{u}_h$ and $\widehat{\boldsymbol{q}}_h$ are single-valued, we have

$$\begin{aligned} \Theta_h &= \sum_{e\in\varepsilon^\circ}\int_e \left(\llbracket u_h(\boldsymbol{q}_h\cdot\boldsymbol{n})\rrbracket - \widehat{u}_h\llbracket\boldsymbol{q}_h\cdot\boldsymbol{n}\rrbracket - \llbracket u_h\boldsymbol{n}\rrbracket(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\right)\, de \\ &\quad + \sum_{e\in\varepsilon^\partial}\int_e \left(u_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - \widehat{u}_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - u_h(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\right)\, de \\ &= \sum_{e\in\varepsilon^\circ}\int_e \left(\{\{u_h\}\}\,\llbracket\boldsymbol{q}_h\cdot\boldsymbol{n}\rrbracket + \llbracket u_h\boldsymbol{n}\rrbracket\,\{\{\boldsymbol{q}_h\cdot\boldsymbol{n}\}\} - \widehat{u}_h\llbracket\boldsymbol{q}_h\cdot\boldsymbol{n}\rrbracket - \llbracket u_h\boldsymbol{n}\rrbracket(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\right)\, de \\ &\quad + \sum_{e\in\varepsilon^\partial}\int_e \left(u_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - \widehat{u}_h(\boldsymbol{q}_h\cdot\boldsymbol{n}) - u_h(\widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n})\right)\, de \end{aligned}$$

where we have applied the identity in equation (7.3) in the second step. Regrouping yields our final expression for Θ_h,

$$\begin{aligned}\Theta_h = &\sum_{e\in\varepsilon^\circ}\int_e \Big[\left(\{\{u_h\}\} - \widehat{u}_h\right)[\![\boldsymbol{q}_h\cdot\boldsymbol{n}]\!] + [\![u_h\boldsymbol{n}]\!]\left(\{\{\boldsymbol{q}_h\cdot\boldsymbol{n}\}\} - \widehat{\boldsymbol{q}}_h\cdot\boldsymbol{n}\right)\Big]\,de \\ &+ \sum_{e\in\varepsilon^\partial}\int_e \Big[u_h\left(\boldsymbol{q}_h - \widehat{\boldsymbol{q}}_h\right)\cdot\boldsymbol{n} - \widehat{u}_h\left(\boldsymbol{q}_h\cdot\boldsymbol{n}\right)\Big]\,de.\end{aligned} \tag{7.10}$$

Now it is easily verified that the choices

$$\begin{aligned}\widehat{\boldsymbol{q}}_h &= \{\{\boldsymbol{q}_h\}\} - c_{11}[\![u_h\boldsymbol{n}]\!] + \boldsymbol{c}_{12}[\![\boldsymbol{q}_h\cdot\boldsymbol{n}]\!],\\ \widehat{u}_h &= \{\{u_h\}\} - \boldsymbol{c}_{12}\cdot[\![u_h\boldsymbol{n}]\!] - c_{22}[\![\boldsymbol{q}_h\cdot\boldsymbol{n}]\!],\end{aligned}$$

on the domain interior edges ε°, and

$$\begin{aligned}\widehat{\boldsymbol{q}}_h &= \boldsymbol{q}_h - c_{11}u_h\boldsymbol{n},\\ \widehat{u}_h &= 0,\end{aligned}$$

on the boundary edges ε^∂, with $c_{11}, c_{22} \geq 0$ on all edges ε, when substituted into equation (7.10), yields

$$\Theta_h = \sum_{e\in\varepsilon^\circ}\int_e \left(c_{22}[\![\boldsymbol{q}_h\cdot\boldsymbol{n}]\!]^2 + c_{11}[\![u_h\boldsymbol{n}]\!]^2\right)de + \sum_{e\in\varepsilon^\partial}\int_e c_{11}u_h^2\,de \;\geq 0,$$

and we have that the DG methods are well-posed.

7.3 Proof: well-posedness under different HDG fluxes

A well-posed choice of HDG flux depends on how we define $\boldsymbol{q}$ when re-writing the elliptic PDE as a first order system. If we choose to define $\boldsymbol{q} \equiv -\kappa \nabla u$, we must choose $\hat{\boldsymbol{q}} = \boldsymbol{q} + \tau(u - \hat{u})\boldsymbol{n}$ for well-posedness [191], if $\tau > 0$. On the other hand, if we define $\boldsymbol{q} \equiv \kappa \nabla u$, we must choose $\hat{\boldsymbol{q}} = \boldsymbol{q} - \tau(u - \hat{u})\boldsymbol{n}$. We'll prove the latter, because the same steps can be used to prove the former.

The first-order system

$$\begin{aligned} \boldsymbol{q} - \kappa \nabla u &= 0, \\ -\nabla \cdot \boldsymbol{q} &= f, \\ u &= g_D, \\ \boldsymbol{q} \cdot \boldsymbol{n} &= g_N, \end{aligned} \tag{7.11}$$

Taking the usual approach of multiplying by a suitable test function, integrating by parts, and adding the global condition $\langle \hat{\boldsymbol{q}} \cdot \boldsymbol{n},\, \mu \rangle_{\partial \mathcal{T}_h} = 0$ admits the following HDG weak form: we seek $(\boldsymbol{q}_h,\, u_h,\, \hat{u}_h) \in \boldsymbol{V}_h^k \times W_h^k \times M_h^k$ such that

$$\begin{aligned} (\boldsymbol{v},\, \kappa^{-1}\boldsymbol{q}_h)_{\mathcal{T}_h} + (\nabla \cdot \boldsymbol{v},\, u_h) - \langle \boldsymbol{v} \cdot \boldsymbol{n},\, \hat{u}_h \rangle_{\partial \mathcal{T}_h} &= 0, \\ (\nabla w,\, \boldsymbol{q}_h)_{\mathcal{T}_h} - \langle w,\, \hat{\boldsymbol{q}}_h \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h} &= (w, f)_{\mathcal{T}_h}, \\ \langle \mu,\, \hat{\boldsymbol{q}}_h \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h \setminus \Gamma_D} + \langle \mu,\, \hat{u}_h - g_D \rangle_{\Gamma_D} &= \langle \mu,\, g_N \rangle, \end{aligned} \tag{7.12}$$

for all $(\boldsymbol{v},\, w,\, \mu) \in \boldsymbol{V}_h^k \times W_h^k \times M_h^k$. For well-posedness, we need to show that $(\boldsymbol{q}_h, u_h, \hat{u}_h) = (\boldsymbol{0}, 0, 0)$ if $f = g_D = g_N = 0$. Choosing $(\boldsymbol{v}, w, \mu) = (\boldsymbol{q}_h, u_h, \hat{u}_h)$ and $\hat{\boldsymbol{q}} = \boldsymbol{q}_h - \tau(u_h - \hat{u}_h)$, we write the weak form in a single equation as

$$\begin{aligned} (\boldsymbol{q}_h,\, \kappa^{-1}\boldsymbol{q}_h)_{\mathcal{T}_h} + (\nabla \cdot \boldsymbol{q}_h,\, u_h)_{\mathcal{T}_h} - \langle \boldsymbol{q}_h \cdot \boldsymbol{n},\, \hat{u}_h \rangle_{\partial \mathcal{T}_h} \\ + (\nabla u_h,\, \boldsymbol{q}_h)_{\mathcal{T}_h} - \langle u_h,\, \boldsymbol{q}_h \cdot \boldsymbol{n} - \tau(u_h - \hat{u}_h) \rangle_{\partial \mathcal{T}_h} \\ \langle \boldsymbol{q}_h \cdot \boldsymbol{n} - \tau(u_h - \hat{u}_h),\, \hat{u}_h \rangle_{\partial \mathcal{T}_h} = 0 \end{aligned}$$

By integrating $(\nabla u_h,\, \boldsymbol{q}_h)\mathcal{T}_h = -(\nabla \cdot \boldsymbol{q}_h,\, u_h)_{\mathcal{T}_h} + \langle u_h,\, \boldsymbol{q}_h \cdot \boldsymbol{n} \rangle_{\partial \mathcal{T}_h}$ by parts and cancelling terms, we have

$$(\boldsymbol{q}_h,\, \kappa^{-1}\boldsymbol{q}_h)_{\mathcal{T}_h} + \langle \tau(u_h - \hat{u}_h),\, (u_h - \hat{u}_h) \rangle_{\partial \mathcal{T}_h} = 0$$

from where we see that as long as $\tau > 0$, the only solution is that $(\boldsymbol{q}_h, u_h, \hat{u}_h) = (\boldsymbol{0}, 0, 0)$, and we have shown well-posedness. From this reasoning, it should be clear that we need to modify the definition of $\hat{\boldsymbol{q}}$ in order to arrive that the same result for the weak problem resulting from the alternative choice $\boldsymbol{q} \equiv \kappa \nabla u$.

7.4 A unifying abstraction for HDG assembly

$$\begin{pmatrix} \mathcal{A}_{\text{loc}} & B \\ C & D \end{pmatrix} \begin{pmatrix} U \\ \hat{U} \end{pmatrix} = \begin{pmatrix} R \\ G \end{pmatrix} \tag{7.13}$$

where $\mathcal{A}_{\text{loc}}$ represents the element local system with respect to U, which contains the solution u_h and its gradient $\boldsymbol{q}_h$. The first equation corresponds to the test functions $\boldsymbol{v}$, w, and the second corresponds to the test functions μ. The operators B and D represent the local-to-face and face-to-local couplings, respectively.

The process of static condensation, then, refers to eliminating the system in terms of one for $\hat{U}$ alone, by means of the Schur complement. Namely,

$$U = \mathcal{A}_{\text{loc}}^{-1} \left(R - B\hat{U} \right), \tag{7.14}$$

$$\underbrace{\left(D - C\mathcal{A}_{\text{loc}}^{-1} B \right)}_{\mathbb{K}} \hat{U} = \underbrace{G - C\mathcal{A}_{\text{loc}}^{-1} R}_{\mathbb{F}}, \tag{7.15}$$

where equation (7.15) describes the statically-condensed, globally-coupled linear system $\mathbb{K}\hat{U} = \mathbb{F}$, and equation (7.14) describes the elemental reconstruction of the solution U from the skeleton unknowns $\hat{U}$.

From a software perspective, we only need methods that assemble these operators on each element. The rest of the HDG process can be shared between solvers.

Considering the model problem in Nguyen 2009, we can write the weak form suggestively as

$$\left(\begin{array}{c|c} \left(\boldsymbol{v},\, \kappa^{-1}\boldsymbol{q}_h\right)_{\mathcal{T}_h} - (\nabla\cdot\boldsymbol{v},\, u_h)_{\mathcal{T}_h} & + \langle \boldsymbol{v}\cdot\boldsymbol{n},\, \hat{u}_h\rangle_{\partial\mathcal{T}_h} \\ -(w, \boldsymbol{c}u_h)_{\mathcal{T}_h} + (w,\, \nabla\cdot\boldsymbol{q}_h)_{\mathcal{T}_h} + \langle w,\, \tau u_h\rangle_{\partial\mathcal{T}_h} & \langle w,\, (\boldsymbol{c}\cdot\boldsymbol{n} - \tau)\hat{u}_h\rangle_{\partial\mathcal{T}_h} \\ \hline \langle \mu,\, \boldsymbol{q}_h\cdot\boldsymbol{n}\rangle_{\partial\mathcal{T}_h} - \langle \mu,\, \tau u_h\rangle_{\partial\mathcal{T}_h} & \langle \mu,\, (\boldsymbol{c}\cdot\boldsymbol{n}\hat{u}_h)\rangle_{\partial\mathcal{T}_h} + \langle \mu,\, -\tau\hat{u}_h\rangle_{\partial\mathcal{T}_h} \end{array} \right) \begin{pmatrix} Q \\ U \\ \hat{U} \end{pmatrix} = \begin{pmatrix} 0 \\ (w,\, f)_{\mathcal{T}_h} \\ \langle \mu,\, g_N\rangle_{\partial\mathcal{T}_h} \end{pmatrix} \tag{7.16}$$

To delineate the operators A, B, C, and D. However, in light of the fact that the weak form is ultimately a scalar equation, it is apparent that the division into three separate equations, one for w, $\boldsymbol{v}$, and μ, is artificial, and we can write all three together as a single equation. Expanding the solutions $u_h = \sum_j u_j \phi_{u,j}$, $\boldsymbol{q}_h = \sum_j q_j \boldsymbol{\phi}_{q,j}$ and $\hat{u}_h = \sum_j \lambda_i \phi_{\mu,j}$, where the index j indexes all system degrees of freedom, we have the monolithic linear system $\mathbb{A}$ (before hybridization):

$$\begin{aligned} \mathbb{A}_{ij} = & \left(\boldsymbol{\phi}_{q,i},\, \kappa^{-1}\boldsymbol{\phi}_{q,j}\right) - (\nabla\cdot\boldsymbol{\phi}_{q,i},\, \phi_{u,j}) && + \langle \boldsymbol{\phi}_{q,i}\cdot\boldsymbol{n},\, \phi_{\mu,j}\rangle \\ & - (\nabla\phi_{u,i}\cdot\boldsymbol{c},\, \phi_{u,j}) + (\phi_{u,i},\, \nabla\cdot\boldsymbol{\phi}_{q,j}) + (\phi_{u,i},\, \tau\phi_{u,j}) && + \langle \phi_{u,i},\, (\boldsymbol{c}\cdot\boldsymbol{n} - \tau)\phi_{\mu,j}\rangle \\ & + \langle \phi_{\mu,i},\, \boldsymbol{\phi}_{q,j}\cdot\boldsymbol{n}\rangle - \langle \phi_{\mu,i},\, \tau\phi_{u,j}\rangle && + \langle \phi_{\mu,i}, (\boldsymbol{c}\cdot\boldsymbol{n} - \tau),\, \phi_{\mu,j}\rangle \\ = & (\phi_{u,j}, f) && + \langle \phi_{\mu,j},\, g_N\rangle, \end{aligned} \tag{7.17}$$

where the vast majority of these terms will be zero in the matrix; the linear system exhibits the usual block-diagonal pattern over each element in the upper right $\mathcal{A}_{\text{loc}}$ block.

However, in [191] the flux unknown $\boldsymbol{q}_h$ is defined by $\boldsymbol{q} \equiv -\kappa\nabla u$, which mimics a pure diffusive flux in the sense of Fick's Law, but inconvenient for when we just care about the gradient ∇u. Making the alternate definition $\boldsymbol{q} \equiv \nabla u$, we arrive instead at the first-order system

$$\begin{aligned} \boldsymbol{q} - \nabla u &= 0, \\ \frac{\partial u}{\partial t} + \nabla\cdot(\kappa\boldsymbol{q}) &= f. \end{aligned} \tag{7.18}$$

Unfortunately, from both a software and a formulation perspective, this choice of $\boldsymbol{q}$ is not desirable. First, in a resultant weak form, evaluating a term of the form $(\boldsymbol{v},\, \nabla\cdot(\kappa\boldsymbol{q}))_K$ can be inconvenient, because computing the divergence of the basis function $\boldsymbol{\phi}_{q,j}$ is readily available, but not the product $\kappa\boldsymbol{\phi}_{q,j}$; further, in the case of a spatially variable $\kappa(\boldsymbol{x})$, the product rule would have to be applied to the divergence term, or an additional integration by parts would be required, resulting in a term like $(\nabla\boldsymbol{v}, \kappa\boldsymbol{q})_K$ plus additional edge terms.

7.5 Time integration methods

In the body of the thesis, we did not give any details as to the time-marching schemes used. In this section, we will describe the numerical methods used for evolving the systems of ordinary differential equations (ODEs) that arise in the finite element discretizations considered in this thesis. A solvable ODE with a particular solution is often referred to as an initial value problem, which consist of an ordinary differential equation (ODE) and an initial condition (IC) written in the form

$$\begin{aligned} y'(t) &= f(y(t)), \\ y(t_0) &= y_0, \end{aligned} \tag{7.19}$$

where $f : \mathbf{R} \to \mathbf{R}$.

The generalization of the initial value problem in equation (7.19) to a system of ODEs can be written as

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}(t), t), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{7.20}$$

where $\boldsymbol{y}(t) \in \mathbf{R}^N$ is the state vector and $\boldsymbol{r}(\boldsymbol{y}) \in \mathbf{R}^N$ is a vector-valued function of y. In principle, the general problem in equation (7.20) is the only case necessary to consider. However, because there are practical concerns with associated with systems of ODEs that are irrelevant to scalar ODEs, we use bolded vector notation to be explicit about the case to which we are referring.

An example of a concern specific to systems of ODEs occurs in the semi-discretization of PDEs using finite element methods. In this case, semi-discretization often results in an ODE system of the following form:

$$M\frac{d\boldsymbol{y}}{dt} = \boldsymbol{r}(\boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{7.21}$$

where the matrix $M \in \mathbf{R}^{N\times N}$ is invertible, and the right-hand side $\boldsymbol{r}(\boldsymbol{y})$ has the interpretation of a residual. In classical CG-FEM schemes, this matrix may be non-trivial to invert and much more computationally efficient to apply. In DG-FEM schemes, the matrix M is often trivially invertible on each element, and can efficiently be both written and implemented in the form of equation (7.20) where we take

$$\boldsymbol{f}(\boldsymbol{y}(t), t) = M^{-1}\boldsymbol{r}(\boldsymbol{y}). \tag{7.22}$$

It will usually be simpler, algebraically and implementation-wise, to write and solve problems of the form in equation (7.20) rather than in equation (7.21).

7.5.0.1 Runge-Kutta Methods

The material in this section comes primarily from [36, 102, 103]. We will consider the Runge-Kutta (RK) approach for advancing a state variable y_n at t to y_{n+1} at $t + \Delta t$. In the course of each time-step, we consider s stages, and the relations

$$\begin{aligned} y_{n+1} &= y_n + \Delta t \sum_{i=1}^{s} b_i k_i \\ k_i &= f\left(y_n + \Delta t \sum_{j=1}^{s} a_{ij} k_j, \quad t_n + c_i \Delta t\right) \end{aligned} \tag{7.23}$$

determine the RK scheme. We will write y_n as y_0 and y_{n+1} as y_1 without loss of generality for convenience. RK methods are specified by the coefficients a_{ij}, b_i, and c_i. A common and convenient way of representing these coefficients is with the so-called “Butcher table”,

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad = \quad \begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & bs \end{array}, \tag{7.24}$$

in which the row vector contains the quadrature weights that specify how the state at the next time-step y_1 depends on the derivatives computed at the various stages. The matrix A indicates dependence of the stage values on the derivatives found at other stages. The vector c contains the positions within the time-step of the stage values.

Example 1. Forward Euler

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \tag{7.25}$$

$$\begin{aligned} y_1 &= y_0 + \Delta t \underbrace{(1)}_{b_1} k_1 \\ k_1 &= f(t_0 + \underbrace{(0)}_{c_1} \Delta t,\, y_0 + \Delta t \underbrace{(0)}_{a_{11}} k_1) \end{aligned}$$

yielding the familiar

$$y_1 = y_0 + \Delta t f(y_0).$$

Example 2. Midpoint rule method, RK22

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} \tag{7.26}$$

Gives

$$\begin{aligned} y_1 &= y_0 + \Delta t(\underbrace{(0)}_{b_1} k_1 + \underbrace{(1)}_{b_2} k_2) \\ k_1 &= f(t_0 + \underbrace{(0)}_{c_1} \Delta t,\, y_0 + \Delta t(\underbrace{(0)}_{a_{11}} k_1 + \underbrace{(0)}_{a_{12}} k_2) = f(t_0, y_0) \\ k_2 &= f\left(t_0 + \frac{1}{2}\Delta t,\, y_0 + \Delta t\left(\frac{1}{2}k_1 + 0k_2\right)\right) = f\left(t_0 + \frac{1}{2}\Delta t,\, y_0 + \Delta t\frac{1}{2}f(t_0,\, y_0)\right) \end{aligned}$$

yielding

$$y_1 = y_0 + \Delta t f\left(t_0 + \frac{1}{2}\Delta t,\, y_0 + \Delta t\frac{1}{2}f(t_0,\, y_0)\right).$$

Example 3. Backward Euler

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \tag{7.27}$$

Gives

$$\begin{aligned} y_1 &= y_0 + \Delta t k_1 \\ k_1 &= f(t_0 + \Delta t, y_0 + \Delta t k_1), \end{aligned}$$

yielding a coupled system. However, we can simplify the relations algebraically, since the first equation implies

$$k_1 = \frac{y_1 - y_0}{\Delta t},$$

which can be substituted into the second equation to yield

$$y_1 = y_0 + \Delta t f(t_1, y_1)$$

which must be solved implicitly for y_1.

In Example 1 and Example 2, the starting state could be used to generate k_1 because the first row $a_{1j} = 0$. Then k_2 could be computed directly from k_1, since the diagonal of A in each case was zero. These properties made the schemes explicit.

The A matrix of the Butcher table determines the implicit or explicit nature of the RK method. Whenever the matrix A is lower triangular (zeros on and above the diagonal) the current stage value only depends on past stage values, and the scheme will be explicit. However, in the general case of a completely non-zero matrix A, the relations in equation (7.23) are coupled and the entire algebraic system must be solved simultaneously. If A contains entries on or above the diagonal, the scheme will be implicit. Common classifications based on the matrix A of the Butcher table are:

- Implicit Runge-Kutta (IRK) methods (in the general case) have a full A matrix.
- Diagonally implicit Runge-Kutta (DIRK) methods have $a_{ij} = 0$ for $i < j$. If all diagonal elements are equal ($a_{ii} = \gamma$, $i = 1, \ldots, s$), the scheme is called a *singly diagonal* implicit Runge-Kutta (SDIRK) method.
- Explicit Runge-Kutta (ERK) methods have $a_{ij} = 0$ for $i \leq j$.
- Low-storage explicit Runge-Kutta (LSERK) methods are explicit methods where all stage values k_i depend only on k_{i-1}; when this is the case, all stage values k_i not need to be stored and can instead be implemented as a running sum.

depicted in Figure 7-1.

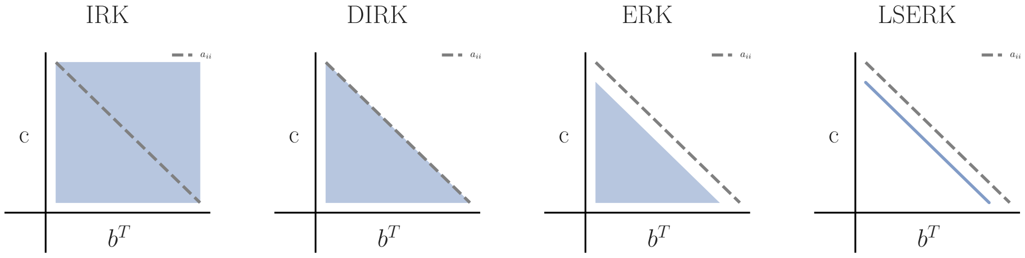


Figure 7-1: Nonzero entries in the matrix A of the Butcher table for different RK methods.

Example 4. Classical fourth-order ERK (RK4)

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{7.28}$$

Example 5. Second-order DIRK(2,2) [196].

$$\begin{array}{c|cc} \alpha & \alpha & 0 \\ 1 & 1-\alpha & \alpha \\ \hline & 1-\alpha & \alpha \end{array} \tag{7.29}$$

where $\alpha = 1 + \sqrt{2}/2$. The scheme is evaluated as

$$k_1 = f\left(t_n + \alpha\Delta t,\, y_n + \Delta t\alpha k_1\right), \tag{7.30}$$

$$k_2 = f\left(t_n + \Delta t,\, y_n + \Delta t\left((1-\alpha)k_1 + \alpha k_2\right)\right), \tag{7.31}$$

$$y_{n+1} = y_n + \Delta t\left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right), \tag{7.32}$$

where we solve equation (7.30) for k_1, use its value to solve equation (7.31) for k_2, then evaluate y_1 with equation (7.32). The lower-triangular nature of A for DIRK schemes means we can solve for each k_i at each stage, rather than all of them at once.

Example 6. Third-order SDIRK(3,3) method [102].

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \gamma = \frac{3 \pm \sqrt{3}}{6} \tag{7.33}$$

Example 7. Third-order IRK (Radau IA)

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \tag{7.34}$$

The scheme is evaluated as

$$\begin{aligned} k_1 &= f\left(t_n,\, y_n + \Delta t\left(\frac{1}{4}k_1 - \frac{1}{4}k_2\right)\right) \\ k_2 &= f\left(t_n + \frac{2}{3}\Delta t,\, y_n + \Delta t\left(\frac{1}{4}k_1 + \frac{5}{12}k_2\right)\right) \\ y_{n+1} &= y_n + \Delta t\left(\frac{1}{4}k_1 + \frac{3}{4}k_2\right) \end{aligned} \tag{7.35}$$

Here, the first two equations must be solved simultaneously for k_1 and k_2, a potentially expensive operation. Only once the coupled system has been solved can y_{n+1} be computed.

7.5.0.2 RK Methods for systems of ODEs

Then the RK method for the system ODEs in equation (7.20) with state vector y can be written as

$$M\boldsymbol{k}_i = \boldsymbol{r}\left(t_n + c_i\Delta t,\, \boldsymbol{y}_n + \Delta t\sum_{j=1}^{s} a_{ij}\boldsymbol{k}_j\right) \tag{7.36}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta t\sum_{i=1}^{s} b_i\boldsymbol{k}_i \tag{7.37}$$

Definition from [206].

Discretization of a linear PDE by method of lines will often result in a matrix equation which contains both a time derivative term and a matrix operator representing the differential operators. For example,

$$M\frac{\partial\boldsymbol{u}}{\partial t} + A\boldsymbol{u} = \boldsymbol{f}, \tag{7.38}$$

where equation (7.37) can be applied by noting that $\boldsymbol{r}(\boldsymbol{y}) = \boldsymbol{f} - A\boldsymbol{u}$.

For DG-FEM methods in particular, it is often the case that M represents an elemental mass matrix, which can be inverted on each element and applied to the whole equation algebraically, leading to a semidiscrete problem of the form

$$\frac{d\boldsymbol{u}_h}{dt} = \mathcal{L}_h(t, \boldsymbol{u}_h), \tag{7.39}$$

which is the standard way to handle time-marching for DG-FEM. Further details can be found in [44, 110].

Example 8. Fourth-order ERK method for the semidiscrete problem equation (7.39). We use the Butcher table in Example 4 to yield the four-stage explicit method

$$\begin{aligned}
\boldsymbol{k}_1 &= \mathcal{L}_h(t_n,\, \boldsymbol{u}_h^n),\\
\boldsymbol{k}_2 &= \mathcal{L}_h\left(t_n + \frac{1}{2}\Delta t,\, \boldsymbol{u}_h^n + \frac{1}{2}\Delta t \boldsymbol{k}_1\right),\\
\boldsymbol{k}_3 &= \mathcal{L}_h\left(t_n + \frac{1}{2}\Delta t,\, \boldsymbol{u}_h^n + \frac{1}{2}\Delta t \boldsymbol{k}_2\right),\\
\boldsymbol{k}_4 &= \mathcal{L}_h\left(t_n + \Delta t,\, \boldsymbol{u}_h^n + \Delta t \boldsymbol{k}_3\right),\\
\boldsymbol{u}_h^{n+1} &= \boldsymbol{u}_h^n + \frac{1}{6}\Delta t\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right).
\end{aligned} \tag{7.40}$$

By using different quadrature points in time and keeping a running sum, we can improve the ERK method in Example 8.

Example 9. Fourth-order five-stage low-storage ERK method (LSRK4).

$$\begin{aligned}
\boldsymbol{p}^0 &= \boldsymbol{u}_h^n,\\
&\ i = 1, \ldots, 5:\\
&\ \begin{cases} \boldsymbol{k}_i = a_i \boldsymbol{k}_{i-1} + \Delta t \mathcal{L}_h\left(t_n + c_i \Delta t,\, \boldsymbol{p}_{i-1}\right),\\ \boldsymbol{p}_i = \boldsymbol{p}_{i-1} + b_i \boldsymbol{k}_i, \end{cases}\\
\boldsymbol{u}_h^{n+1} &= \boldsymbol{p}_5,
\end{aligned} \tag{7.41}$$

where the coefficients a_i, b_i, and c_i are given in [110], p.64.

Example 10. DIRK(2,2) for the semidiscrete problem

$$\frac{d\boldsymbol{u}_h}{dt} = A\boldsymbol{u}_h \tag{7.42}$$

$$\begin{aligned}
\boldsymbol{u}^{(1)} &= A\left(\boldsymbol{u}_h^n + \Delta t \alpha \boldsymbol{u}^{(1)}\right)\\
\boldsymbol{u}^{(1)} &= A\boldsymbol{u}_h^n + \Delta t \alpha A \boldsymbol{u}^{(1)}\\
\boldsymbol{u}^{(1)} - \Delta t \alpha A \boldsymbol{u}^{(1)} &= A\boldsymbol{u}_h^n\\
\frac{\boldsymbol{u}^{(1)}}{\alpha \Delta t} - A\boldsymbol{u}^{(1)} &= \frac{A\boldsymbol{u}_h^n}{\alpha \Delta t}\\
\boldsymbol{k}_1 &= \mathcal{L}_h\left(t_n + \alpha \Delta t,\, \boldsymbol{u}_h^n + \Delta t \alpha \boldsymbol{k}_1\right)
\end{aligned} \tag{7.43}$$

7.5.0.3 IMEX-RK Methods

7.5.0.4 General IMEX methods

This exposition is cursory; for a more in-depth discussion, we refer the reader to [14]. While section 7.5.0.1 considered the ordinary differential equation in equation (7.19), the material in this section is motivated by the challenges of time-marching for PDEs; therefore we consider the nonlinear convection diffusion equation

$$\frac{\partial u}{\partial t} = u\frac{\partial u}{\partial x} + \nu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) \tag{7.44}$$

with $\nu > 0$ and appropriate initial and boundary conditions. The computational issue with this PDE is that the convective (hyperbolic) term is nonlinear but generally not stiff, whereas the diffusion term is linear but stiff. An explicit time-marching scheme would have severe time-step restrictions on Δt due to CFL requirements imposed by the linear diffusion term, and an implicit discretization would require a nonlinear solver due to the nonlinearity of the convective term. More generally, for PDEs of the form

$$u_t = f(u) + g(u) \tag{7.45}$$

where f corresponds to a non-stiff term and g corresponds to a stiff term, IMEX schemes apply an implicit discretization for g and an explicit one for f. An IMEX formulation for equation (7.45) is as follows

$$\begin{aligned}
&\widehat{\boldsymbol{k}}_1 = f(\boldsymbol{u}_h^n) \\
&\quad i = 1, \dots, s: \\
&\quad\begin{cases} \text{solve } \boldsymbol{k}_i = g(\boldsymbol{u}_h^i) \\ \qquad \text{where } \boldsymbol{u}_h^i = \boldsymbol{u}_h^n + \Delta t\sum_{j=1}^{i} a_{ij}\boldsymbol{k}_j + \Delta t\sum_{j=1}^{i}\widehat{a}_{i+1,j}\,\widehat{\boldsymbol{k}}_j \\ \text{evaluate } \widehat{\boldsymbol{k}}_{i+1} = f(\boldsymbol{u}_h^i) \end{cases} \\
&\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_h^n + \Delta t\sum_{j=1}^{s} b_j\boldsymbol{k}_j + \Delta t\sum_{j=1}^{s+1}\widehat{b}_j\widehat{\boldsymbol{k}}_j.
\end{aligned} \tag{7.46}$$

The last equation in equation (7.46) is sometimes referred to as the “recombination” step. IMEX schemes in [14] are described by a triplet: (implicit stages, explicit stages, time order).

7.5.0.5 IMEX methods in [246]

We consider a subset of IMEX schemes with a specific set of properties. Consider the semidiscrete problem

$$\frac{d\boldsymbol{u}_h}{dt} = f^{im}(\boldsymbol{u}_h) + f^{ex}(\boldsymbol{u}_h) \tag{7.47}$$

and the IMEX scheme for stages $i = 0, \dots, s-1$, where the stage values of $\boldsymbol{u}_h^i$ are (starting from $\boldsymbol{u}_h^k$

$$\boldsymbol{u}_h^i = \boldsymbol{u}_h^k + \Delta t\sum_{j=0}^{i} a_{ij}^{im} f^{im}(\boldsymbol{u}_h^j) + \Delta t\sum_{j=0}^{i-1} a_{ij}^{ex} f^{ex}(\boldsymbol{u}_h^j) \tag{7.48}$$

which can be suggestively written as

$$\boldsymbol{u}_h^i - \Delta t a_{ii}^{im} f^{im}(\boldsymbol{u}_h^i) = \boldsymbol{u}_h^k + \Delta t\sum_{j=0}^{i-1}\left(a_{ij}^{im} f^{im}(\boldsymbol{u}_h^j) + a_{ij}^{ex} f^{ex}(\boldsymbol{u}_h^j)\right) \tag{7.49}$$

to show that the left hand side is discretized implicitly and must be solved, whereas the right-hand side can be evaluated. The value of $\boldsymbol{u}_h^{k+1}$ is then computed as

$$\boldsymbol{u}_h^{k+1} = \boldsymbol{u}_h^k + \Delta t\sum_{i=0}^{s-1} b_i\left(f^{im}(\boldsymbol{u}_h^i) + f^{ex}(\boldsymbol{u}_h^i)\right) \tag{7.50}$$

where the coefficients c, A^{ex}, A^{im}, and b are given in the following explicit and implicit Butcher tables:

$$\begin{array}{c|cccc} 0 & 0 & \cdots & \cdots & 0 \\ c_1 & a^{ex}_{1,0} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c_{s-1} & a^{ex}_{s-1,0} & \cdots & a^{ex}_{s-1,s-2} & 0 \\ \hline & b_0 & \cdots & b_{s-2} & a \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & \cdots & \cdots & 0 \\ c_1 & a^{im}_{1,0} & a & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c_{s-1} & a^{im}_{s-1,0} & \cdots & a^{im}_{s-1,s-2} & a \\ \hline & b_0 & \cdots & b_{s-2} & a \end{array} \tag{7.51}$$

These schemes often have $b_i = b^{ex}_i = b^{im}_i$, with the exception of IMEX (1,1,1).

Some IMEX schemes (like IMEX(1,2,2) in the code) have c_i such that the last entry is not 1. This means that the IMEX scheme does not evolve the PDE from timestep k to the next time step $k+1$. This complicates things like boundary condition implementation and such schemes should practically be avoided. We assume that schemes have c_i with a final entry of 1.

Example 11. IMEX(1,1,1).
Here the implicit and explicit Butcher tables are, respectively:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 0 & 1 \\ \hline & 0 & 1 \end{array} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1 & 0 \end{array} \tag{7.52}$$

where $b^{im}_i \neq b^{ex}_i$, but yields the scheme

$$\boldsymbol{u}^{k+1}_h = \boldsymbol{u}^k_h + \Delta t \left(f^{ex}(\boldsymbol{u}^k_h) + f^{im}(\boldsymbol{u}^{k+1}_h) \right) \tag{7.53}$$

Example 12. IMEX(1,2,1).

which can be written out as the scheme

$$\begin{aligned} \boldsymbol{u}^1_h &= \boldsymbol{u}^k_h + \Delta t \left(f^{ex}\left(\boldsymbol{u}^k_h\right) + f^{im}\left(\boldsymbol{u}^1_h\right)\right), \\ \boldsymbol{u}^{k+1}_h &= \boldsymbol{u}^k_h + \Delta t \left(f^{ex}\left(\boldsymbol{u}^1_h\right) + f^{im}\left(\boldsymbol{u}^1_h\right)\right). \end{aligned} \tag{7.54}$$

7.5.0.6 Linear multi-step methods

The material in this section is largely derived from and inspired by the lecture material in [196]. For the system of ODEs in equation (7.20), a linear k-step method is defined as

$$\sum_{j=0}^{k} \alpha_j \boldsymbol{y}_{n+j} = \Delta t \sum_{j=0}^{k} \beta_j r(\boldsymbol{y}_{n+j}), \qquad n = 0, 1, \ldots, N-k, \tag{7.55}$$

where $\boldsymbol{y}_{n-k}$ is the unknown of the system and $\boldsymbol{y}_{n+j}$ are approximations to $\boldsymbol{y}(t_{n+j})$. The coefficient $\alpha_k = 1$, and the other coefficients have to be determined. The method is implicit if $\beta_k \neq 0$, and is explicit if $\beta_k = 0$. These methods are not self-starting, and need the approximations $\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}$ to start. Different choices of coefficient sets $(\alpha_j,\ \beta_j)$, $0 \leq j \leq k$ lead to different multi-step methods.

7.5.0.7 Backward difference formula (BDF) methods

BDF methods are characterized by $\beta_j = 0$ for $j < k$. BDF methods of order J are implicit, and require the solution of a linear or non-linear system for the unknown $\boldsymbol{y}_{n+1}$. They can be written in a more specific way than in equation (7.55):

$$\frac{\gamma_0 \boldsymbol{y}^{n+1} - \sum_{i=0}^{J-1} \left(\alpha_i \boldsymbol{y}^{n-i} \right)}{\Delta t} = r\left(\boldsymbol{y^{n+1}}, t_{n+1} \right) \tag{7.56}$$

where the leading coefficient $\gamma_0 = 1$. BDF methods are A-stable up to order 2.

Example 13. BDF schemes for scalar ODE equation (7.19).

BDF1 (backward Euler):

$$\frac{y_{n+1}-y_n}{\Delta t}=f(y_{n+1},t_{n+1}) \tag{7.57}$$

where we see that $\gamma_0=\alpha_0=1$

BDF2:

$$\frac{y_{n+2}-\frac{4}{3}y_{n+1}+\frac{1}{3}y_n}{\Delta t}=\frac{2}{3}f(y_{n+2},t_{n+2}) \tag{7.58}$$

7.5.0.8 Implementations

A clean interface for systems of ODEs from a software engineering perspective is to have a time integrator take in a callable function which can evaluate the RHS.

7.5.0.9 DIRK methods

In this section, we provide additional information pertaining specifically to DIRK schemes; a more detailed summary is given in the review paper [123].

We start from the vector form of equation (7.23), making a suggestive change of notation to $\boldsymbol{u}$, since our primary application of interest will be stiff systems of ODEs that arise as a result of a discretization of a PDE.

$$\begin{aligned}\boldsymbol{u}^{n+1}&=\boldsymbol{u}^n+\Delta t\sum_{i=1}^{s}b_i\boldsymbol{k}_i\\ \boldsymbol{k}_i&=\mathcal{L}_h\left(\boldsymbol{u}_n+\Delta t\sum_{j=1}^{s}a_{ij}\boldsymbol{k}_j,\quad t_n+c_i\Delta t\right)\end{aligned} \tag{7.59}$$

Using the structure of the matrix A in Butcher tables for DIRK schemes, we can separate the implicit portion of the computation

$$\begin{aligned}\boldsymbol{k}_i&=\mathcal{L}_h\left(\boldsymbol{u}^n+\Delta t\sum_{j=1}^{i-1}a_{ij}\boldsymbol{k}_j+\Delta t a_{ii}\boldsymbol{k}_i,\ \ t_n+c_i\Delta t\right)\\ &=\mathcal{L}_h\left(\boldsymbol{x}_i+a_{ii}\Delta t\boldsymbol{k}_i,\ \ t_n+c_i\Delta t\right),\end{aligned} \tag{7.60}$$

where $\boldsymbol{x}_i=\boldsymbol{u}^n+\Delta t\sum_{j=1}^{i-1}a_{ij}\boldsymbol{k}_j$, and can be computed from data. If the operator $\mathcal{L}_h$ is nonlinear, no further simplification can be written. However, if the operator $\mathcal{L}_h$ is linear, we can isolate the implicit problem further from the data

$$\boldsymbol{k}_i=a_{ii}\Delta t\mathcal{L}_h\left(\boldsymbol{k}_i,\,t_n+c_i\Delta t\right)+\mathcal{L}_h\left(\boldsymbol{x}_i,\,t_n+c_i\Delta t\right)$$

yielding the problem

$$\frac{1}{a_{ii}\Delta t}\boldsymbol{k}_i-\mathcal{L}_h\left(\boldsymbol{k}_i,\,t_n+c_i\Delta t\right)=\frac{1}{a_{ii}\Delta t}\mathcal{L}_h\left(\boldsymbol{x}_i,\,t_n+c_i\Delta t\right). \tag{7.61}$$

Although the form of equation (7.61) matches the residual form of general RK methods, an implementation should take as initialization a function which takes as input the constants a_{ii} and Δt, as well as the data $\boldsymbol{u}^n$ and $\boldsymbol{x}_i$ and returns the solution to the implicit problem in equation (7.61). We have to store the stage values $\boldsymbol{k}_i$ in order to compute $\boldsymbol{x}_i$.

7.5.0.10 Numerical experiments

We take as a numerical test case the differential equation

$$\frac{dy}{dt} = t\sqrt{y} \tag{7.62}$$

with initial condition $t_0 = 0$ and $y(0) = y_0 = 1$; it is straightforward to verify that the analytical solution to this ODE is

$$y(t) = \frac{1}{16}\left(t^2+4\right)^2,$$

to which we compare the numerical solution at the final time $T = 10$. The residual function $\mathcal{L}_h(y,t) = t\sqrt{y}$ is sufficient to construct an explicit RK scheme of choice.

Deriving an implicit scheme requires some more work. Since the operator $\mathcal{L}_h$ is nonlinear, we have to solve the nonlinear equation (7.60) at each time stage, namely,

$$k_i = \mathcal{L}_h\left(y^n + \Delta t\sum_{j=1}^{i-1} a_{ij}k_j + a_{ii}\Delta t k_i,\; t_n + c_i\Delta t\right),$$

corresponding to the nonlinear equation

$$f(k_i) = k_i - t^*\sqrt{x_i + a_{ii}\Delta t k_i} \quad = 0, \tag{7.63}$$

where the data $x_i = y^n + \sum_{j=1}^{i-1} a_{ij}k_j$ and the time $t^* = t_n + c_i\Delta t$ are both known. The nonlinear equation (7.63) can be solved by using a Newton iteration, advancing

$$k_i^{(m+1)} = k_i^{(m)} - \frac{f\left(k_i^{(m)}\right)}{f'\left(k_i^{(m)}\right)}, \qquad \text{with } f'(k_i) = 1 - \frac{1}{2}t^*\left(x_i + a_{ii}\Delta t k_i\right)^{-1/2}\cdot a_{ii}\Delta t \tag{7.64}$$

and where the integer m corresponds to Newton iteration.

The results for different ERK and DIRK methods are shown in figure 7-2. All expected convergence rates are obtained.

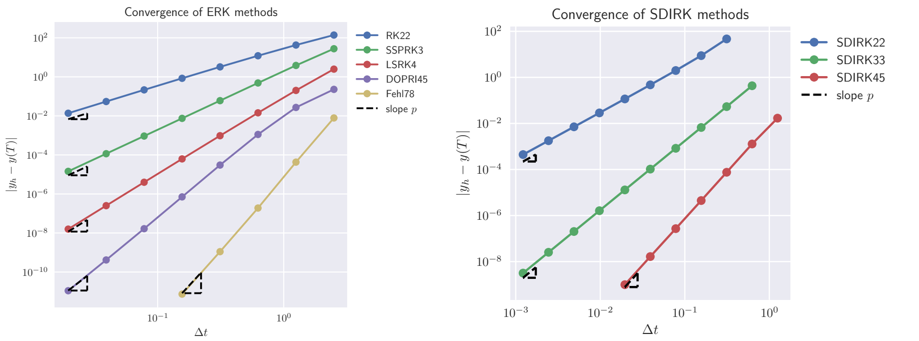


Figure 7-2: Convergence of ERK (left) and DIRK (right) solvers for the scalar ODE test problem in equation 7.62. Comparison slopes are 5 and 7 for the DOPRI45 and Fehlenberg78 schemes, respectively.

7.5.0.11 A prototypical "stiff" ODE

We take as a numerical test case the differential equation

$$\frac{dy}{dt} = -25y \tag{7.65}$$

with initial condition $t_0 = 0$ and $y(0) = 1$. Equation 7.65 has the exact solution

$$y(t) = e^{-25t},$$

and has the behavior that $y(t) \to 0$ as $t \to \infty$. In this case, the ODE is linear, and the implicit time integration scheme can be written in closed form without need for a Newton iteration. Equation (7.60) can be simplified using the linearity of the residual $\mathcal{L}_h(y, t) = -25y$ to yield

$$k_i = \frac{1}{1 + 25a_{ii}\Delta t}\mathcal{L}_h(x_i, t^*)$$

at each time integration stage, where the data $x_i = y^n + \sum_{j=1}^{i-1} a_{ij}k_j$ and the time $t^* = t_n + c_i\Delta t$ are both known. The numerical solution $y_h(t)$ is given in figure 7-3. At coarse timestep sizes, the implicit time integration scheme vastly outperforms the explicit scheme. However, as the timestep is refined, the performance of the explicit and implicit scheme becomes comparable.

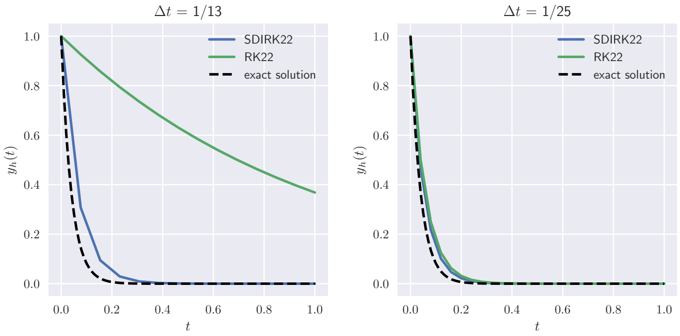


Figure 7-3: Numerical solution of stiff ODE in equation (7.65) using RK22 and SDIRK22.

7.6 Convergence history for the pure Neumann problem

We present the error and convergence history in L^2-norm corresponding to the plots in Figure 2-21 in Table 7.1 for the numerical solution of the pure Neumann problem using the method of subspace projection for polynomial orders $p = 1, \ldots, 6$. All results are reported in the asymptotic regime, before errors associated with floating point precision begin to pollute the convergence order. Optimal convergence of $p+1$ for both u_h and $\boldsymbol{q}_h$ were achieved in all test cases for all dimensions.

Degree	Mesh	$\|\|u - u_h\|\|_{L^2(\Omega)}$		$\|\|\boldsymbol{q} - \boldsymbol{q}_h\|\|_{L^2(\Omega)}$	
p	N_K	error	order	error	order
1	1	1.323e+01	-	1.180e+01	-
	4	2.077e+00	2.67	3.348e+00	1.82
	16	9.620e-01	1.11	1.785e+00	0.91
	64	2.134e-01	2.17	3.911e-01	2.19
	256	5.828e-02	1.87	1.044e-01	1.91
	1024	1.514e-02	1.94	2.685e-02	1.96
	4096	3.851e-03	1.97	6.801e-03	1.98
	16384	9.709e-04	1.99	1.711e-03	1.99
	65536	2.437e-04	1.99	4.291e-04	2.00
2	1	4.377e+00	1.77	7.488e+00	1.76
	4	1.268e+00	1.79	2.488e+00	1.59
	16	4.212e-02	4.91	1.270e-01	4.29
	64	2.740e-02	0.62	5.035e-02	1.33
	256	3.642e-03	2.91	6.592e-03	2.93
	1024	4.668e-04	2.96	8.395e-04	2.97
	4096	5.901e-05	2.98	1.058e-04	2.99
	16384	7.415e-06	2.99	1.327e-05	2.99
	65536	9.292e-07	3.00	1.662e-06	3.00
3	1	1.370e+00	2.56	2.964e+00	2.60
	4	1.056e-01	3.70	3.524e-01	3.07
	16	5.215e-02	1.02	9.719e-02	1.86
	64	2.667e-03	4.29	4.912e-03	4.31
	256	1.748e-04	3.93	3.185e-04	3.95
	1024	1.114e-05	3.97	2.019e-05	3.98
	4096	7.019e-07	3.99	1.269e-06	3.99
	16384	4.404e-08	3.99	7.955e-08	4.00
4	1	1.584e+00	3.64	3.446e+00	3.67
	4	1.858e-01	3.09	3.587e-01	3.26
	16	1.330e-03	7.13	3.375e-03	6.73
	64	2.086e-04	2.67	3.848e-04	3.13
	256	6.778e-06	4.94	1.240e-05	4.96
	1024	2.151e-07	4.98	3.919e-07	4.98
	4096	6.767e-09	4.99	1.230e-08	4.99
	16384	2.121e-10	5.00	3.853e-10	5.00
5	1	2.630e-01	4.40	8.666e-01	4.51
	4	9.138e-03	4.85	2.661e-02	5.03
	16	1.106e-03	3.05	2.060e-03	3.69
	64	1.362e-05	6.34	2.516e-05	6.36
	256	2.200e-07	5.95	4.034e-07	5.96
	1024	3.482e-09	5.98	6.364e-09	5.99
	4096	5.470e-11	5.99	9.984e-11	5.99
6	1	5.639e-01	4.98	1.158e+00	4.84
	4	1.217e-02	5.53	2.323e-02	5.64
	16	1.995e-05	9.25	4.649e-05	8.96
	64	7.628e-07	4.71	1.411e-06	5.04
	256	6.134e-09	6.96	1.127e-08	6.97
	1024	4.846e-11	6.98	8.880e-11	6.99

Table 7.1: Convergence history for the singular pure Neumann problem using subspace projection (2D).

7.7 Derivation of free-surface corrector boundary conditions

The conceptual basis of the projection scheme is the idea that we can define a "total effective nonhydrostatic pressure" which has component due to the free-surface η and a component due to the nonhydrostatic pressure p'.

$$\mathrm{P}'_{\text{eff}} = p' + g\eta, \tag{7.66}$$

which explains the terms $\nabla \mathrm{P}' = \nabla p' + g\nabla\eta$ in the momentum equation (3.1). The free-surface corrector and intermediate projection step enforces the free-surface continuity equation and projects out the velocity divergence due to the free-surface pressure $g\eta$. The pressure-corrector step projects out the velocity divergence due to the nonhydrostatic pressure p'.

To enforce the free-surface evolution equation (3.3), we approximate $\boldsymbol{u}^{k+1}$ by $\bar{\bar{\boldsymbol{u}}}^{k+1} = \bar{\boldsymbol{u}}^{k+1} - a\Delta t \nabla_{xy}\left(g\delta\eta^{k+1}\right)$:

$$\begin{aligned} \frac{\eta^{k+1}}{a\Delta t} + \nabla\cdot\int_{-H}^{\eta}\left(\bar{\boldsymbol{u}}^{k+1} - a\Delta t\nabla_{xy}\left(g\delta\eta^{k+1}\right)\right)\,dz &= \frac{\eta^{k}}{a\Delta t} \\ \Rightarrow \frac{\delta\eta^{k+1}}{a\Delta t} - \nabla\cdot\left(a\Delta t g\left[\eta^{k} + H\right]\nabla\delta\eta^{k+1}\right) &= -\nabla\cdot\int_{-H}^{\eta}\bar{\boldsymbol{u}}^{k+1}\,dz. \end{aligned} \tag{7.67}$$

For a Dirichlet boundary condition on the velocity $\boldsymbol{u}^{k+1}$, making use of the definition of $\bar{\bar{\boldsymbol{u}}}^{k+1}$, we have

$$\begin{aligned} \nabla\delta\eta^{k+1} &= \frac{\bar{\boldsymbol{u}}^{k+1} - \bar{\bar{\boldsymbol{u}}}^{k+1}}{a\Delta t g} \\ \Rightarrow g_{N,\delta\eta} = \nabla\delta\eta^{k+1}\cdot\boldsymbol{n} &= \frac{1}{a\Delta t g}\left(\bar{\boldsymbol{u}}^{k+1} - \boldsymbol{g}_{D,\boldsymbol{u}}\right)\cdot\boldsymbol{n} \end{aligned} \tag{7.68}$$

which enforces that the horizontal component of the second intermediate velocity $\bar{\bar{\boldsymbol{u}}}^{k+1}$ has the value $\boldsymbol{g}_{D,\boldsymbol{u}}$ on the boundary Γ_D after the intermediate updates[1]. In the case where we were able to enforce that $\bar{\boldsymbol{u}}^{k+1} = \boldsymbol{g}_{D,\boldsymbol{u}}$ exactly, this reduces to a zero-Neumann condition; however, since HDG methods may only impose boundary conditions weakly through the edge-space M_h^p, the value $g_{N,\delta\eta}$ may not be exactly zero numerically. However, since this condition is enforced only at the surface, the difference can only be enforced in a depth-averaged sense, and it may be better to take an exact zero-Neumann condition and rather enforce the condition on $\bar{\boldsymbol{u}}^{k+1}$ exactly instead.

The first velocity predictor is like a classical projection method—use $p'^{,k}$ instead of $p'^{,k+1}$.

[1] The vertical component of the second intermediate velocity $\bar{\bar{\boldsymbol{u}}}^{k+1}$ is not corrected by the free-surface updates.

7.8 2D Projection method reduction

Nonhydrostatic equations.

Velocity predictor:

$$\begin{aligned}
&\frac{\partial \bar{u}^{k+1}}{\partial t} - \frac{\partial}{\partial x}\left(\nu_x \frac{\partial \bar{u}^{k+1}}{\partial x}\right) - \frac{\partial}{\partial z}\left(\nu_z \frac{\partial \bar{u}^{k+1}}{\partial z}\right) = -\frac{\partial p^{\prime,k}}{\partial x} - g\frac{\partial \eta^k}{\partial x} - \frac{g}{\rho_0}\int_z^{\eta^k} \frac{\partial \rho^{\prime,k}}{\partial x}\, dz' - \nabla\cdot\left(u^k \otimes \boldsymbol{u}^k\right) - \frac{1}{\rho_0} f_x \\
&\frac{\partial \bar{w}^{k+1}}{\partial t} - \frac{\partial}{\partial x}\left(\nu_x \frac{\partial \bar{w}^{k+1}}{\partial x}\right) - \frac{\partial}{\partial z}\left(\nu_z \frac{\partial \bar{w}^{k+1}}{\partial z}\right) = -\frac{\partial p^{\prime,k}}{\partial z} - \nabla\cdot\left(v^k \otimes \boldsymbol{u}^k\right) - \frac{1}{\rho_0} f_z
\end{aligned} \tag{7.69}$$

Free-surface correction:

$$\frac{\delta \eta^{k+1}}{a\Delta t} - \nabla\cdot\left(a\Delta t g\left(\eta^k + H\right)\nabla\delta\eta^{k+1}\right) = -\nabla\cdot\int_{-H}^{\eta^k} \bar{u}^{k+1}\, dz \tag{7.70}$$

Intermediate updates:

$$\begin{aligned}
\bar{\bar{u}}^{k+1} &= \bar{u}^{k+1} - a\Delta t g\frac{\partial}{\partial x}\delta\eta^{k+1} = \bar{u}^{k+1} - \frac{q_{\delta\eta,\,x}^{k+1}}{H + \eta^k} \\
\bar{\bar{w}}^{k+1} &= \bar{w}^{k+1} \\
\eta^{k+1} &= \eta^k + \delta\eta^k
\end{aligned} \tag{7.71}$$

Pressure correction:

$$-\nabla\cdot\left(\nabla\delta p^{\prime,k+1}\right) = -\frac{\nabla\cdot\bar{\bar{\boldsymbol{u}}}^{k+1}}{a\Delta t} \tag{7.72}$$

Final velocity and pressure corrections:

$$\begin{aligned}
\boldsymbol{u}^{k+1} &= \bar{\bar{\boldsymbol{u}}}^{k+1} - a\Delta t\nabla\delta p^{\prime,k+1} \\
p^{\prime,k+1} &= p^{\prime,k} + \delta p^{\prime,k+1}
\end{aligned} \tag{7.73}$$

Density perturbation update:

$$\frac{\partial \rho'}{\partial t} - \nabla\cdot\left(\kappa\nabla\rho'\right) = -\nabla\cdot\left(\boldsymbol{u}\rho'\right) + f_{\rho'} \tag{7.74}$$

Hydrostatic equations.

Velocity predictor:

$$\begin{aligned}
&\frac{\partial \bar{u}^{k+1}}{\partial t} - \frac{\partial}{\partial x}\left(\nu_x \frac{\partial \bar{u}^{k+1}}{\partial x}\right) - \frac{\partial}{\partial z}\left(\nu_z \frac{\partial \bar{u}^{k+1}}{\partial z}\right) = -g\frac{\partial \eta^k}{\partial x} - \frac{g}{\rho_0}\int_z^{\eta^k} \frac{\partial \rho^{\prime,k}}{\partial x}\, dz' - \nabla\cdot\left(u^k \otimes \boldsymbol{u}^k\right) - \frac{1}{\rho_0} f_x \\
&\frac{\partial \bar{w}^{k+1}}{\partial t} - \frac{\partial}{\partial x}\left(\nu_x \frac{\partial \bar{w}^{k+1}}{\partial x}\right) - \frac{\partial}{\partial z}\left(\nu_z \frac{\partial \bar{w}^{k+1}}{\partial z}\right) = -\nabla\cdot\left(v^k \otimes \boldsymbol{u}^k\right) - \frac{1}{\rho_0} f_z
\end{aligned} \tag{7.75}$$

Free-surface correction:

$$\frac{\delta \eta^{k+1}}{a\Delta t} - \nabla\cdot\left(a\Delta t g\left(\eta^k + H\right)\nabla\delta\eta^{k+1}\right) = -\nabla\cdot\int_{-H}^{\eta^k} \bar{u}^{k+1}\, dz \tag{7.76}$$

Updates:

$$\begin{aligned} u^{k+1} &= \bar{u}^{k+1} - a\Delta t g \frac{\partial}{\partial x}\delta\eta^{k+1} = \bar{u}^{k+1} - \frac{q_{\delta\eta,\,x}^{k+1}}{H+\eta^k} \\ \eta^{k+1} &= \eta^k + \delta\eta^k \end{aligned} \tag{7.77}$$

Vertical-velocity recovery:

$$\frac{\partial w^{k+1}}{\partial z} = -\frac{\partial u^{k+1}}{\partial x} \tag{7.78}$$

Density perturbation update:

$$\frac{\partial \rho'}{\partial t} - \nabla\cdot(\kappa\nabla\rho') = -\nabla\cdot(\boldsymbol{u}\rho') + f_{\rho'} \tag{7.79}$$

7.9 Deep reinforcement learning: tunable policies

Computational resources are user- and time-dependent. For different applications, or even in the middle of a simulation, we may seek to adjust how aggressive our reinforcement learning agent is in refining or coarsening our mesh. We directly encode the trade-off between accuracy and computational cost in hyperparameter γ_c, which the user may tune freely. However, naively changing γ_c would require re-training our reinforcement learning agent. Next, we show how we can avoid re-training while simultaneously allowing for policy tuning.

The more general setting to this problem is multi-objective reinforcement learning (MORL). We seek an agent that is optimized over a continuum of objective functions depending upon γ_c. The value function $V_t^\pi(s_t)$ is the expected reward following policy π given state s_t. Traditionally, $V_t^\pi : S \to \mathbb{R}$ maps an observation to a scalar, but in MORL, the value function $V_t^\pi : S \to \mathbb{R}^d$ maps an observation to a d-dimensional vector indexed by the different objective functions. In our case, the value function $V_t^\pi(s_t; \gamma_c) : S \to \mathcal{C}_b(\mathbb{R})$ maps an observation to a bounded continuous function. Most approaches in the MORL literature attempt to find either a single policy by scalarizing the vector-valued V_t^π or multiple policies by repeatedly training over different objective functions (see [109] for a literature review). However, due to a particular feature of our expected reward function Q_t, we are able to simply learn over a continuum of objective functions.

Consider an expected reward function $Q_t : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ that can be split into two parts: a function that can be explicitly computed given the observation, i.e. the "known" function $Q^{(k)}$, and a function that needs to be learned, $Q^{(l)}$.

$$Q_t(s, a; \gamma_c) = Q_t^{(k)}(s, a; \gamma_c) + Q_t^{(l)}(s, a) \tag{7.80}$$

There is no reason to learn $Q^{(k)}$ if it can be explicitly computed; instead, we should just represent $Q^{(l)}$ with a deep neural network as in deep Q-learning. Then, the outputs of the two Q functions can be combined at the end, and the argmax of the resulting sum will be the action that we take. Importantly, the hyperparameter γ_c is only an argument of the known function that we can explicitly compute. This allows us to learn one function Q^l but then change our policy adaptively as a function γ_c. Computationally, we really learn the function $\hat{Q}_t : S \to \mathbb{R}^{|\mathcal{A}|}$ that maps an observation to the expected reward for every action in the action space, and we denote this modified and approximate expected reward function with a hat. In Figure 7-4, we delineate a traditional deep Q-learning policy π from a tunable policy $\tilde{\pi}$.

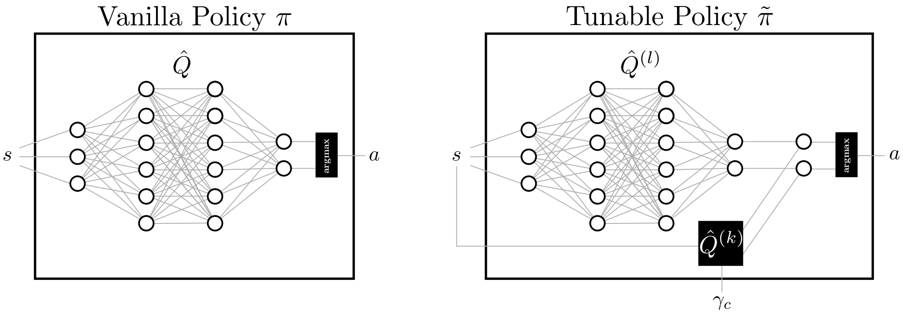


Figure 7-4: We depict a vanilla policy based on Deep Q-Learning and a tunable policy. The tunable policy splits the $\hat{Q}$ function into known and learned functions. The learned function is represented by a neural network, while the known policy can be evaluated explicitly given the observations. In the tunable policy, the hyperparameter γ_c is an input to the known $\hat{Q}^{(k)}$ function which does not have to be re-learned.

In our particular case, the reward function (5.2) is naturally decomposed into a part that needs to be learned and one that can be explicitly computed: the accuracy needs to be learned but the computational cost and barrier function are explicitly known. Call these partial rewards $R^{(l)}$ and

$R^{(k)}$, respectively. We can write our Q function as follows.

$$Q_t(s,a) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma_k^i R_{t+i+1}^{(k)} + \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}^{(l)} \middle| S_t = s, A_t = a\right] \tag{7.81}$$

Above, we have used different discount factors γ_k and γ for $R^{(k)}$ and $R^{(l)}$, respectively. We consider the case where $\gamma_k = 0$. In our example, setting the discount factor on the computational cost to zero is justified in that we are usually only concerned with the computational cost at a given instant. Furthermore, the computational load on a machine is often highly unpredictable; for example, other users may submit jobs in the middle of our simulation. As such, a greedy approach to discounting the computational penalty is desirable, and we may rewrite our expected reward as follows.

$$Q_t(s,a) = \underbrace{\mathbb{E}\left[R_{t+1}^{(k)} \middle| S_t = s, A_t = a\right]}_{Q_t^{(k)}} + \underbrace{\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i+1}^{(l)} \middle| S_t = s, A_t = a\right]}_{Q_t^{(l)}} \tag{7.82}$$

With that, we have defined our known and learned Q functions, and to train we simply omit $R^{(k)}$ from our reward entirely by setting $\gamma_c = 0$. Once trained, we append $Q^{(k)}$ to the end of our $Q^{(k)}$ network in order to make predictions, and γ_c may be set arbitrarily without retraining. We note that separable rewards are not limited to this case where we balance computational cost and simulation accuracy, and we posit that there are many other areas where this framework may be beneficial.

While we have described how to perform MORL in instances where Q is separable and the action space is discrete, this framework should be extendable to the continuous case. In actor-critic policies where the critic learns the Q function (e.g. deep deterministic policy gradient (DDPG) [166], twin delayed DDPG (TD3) [86], and soft actor critic (SAC) [99] methods), one should be able to apply a similar methodology: after learning $Q^{(l)}$, one should be able to append $Q^{(k)}$. In the continuous case, however, the actor must be re-trained since we cannot just take the argmax of a continuous action space. Fortunately, re-training the actor should be computationally inexpensive since the policy networks tend to be small, and we can just randomly sample from the action space. That is, we do not need to re-sample from our environment; instead, as we already know our Q function, we simply need to re-train the actor to maximize the estimated reward given an action from our action space. This procedure is the subject of ongoing research.

7.10 Exact solution, §5.4.6

The exact solution is given by

$$u(\boldsymbol{x}) = \sum_{i=1}^{25} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{\sigma^2} \left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|^2\right)$$

with parameter $\sigma = 1/10$ and source centers $\boldsymbol{x}_i$ described in Table 7.2.

i	$\boldsymbol{x_i} = (x_1, x_2/\alpha)$								
1	(-0.75, -0.25)	6	(-0.25, -0.25)	11	(0.125, -0.75)	16	(-0.75, -0.625)	21	(-0.25, -0.375)
2	(-0.75, -0.75)	7	(-0.25, -0.50)	12	(0.5, -0.75)	17	(-0.50, -0.625)	22	(0.125, -0.625)
3	(-0.75, -0.50)	8	(-0.25, -0.75)	13	(0.5, -0.5)	18	(-0.25, -0.625)	23	(0.5 , -0.625)
4	(-0.50, -0.25)	9	(0.125, -0.25)	14	(0.5 , -0.25)	19	(-0.75, -0.375)	24	(0.625 ,-0.25)
5	(-0.50, -0.50)	10	(0.125, -0.50)	15	(0.75 , -0.25)	20	(-0.50, -0.375)	25	(0.125, -0.375)

Table 7.2: Description of source centers, where $\alpha = 1.1$.

7.11 Deep reinforcement learning: additional training parameters

For all numerical experiments, we used a time discount factor $\gamma = 0.99$, which is the default for the Stable-Baselines-3 library. The large negative reward accrued by the agent upon running out of computational resources was taken to be $R_{\text{exceed}} = -1 \cdot 10^3$; this was important as a learning signal in the cases where the RL agent was trained on a small computational budget, as the barrier function $B(p)$ in (5.2) is undefined outside $p = 1$. Training episodes were terminated after a finite number of iterations, we used 200; this choice is arbitrary, as long as the number of iterations is large enough that it is possible for the agent to approach its computational budget and receive the large negative reward for doing so. The parameters used for RL training were given in Table 5.2.

Bibliography

[1] Alistair Adcroft, Chris Hill, Jean-Michel Campin, John Marshall, and Patrick Heimbach. Overview of the formulation and numerics of the MIT GCM. *Proceedings of the ECMWF Seminar Series on Numerical Methods, Recent Developments in Numerical Methods for Atmosphere and Ocean Modeling*, January 2004.

[2] Mark Ainsworth and J Tinsley Oden. A posteriori error estimation in finite element analysis. *Computer methods in applied mechanics and engineering*, 142(1-2):1–88, 1997. Publisher: Elsevier.

[3] Wael Hajj Ali, Manmeet S. Bhabra, Pierre F. J. Lermusiaux, Andrew March, Joseph R. Edwards, Katherine Rimpau, and Paul Ryu. Stochastic oceanographic-acoustic prediction and Bayesian inversion for wide area ocean floor mapping. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, Seattle, October 2019. IEEE.

[4] Wael Hajj Ali, Aaron Charous, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux. MSEAS-ParEq for ocean-acoustic modeling around the globe. In *OCEANS 2023 IEEE/MTS Gulf Coast*, Biloxi, MS, September 2023. IEEE. In press.

[5] Wael Hajj Ali, Yorand Gao, Corbin Foucart, Manan Doshi, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux. High-performance visualization for ocean modeling. In *OCEANS 2022 IEEE/MTS*, pages 1–10, Hampton Roads, VA, October 2022. IEEE.

[6] Wael Hajj Ali, Mohamad H. Mirhi, Abhinav Gupta, Chinmay S. Kulkarni, Corbin Foucart, Manan M. Doshi, Deepak N. Subramani, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux. Seavizkit: Interactive maps for ocean visualization. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, Seattle, October 2019. IEEE.

[7] Jean Aoussou, Jing Lin, and Pierre F. J. Lermusiaux. Iterated pressure-correction projection methods for the unsteady incompressible Navier–Stokes equations. *Journal of Computational Physics*, 373:940–974, November 2018.

[8] David Aristoff and Wolfgang Bangerth. A benchmark for the Bayesian inversion of coefficients in partial differential equations, February 2022. arXiv:2102.07263 [cs, math].

[9] Daniel Arndt, Wolfgang Bangerth, Denis Davydov, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Jean-Paul Pelteret, Bruno Turcksin, and David Wells. The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 81:407–422, January 2021.

[10] Daniel Arndt, Wolfgang Bangerth, Marco Feder, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Simon Sticko, Bruno Turcksin, and David Wells. The deal.II library, Version 9.4. *Journal of Numerical Mathematics*, 30(3):231–246, September 2022. Publisher: De Gruyter.

[11] Douglas N Arnold. An interior penalty finite element method with discontinuous elements. *SIAM journal on numerical analysis*, 19(4):742–760, 1982. Publisher: SIAM.

[12] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, January 2002.

[13] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. Publisher: IEEE.

[14] Uri M Ascher, Steven J Ruuth, and Raymond J Spiteri. Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151–167, 1997.

[15] H Atkins and H Atkins. Continued development of the discontinuous Galerkin method for computational aeroacoustic applications. In *3rd AIAA/CEAS Aeroacoustics Conference*, page 1581, 1997.

[16] Harold L Atkins and Chi-Wang Shu. Quadrature-free implementation of discontinuous Galerkin method for hyperbolic equations. *AIAA Journal*, 36(5):775–782, 1998.

[17] Francis Auclair, Claude Estournel, Jochem W. Floor, Marine Herrmann, Cyril Nguyen, and Patrick Marsaleix. A non-hydrostatic algorithm for free-surface ocean modelling. *Ocean Modelling*, 36(1):49–70, January 2011.

[18] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, March 1996. Google-Books-ID: hNpJg_pUsOwC.

[19] Ivo Babuska, John Whiteman, and Theofanis Strouboulis. *Finite elements: an introduction to the method and error estimation*. Oxford University Press, 2010.

[20] Wolfgang Bangerth, Carsten Burstedde, Timo Heister, and Martin Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Transactions on Mathematical Software (TOMS)*, 38(2):1–28, 2012. Publisher: ACM New York, NY, USA.

[21] Francesco Bassi and Stefano Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. *Journal of computational physics*, 131(2):267–279, 1997. Publisher: Elsevier.

[22] Ş. T. Beşiktepe, P. F. J. Lermusiaux, and A. R. Robinson. Coupled physical and biogeochemical data-driven simulations of Massachusetts Bay in late summer: Real-time and post-cruise data assimilation. *Journal of Marine Systems*, 40–41:171–212, 2003.

[23] Peter Binev, Wolfgang Dahmen, and Ron DeVore. Adaptive finite element methods with convergence rates. *Numerische Mathematik*, 97(2):219–268, 2004. Publisher: Springer.

[24] Christopher M Bishop. *Pattern Recognition and Machine Learning*.

[25] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[26] Pavel Bochev and R. B. Lehoucq. On the Finite Element Solution of the Pure Neumann Problem. *SIAM Review*, 47(1):50–66, January 2005. Publisher: Society for Industrial and Applied Mathematics.

[27] Guido Boffetta and Andrea Mazzino. Incompressible Rayleigh–Taylor Turbulence. *Annual Review of Fluid Mechanics*, 49(1):119–143, January 2017.

[28] Jan Bohn and Michael Feischl. Recurrent neural networks as optimal mesh refinement strategies. *Computers & Mathematics with Applications*, 97:61–76, 2021. Publisher: Elsevier.

[29] Jesus Bonilla and Santiago Badia. Monotonicity-preserving finite element schemes with adaptive mesh refinement for hyperbolic problems. *Journal of Computational Physics*, 416:109522, 2020. Publisher: Elsevier.

[30] A Bourgeat and B Cockburn. The TVD-projection method for solving implicit numeric schemes for scalar conservation laws: A numerical study of a simple case. 1987.

[31] Dietrich Braess, Carsten Carstensen, and Ronald HW Hoppe. Convergence analysis of a conforming adaptive finite element method for an obstacle problem. *Numerische Mathematik*, 107(3):455–471, 2007. Publisher: Springer.

[32] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics*. Springer, New York, NY, 1994.

[33] Susanne C Brenner, L Ridgway Scott, and L Ridgway Scott. *The mathematical theory of finite element methods*, volume 3. Springer, 2008.

[34] F Brezzi, D Boffi, L Demkowicz, RG Duran, RS Falk, and M Fortin. *Mixed finite elements, compatibility conditions, and applications*. Springer, 2008.

[35] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[36] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[37] Mark H Carpenter and Christopher A Kennedy. Fourth-order 2N-storage Runge–Kutta schemes. Technical report, NASA, 1994.

[38] Thomas C Clevenger, Timo Heister, Guido Kanschat, and Martin Kronbichler. A flexible, parallel, adaptive geometric multigrid method for fem. *ACM Transactions on Mathematical Software (TOMS)*, 47(1):1–27, 2020.

[39] B. Cockburn. Discontinuous Galerkin methods. *ZAMM*, 83(11):731–754, November 2003.

[40] B Cockburn. *Discontinuous Galerkin methods for Computational Fluid Dynamics, vol. 3 of Encyclopedia of Computational Mechanics, ch. 4*. Wiley, 2004.

[41] Bernardo Cockburn, Bo Dong, and Johnny Guzman. A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Mathematics of Computation*, 77(264):1887–1916, 2008.

[42] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Raytcho Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319–1365, 2009. Publisher: SIAM.

[43] Bernardo Cockburn, Jayadeep Gopalakrishnan, and Francisco-Javier Sayas. A projection-based error analysis of HDG methods. *Mathematics of Computation*, 79(271):1351–1367, 2010.

[44] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. The multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

[45] Bernardo Cockburn, George E. Karniadakis, Chi-Wang Shu, M. Griebel, D. E. Keyes, R. M. Nieminen, D. Roose, and T. Schlick, editors. *Discontinuous Galerkin Methods: Theory, Computation and Applications*, volume 11 of *Lecture Notes in Computational Science and Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[46] Bernardo Cockburn, Fengyan Li, and Chi-Wang Shu. Locally divergence-free discontinuous Galerkin methods for the Maxwell equations. *Journal of Computational Physics*, 194(2):588–610, 2004. Publisher: Elsevier.

[47] Bernardo Cockburn and Chi-Wang Shu. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework. *Mathematics of computation*, 52(186):411–435, 1989.

[48] Bernardo Cockburn and Chi-Wang Shu. The Runge–Kutta local projection-discontinuous-Galerkin finite element method for scalar conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 25(3):337–361, 1991. Publisher: EDP Sciences.

[49] Bernardo Cockburn and Chi-Wang Shu. The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998. Publisher: Elsevier.

[50] Bernardo Cockburn and Chi-Wang Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *Journal of scientific computing*, 16(3):173–261, 2001. Publisher: Springer.

[51] Gary Cohen, Xavier Ferrieres, and Sébastien Pernet. A spatial high-order hexahedral discontinuous Galerkin method to solve Maxwell's equations in time domain. *Journal of Computational Physics*, 217(2):340–363, 2006. Publisher: Elsevier.

[52] M. E. G. D. Colin, T. F. Duda, L. A. te Raa, T. van Zon, P. J. Haley, Jr., P. F. J. Lermusiaux, W. G. Leslie, C. Mirabito, F. P. A. Lam, A. E. Newhall, Y.-T. Lin, and J. F. Lynch. Time-evolving acoustic propagation modeling in a complex ocean environment. In *OCEANS - Bergen, 2013 MTS/IEEE*, pages 1–9, 2013.

[53] Ronald Cools. Monomial cubature rules since Stroud: a compilation–part 2. *Journal of Computational and Applied Mathematics*, 112(1-2):21–27, 1999. Publisher: North-Holland.

[54] Ronald Cools. An encyclopaedia of cubature formulas. *Journal of complexity*, 19(3):445–453, 2003. Publisher: Elsevier.

[55] G. Cossarini, P. F. J. Lermusiaux, and C. Solidoro. Lagoon of Venice ecosystem: Seasonal dynamics and environmental guidance with uncertainty analyses and error subspace data assimilation. *Journal of Geophysical Research: Oceans*, 114(C6), June 2009.

[56] Benoit Cushman-Roisin and Jean-Marie Beckers. *Introduction to geophysical fluid dynamics: physical and numerical aspects*, volume 101. Academic press, 2011.

[57] Stuart B. Dalziel. Rayleigh–Taylor instability: experiments with image analysis. *Dynamics of Atmospheres and Oceans*, 20(1-2):127–153, November 1993.

[58] Patrik Daniel, Alexandre Ern, Iain Smears, and Martin Vohralik. An adaptive hp-refinement strategy with computable guaranteed bound on the error reduction factor. *Computers & Mathematics with Applications*, 76(5):967–983, 2018. Publisher: Elsevier.

[59] Eric A D'Asaro. Generation of submesoscale vortices: A new mechanism. *Journal of Geophysical Research: Oceans*, 93(C6):6685–6693, 1988.

[60] M. K. Davey and J. A. Whitehead. Rotating Rayleigh–Taylor instability as a model of sinking events in the ocean. *Geophysical & Astrophysical Fluid Dynamics*, 17(1):237–253, January 1981.

[61] Hans De Sterck, T Manteuffel, S McCormick, Joshua Nolting, John Ruge, and Lei Tang. Efficiency-based h-and hp-refinement strategies for finite element methods. *Numerical Linear Algebra with Applications*, 15(2-3):89–114, 2008. Publisher: Wiley Online Library.

[62] Eric Deleersnijder, Vincent Legat, and Pierre F. J. Lermusiaux. Multi-scale modelling of coastal, shelf and global ocean dynamics. *Ocean Dynamics*, 60(6):1357–1359, December 2010.

[63] Eric Deleersnijder and Pierre F. J. Lermusiaux. Multi-scale modeling: nested-grid and unstructured-mesh approaches. *Ocean Dynamics*, 58(5–6):335–336, December 2008.

[64] M Delfour, W Hager, and F Trochu. Discontinuous Galerkin methods for ordinary differential equations. *Mathematics of Computation*, 36(154):455–473, 1981.

[65] B. Desjardins and E. Grenier. On Nonlinear Rayleigh-Taylor Instabilities. *Acta Mathematica Sinica*, 22(4):1007–1016, 2006.

[66] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical aspects of discontinuous Galerkin methods*, volume 69. Springer Science & Business Media, 2011.

[67] Manan M. Doshi, Manmeet S. Bhabra, and Pierre F. J. Lermusiaux. Energy-time optimal path planning in dynamic flows: Theory and schemes. *Computer Methods in Applied Mechanics and Engineering*, 405:115865, February 2023.

[68] Timothy F. Duda, Ying-Tsong Lin, W. Zhang, Bruce D. Cornuelle, and Pierre F. J. Lermusiaux. Computational studies of three-dimensional ocean sound fields in areas of complex seafloor topography and active ocean dynamics. In *Proceedings of the 10th International Conference on Theoretical and Computational Acoustics*, Taipei, 2011.

[69] Arkopal Dutt, Deepak N. Subramani, Chinmay S. Kulkarni, and Pierre F. J. Lermusiaux. Clustering of massive ensemble of vehicle trajectories in strong, dynamic and uncertain ocean flows. In *OCEANS Conference 2018*, Charleston, SC, October 2018. IEEE.

[70] Alexandre Ern, Annette F Stephansen, and Martin Vohralik. Guaranteed and robust discontinuous Galerkin a posteriori error estimates for convection–diffusion–reaction problems. *Journal of computational and applied mathematics*, 234(1):114–130, 2010. Publisher: Elsevier.

[71] Claes Eskilsson and Spencer J Sherwin. A triangular spectral/hp discontinuous Galerkin method for modelling 2D shallow water equations. *International Journal for Numerical Methods in Fluids*, 45(6):605–623, 2004. Publisher: Wiley Online Library.

[72] Geir Evensen. *Data assimilation: the ensemble Kalman filter.* Springer Science & Business Media, 2009.

[73] Maurice S Fabien, Matthew G Knepley, Richard T Mills, and Beatrice M Riviere. Manycore Parallel Computing for a Hybridizable Discontinuous Galerkin Nested Multigrid Method. *SIAM Journal on Scientific Computing*, 41(2):C73–C96, 2019. Publisher: SIAM.

[74] Sara Faghih-Naini, Sebastian Kuckuk, Vadym Aizinger, Daniel Zint, Roberto Grosso, and Harald Köstler. Quadrature-free discontinuous Galerkin method with code generation features for shallow water equations on automatically generated block-structured meshes. *Advances in Water Resources*, page 103552, 2020. Publisher: Elsevier.

[75] Amir-massoud Farahmand, Saleh Nabi, and Daniel N Nikovski. Deep reinforcement learning for partial differential equation control. In *2017 American Control Conference (ACC)*, pages 3120–3127. IEEE, 2017.

[76] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022.

[77] Niklas Fehn, Wolfgang A. Wall, and Martin Kronbichler. On the stability of projection methods for the incompressible Navier–Stokes equations based on high-order discontinuous Galerkin discretizations. *Journal of Computational Physics*, 351:392–421, December 2017.

[78] Niklas Fehn, Wolfgang A. Wall, and Martin Kronbichler. Robust and efficient discontinuous Galerkin methods for under-resolved turbulent incompressible flows. *Journal of Computational Physics*, 372:667–693, November 2018.

[79] C. Foucart, C. Mirabito, P. J. Haley, Jr., and P. F. J. Lermusiaux. Distributed implementation and verification of hybridizable discontinuous Galerkin methods for nonhydrostatic ocean processes. In *OCEANS Conference 2018*, Charleston, SC, October 2018. IEEE.

[80] Corbin Foucart. Efficient matrix-free implementation and automated verification of hybridizable discontinuous Galerkin finite element methods. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Massachusetts, June 2019.

[81] Corbin Foucart, Aaron Charous, and Pierre F. J. Lermusiaux. Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381, October 2023.

[82] Baylor Fox-Kemper. Notions for the Motions of the Oceans. In Eric P. Chassignet, Ananda Pascual, Joaquin Tintoré, and Jacques Verron, editors, *New Frontiers in Operational Oceanography*. GODAE OceanView, August 2018.

[83] Baylor Fox-Kemper, Alistair Adcroft, Claus W. Böning, Eric P. Chassignet, Enrique Curchitser, Gokhan Danabasoglu, Carsten Eden, Matthew H. England, Rüdiger Gerdes, Richard J. Greatbatch, Stephen M. Griffies, Robert W. Hallberg, Emmanuel Hanert, Patrick Heimbach, Helene T. Hewitt, Christopher N. Hill, Yoshiki Komuro, Sonya Legg, Julien Le Sommer, Simona Masina, Simon J. Marsland, Stephen G. Penny, Fangli Qiao, Todd D. Ringler, Anne Marie Treguier, Hiroyuki Tsujino, Petteri Uotila, and Stephen G. Yeager. Challenges and Prospects in Ocean Circulation Models. *Frontiers in Marine Science*, 6:65, February 2019.

[84] Mara A Freilich and Amala Mahadevan. Decomposition of vertical velocity for nutrient transport in the upper ocean. *Journal of Physical Oceanography*, 49(6):1561–1575, 2019.

[85] Oliver B Fringer, James C Mcwilliams, and Robert L Street. ADVANCES IN COMPUTATIONAL OCEANOGR APHY.

[86] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, July 2018.

[87] K Fukunaga. Introduction to Statistical Pattern Recognition. Elsevier Academic Press. San Diego, San Francisco, New York. 1990.

[88] T. Furevik, M. Bentsen, H. Drange, I. K. T. Kindem, N. G. Kvamsto, and A. Sorteberg. Description and evaluation of the bergen climate model: ARPEGE coupled with MICOM. *Climate Dynamics*, 21(1):27–51, July 2003.

[89] Avijit Gangopadhyay, Pierre F.J. Lermusiaux, Leslie Rosenfeld, Allan R. Robinson, Leandro Calado, Hyun Sook Kim, Wayne G. Leslie, and Patrick J. Haley, Jr. The California Current system: A multiscale overview and the development of a feature-oriented regional modeling system (FORMS). *Dynamics of Atmospheres and Oceans*, 52(1–2):131–169, September 2011. Special issue of Dynamics of Atmospheres and Oceans in honor of Prof. A. R. Robinson.

[90] Youran Gao, Wael Hajj Ali, Corbin Foucart, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux. 3DSeaVizKit: An interactive spatiotemporal visualization toolkit for ocean data. *Journal of Atmospheric and Oceanic Technology*, 2023. In preparation.

[91] G Gassner, C Altmann, F Hindenlang, M Staudenmeier, and CD Munz. Explicit discontinuous Galerkin schemes with adaptation in space and time. *36th CFD/ADIGMA course on hp-adaptive and hp-multigrid methods, VKI LS*, 2009.

[92] F.X. Giraldo, J.S. Hesthaven, and T. Warburton. Nodal High-Order Discontinuous Galerkin Methods for the Spherical Shallow Water Equations. *Journal of Computational Physics*, 181(2):499–525, September 2002.

[93] Nickolay Y. Gnedin, Vadim A. Semenov, and Andrey V. Kravtsov. Enforcing the Courant–Friedrichs–Lewy condition in explicitly conservative local time stepping schemes. *Journal of Computational Physics*, 359:93–105, April 2018.

[94] J. L. Guermond, P. Minev, and J. Shen. Error Analysis of Pressure-Correction Schemes for the Time-Dependent Stokes Equations with Open Boundary Conditions. *SIAM Journal on Numerical Analysis*, 43(1):239–258, January 2005. Publisher: Society for Industrial and Applied Mathematics.

[95] J. L. Guermond, P. Minev, and Jie Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44):6011–6045, September 2006.

[96] J.-L. Guermond and L. Quartapelle. On the approximation of the unsteady Navier-Stokes equations by finite element projection methods. *Numerische Mathematik*, 80(2):207–238, August 1998.

[97] Abhinav Gupta, Patrick J. Haley, Deepak N. Subramani, and Pierre F. J. Lermusiaux. Fish modeling and Bayesian learning for the Lakshadweep Islands. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, Seattle, October 2019. IEEE.

[98] Abhinav Gupta and Pierre F. J. Lermusiaux. Bayesian learning of coupled biogeochemical-physical models. *Progress in Oceanography*, 216:103050, May 2023.

[99] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *CoRR*, abs/1801.01290, 2018. arXiv: 1801.01290.

[100] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[101] Carlos Haertel, Eckart Meiburg, and Frieder Necker. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 1. Flow topology and front speed for slip and no-slip boundaries. *Journal of Fluid Mechanics*, 418:189–212, 2000. Publisher: Cambridge University Press.

[102] Ernst Hairer, Syvert P Nørsett, and Gerhard Wanner. *Solving ordinary differential equations I.* Springer-Vlg, 1991.

[103] Ernst Hairer and Gerhard Wanner. *Solving ordinary differential equations II.* Springer Berlin Heidelberg, 1996.

[104] P. J. Haley, Jr., P. F. J. Lermusiaux, A. R. Robinson, W. G. Leslie, O. Logoutov, G. Cossarini, X. S. Liang, P. Moreno, S. R. Ramp, J. D. Doyle, J. Bellingham, F. Chavez, and S. Johnston. Forecasting and reanalysis in the Monterey Bay/California Current region for the Autonomous Ocean Sampling Network-II experiment. *Deep Sea Research Part II: Topical Studies in Oceanography*, 56(3–5):127–148, February 2009.

[105] Patrick J. Haley, Jr., Arpit Agarwal, and Pierre F. J. Lermusiaux. Optimizing velocities and transports for complex coastal regions and archipelagos. *Ocean Modelling*, 89:1–28, May 2015.

[106] Patrick J. Haley, Jr., Abhinav Gupta, Chris Mirabito, and Pierre F. J. Lermusiaux. Towards Bayesian ocean physical-biogeochemical-acidification prediction and learning systems for Massachusetts Bay. In *OCEANS 2020 IEEE/MTS*, pages 1–9. IEEE, October 2020.

[107] Patrick J. Haley, Jr. and Pierre F. J. Lermusiaux. Multiscale two-way embedding schemes for free-surface primitive equations in the "Multidisciplinary Simulation, Estimation and Assimilation System". *Ocean Dynamics*, 60(6):1497–1537, December 2010.

[108] Peter E. Hamlington, Luke P. Van Roekel, Baylor Fox-Kemper, Keith Julien, and Gregory P. Chini. Langmuir–Submesoscale Interactions: Descriptive Analysis of Multiscale Frontal Spindown Simulations. *Journal of Physical Oceanography*, 44(9):2249–2272, September 2014. Publisher: American Meteorological Society Section: Journal of Physical Oceanography.

[109] Conor F Hayes, Roxana Radulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, and others. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022. Publisher: Springer.

[110] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*, volume 54 of *Texts in Applied Mathematics*. Springer New York, New York, NY, 2008.

[111] Ronald HW Hoppe, Yuri Iliash, Chakradhar Iyyunni, and Nasser Hassan Sweilam. A posteriori error estimates for adaptive finite element discretizations of boundary control problems. -, 2006. Publisher: Walter de Gruyter Genthiner Strasse 13 10875 Berlin Germany.

[112] Ronald HW Hoppe, Guido Kanschat, and Tim Warburton. Convergence analysis of an adaptive interior penalty discontinuous Galerkin method. *SIAM journal on numerical analysis*, 47(1):534–550, 2009. Publisher: SIAM.

[113] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. Publisher: Elsevier.

[114] Roumen Iankov, Maria Datcheva, Sabina Cherneva, and D. Stoychev. *Finite Element Simulation of Nanoindentation Process.* January 2013.

[115] S. M. Jachec, O. B. Fringer, M. G. Gerritsen, and R. L. Street. Numerical simulation of internal tides and the resulting energetics within Monterey Bay and the surrounding area. *Geophysical Research Letters*, 33(12):L12605, 2006.

[116] T. M. S. Johnston, J. A. MacKinnon, P. L. Colin, P. J. Haley, Jr., P. F. J. Lermusiaux, A. J. Lucas, M. A. Merrifield, S. T. Merrifield, C. Mirabito, J. D. Nash, C. Y. Ou, M. Siegelman, E. J. Terrill, and A. F. Waterhouse. Energy and momentum lost to wake eddies and lee waves generated by the North Equatorial Current and tidal flows at Peleliu, Palau. *Oceanography*, 32(4):110–125, December 2019.

[117] Eric Jones, Travis Oliphant, Pearu Peterson, and others. SciPy: Open source scientific tools for Python, 2001.

[118] Amit Joshi, Alan B Thompson, Eva M Sevick-Muraca, and Wolfgang Bangerth. Adaptive finite element methods for forward modeling in fluorescence enhanced frequency domain optical tomography. In *Biomedical Topical Meeting*, page WB7. Optical Society of America, 2004.

[119] Jochen Kaempf. Wind-Driven Overturning, Mixing and Upwelling in Shallow Water: A Nonhydrostatic Modeling Study. *Journal of Marine Science and Engineering*, 5(4):47, October 2017.

[120] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018. Publisher: The Royal Society Publishing.

[121] Y. Kanarska, A. Shchepetkin, and J. C. McWilliams. Algorithm for non-hydrostatic dynamics in the Regional Oceanic Modeling System. *Ocean Modelling*, 18(3):143–174, January 2007.

[122] Donald W Kelly, JP De SR Gago, Olgierd C Zienkiewicz, and I Babuska. A posteriori error analysis and adaptive processes in the finite element method: Part I–error analysis. *International journal for numerical methods in engineering*, 19(11):1593–1619, 1983. Publisher: Wiley Online Library.

[123] Christopher A Kennedy and Mark H Carpenter. Diagonally implicit Runge–Kutta methods for ordinary differential equations. A review. 2016.

[124] Robert M Kirby, Spencer J Sherwin, and Bernardo Cockburn. To CG or to HDG: a comparative study. *Journal of Scientific Computing*, 51(1):183–212, 2012. Publisher: Springer.

[125] Richard I Klein. Star formation with 3-D adaptive mesh refinement: the collapse and fragmentation of molecular clouds. *Journal of Computational and Applied Mathematics*, 109(1-2):123–152, 1999. Publisher: Elsevier.

[126] Knut Klingbeil and Hans Burchard. Implementation of a direct nonhydrostatic pressure gradient discretisation into a layered ocean model. *Ocean Modelling*, 65:64–77, May 2013.

[127] Michal A Kopera and Francis X Giraldo. Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations. *Journal of Computational Physics*, 275:92–117, 2014. Publisher: Elsevier.

[128] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[129] Lilia Krivodonova and Joseph E Flaherty. Error estimation for discontinuous Galerkin solutions of two-dimensional hyperbolic problems. *Advances in Computational Mathematics*, 19(1):57–71, 2003. Publisher: Springer.

[130] Lilia Krivodonova, Jianguo Xin, J-F Remacle, Nicolas Chevaugeon, and Joseph E Flaherty. Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. *Applied Numerical Mathematics*, 48(3-4):323–338, 2004. Publisher: Elsevier.

[131] Martin Kronbichler and Katharina Kormann. A generic interface for parallel cell-based finite element operator application. *Computers & Fluids*, 63:135–147, 2012. Publisher: Elsevier.

[132] Martin Kronbichler, Katharina Kormann, and Wolfgang A Wall. Fast matrix-free evaluation of hybridizable discontinuous Galerkin operators. In *European Conference on Numerical Mathematics and Advanced Applications*, pages 581–589. Springer, 2017.

[133] Martin Kronbichler and Per-Olof Persson, editors. *Efficient High-Order Discretizations for Computational Fluid Dynamics*, volume 602 of *CISM International Centre for Mechanical Sciences*. Springer International Publishing, Cham, 2021.

[134] Martin Kronbichler and Wolfgang A Wall. A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SIAM Journal on Scientific Computing*, 40(5):A3423–A3448, 2018. Publisher: SIAM.

[135] C. S. Kulkarni, P. J. Haley, Jr., P. F. J. Lermusiaux, A. Dutt, A. Gupta, C. Mirabito, D. N. Subramani, S. Jana, W. H. Ali, T. Peacock, C. M. Royo, A. Rzeznik, and R. Supekar. Real-time sediment plume modeling in the Southern California Bight. In *OCEANS Conference 2018*, Charleston, SC, October 2018. IEEE.

[136] Chinmay S. Kulkarni and Pierre F. J. Lermusiaux. Advection without compounding errors through flow map composition. *Journal of Computational Physics*, 398:108859, December 2019.

[137] H. J. Kull. Theory of the Rayleigh-Taylor instability. *Physics Reports*, 206(5):197–325, 1991.

[138] Pijush K. Kundu, Ira M. Cohen, and David R. Dowling. *Fluid Mechanics*. Academic Press, June 2015. Google-Books-ID: EehDBAAAQBAJ.

[139] Dmitri Kuzmin. A new perspective on flux and slope limiting in discontinuous Galerkin methods for hyperbolic conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 373:113569, January 2021. arXiv:2008.11981 [cs, math].

[140] Zhigang Lai, Changsheng Chen, Geoffrey W. Cowles, and Robert C. Beardsley. A nonhydrostatic version of FVCOM: 1. Validation experiments. *Journal of Geophysical Research: Oceans*, 115(C11), 2010. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2009JC005525.

[141] Zhigang Lai, Changsheng Chen, Geoffrey W. Cowles, and Robert C. Beardsley. A nonhydrostatic version of FVCOM: 2. Mechanistic study of tidally generated nonlinear internal waves in Massachusetts Bay. *Journal of Geophysical Research: Oceans*, 115(C12), 2010. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2010JC006331.

[142] Frans-Peter A. Lam, Patrick J. Haley, Jr., Jeroen Janmaat, Pierre F. J. Lermusiaux, Wayne G. Leslie, Mathijs W. Schouten, Lianke A. te Raa, and Michel Rixen. At-sea real-time coupled four-dimensional oceanographic and acoustic forecasts during Battlespace Preparation 2007. *Journal of Marine Systems*, 78(Supplement):S306–S320, November 2009.

[143] P. F. J. Lermusiaux. Data assimilation via Error Subspace Statistical Estimation, part II: Mid-Atlantic Bight shelfbreak front simulations, and ESSE validation. *Monthly Weather Review*, 127(7):1408–1432, July 1999.

[144] P. F. J. Lermusiaux. Estimation and study of mesoscale variability in the Strait of Sicily. *Dynamics of Atmospheres and Oceans*, 29(2):255–303, 1999.

[145] P. F. J. Lermusiaux. Evolving the subspace of the three-dimensional multiscale ocean variability: Massachusetts Bay. *Journal of Marine Systems*, 29(1):385–422, 2001.

[146] P. F. J. Lermusiaux. Uncertainty estimation and prediction for interdisciplinary ocean dynamics. *Journal of Computational Physics*, 217(1):176–199, 2006.

[147] P. F. J Lermusiaux. Adaptive modeling, adaptive data assimilation and adaptive sampling. *Physica D: Nonlinear Phenomena*, 230(1):172–196, 2007.

[148] P. F. J. Lermusiaux, C.-S. Chiu, G. G. Gawarkiewicz, P. Abbot, A. R. Robinson, R. N. Miller, P. J. Haley, Jr, W. G. Leslie, S. J. Majumdar, A. Pang, and F. Lekien. Quantifying uncertainties in ocean predictions. *Oceanography*, 19(1):92–105, 2006.

[149] P. F. J. Lermusiaux, C.-S. Chiu, and A. R. Robinson. Modeling uncertainties in the prediction of the acoustic wavefield in a shelfbreak environment. In E.-C. Shang, Q. Li, and T. F. Gao, editors, *Proceedings of the 5th International Conference on Theoretical and Computational Acoustics*, pages 191–200. World Scientific Publishing Co., May 21-25 2002. Refereed invited manuscript.

[150] P. F. J. Lermusiaux, P. J. Haley, W. G. Leslie, A. Agarwal, O. Logutov, and L. J. Burton. Multiscale physical and biological dynamics in the Philippine Archipelago: Predictions and processes. *Oceanography*, 24(1):70–89, 2011. Special Issue on the Philippine Straits Dynamics Experiment.

[151] P. F. J. Lermusiaux, P. J. Haley, Jr., S. Jana, A. Gupta, C. S. Kulkarni, C. Mirabito, W. H. Ali, D. N. Subramani, A. Dutt, J. Lin, A. Shcherbina, C. Lee, and A. Gangopadhyay. Optimal planning and sampling predictions for autonomous and Lagrangian platforms and sensors in the northern Arabian Sea. *Oceanography*, 30(2):172–185, June 2017. Special issue on Autonomous and Lagrangian Platforms and Sensors (ALPS).

[152] P. F. J Lermusiaux, P. J. Haley, Jr, and N. K. Yilmaz. Environmental prediction, path planning and adaptive sampling: sensing and modeling for efficient ocean monitoring, management and pollution control. *Sea Technology*, 48(9):35–38, 2007.

[153] P. F. J. Lermusiaux, T. Lolla, P. J. Haley, Jr., K. Yigit, M. P. Ueckermann, T. Sondergaard, and W. G. Leslie. Science of autonomy: Time-optimal path planning and adaptive sampling for swarms of ocean vehicles. In Tom Curtin, editor, *Springer Handbook of Ocean Engineering: Autonomous Ocean Vehicles, Subsystems and Control*, chapter 21, pages 481–498. Springer, 2016.

[154] P. F. J Lermusiaux, A. J. Miller, and N. Pinardi. Special issue of Dynamics of Atmospheres and Oceans in honor of Prof. A. R. Robinson. *Dynamics of Atmospheres and Oceans*, 52(1–2):1–3, September 2011. Editorial.

[155] P. F. J. Lermusiaux, D. N. Subramani, J. Lin, C. S. Kulkarni, A. Gupta, A. Dutt, T. Lolla, P. J. Haley, Jr., W. H. Ali, C. Mirabito, and S. Jana. A future for intelligent autonomous ocean observing systems. *Journal of Marine Research*, 75(6):765–813, November 2017. The Sea. Volume 17, The Science of Ocean Prediction, Part 2.

[156] Pierre F. J. Lermusiaux. Numerical fluid mechanics. MIT OpenCourseWare, May 2015.

[157] Pierre F. J. Lermusiaux, Manan Doshi, Chinmay S. Kulkarni, Abhinav Gupta, Patrick J. Haley, Jr., Chris Mirabito, Francesco Trotta, S. J. Levang, G. R. Flierl, J. Marshall, Thomas Peacock, and C. Noble. Plastic pollution in the coastal oceans: Characterization and modeling. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, Seattle, October 2019. IEEE.

[158] Pierre F. J. Lermusiaux, Patrick J. Haley, Jr., Chris Mirabito, Wael H. Ali, Manmeet Bhabra, Phil Abbot, Ching-Sang Chiu, and Chris Emerson. Multi-resolution probabilistic ocean physics-acoustic modeling: Validation in the New Jersey continental shelf. In *OCEANS 2020 IEEE/MTS*, pages 1–9. IEEE, October 2020.

[159] Pierre F. J. Lermusiaux, P. Malanotte-Rizzoli, D. Stammer, J. Carton, J. Cummings, and A. M. Moore. Progress and prospects of U.S. data assimilation in ocean research. *Oceanography*, 19(1):172–183, 2006.

[160] Pierre F. J. Lermusiaux, Chris Mirabito, Patrick J. Haley, Jr., Wael Hajj Ali, Abhinav Gupta, Sudip Jana, Eugene Dorfman, Alison Laferriere, Aaron Kofford, G. Shepard, M. Goldsmith, Kevin Heaney, Emanuel Coelho, J. Boyle, J. Murray, L. Freitag, and A. Morozov. Real-time probabilistic coupled ocean physics-acoustics forecasting and data assimilation for underwater GPS. In *OCEANS 2020 IEEE/MTS*, pages 1–9. IEEE, October 2020.

[161] Pierre F. J. Lermusiaux, Jens Schröter, Sergey Danilov, Mohamed Iskandarani, Nadia Pinardi, and Joannes J. Westerink. Multiscale modeling of coastal, shelf and global ocean dynamics. *Ocean Dynamics*, 63(11-12):1341–1344, 2013.

[162] Pierre F. J. Lermusiaux, Jinshan Xu, Chi-Fang Chen, Sen Jan, L.Y. Chiu, and Yiing-Jang Yang. Coupled ocean–acoustic prediction of transmission loss in a continental shelfbreak region: Predictive skill, uncertainty quantification, and dynamical sensitivities. *IEEE Journal of Oceanic Engineering*, 35(4):895–916, October 2010.

[163] Pierre Lesaint and Pierre-Arnaud Raviart. On a finite element method for solving the neutron transport equation. *Publications mathèmatiques et informatique de Rennes*, (S4):1–40, 1974.

[164] W. G. Leslie, A. R. Robinson, P. J. Haley, Jr, O. Logutov, P. A. Moreno, P. F. J. Lermusiaux, and E. Coelho. Verification and training of real-time forecasting of multi-scale ocean dynamics for maritime rapid environmental assessment. *Journal of Marine Systems*, 69(1):3–16, 2008.

[165] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[166] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2015.

[167] Jing Lin. *Bayesian Learning for High-Dimensional Nonlinear Systems: Methodologies, Numerics and Applications to Fluid Flows.* PhD thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Massachusetts, September 2020.

[168] Jing Lin. Minimum-correction second-moment matching: Theory, algorithms and applications. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Massachusetts, February 2020.

[169] O. G. Logutov and P. F. J. Lermusiaux. Inverse barotropic tidal estimation for regional ocean applications. *Ocean Modelling*, 25(1–2):17–34, 2008.

[170] Peter Lu and Pierre F. J. Lermusiaux. Bayesian learning of stochastic dynamical models. *Physica D: Nonlinear Phenomena*, 427:133003, December 2021.

[171] A Mahadevan, A Tandon, and R Ferrari. Rapid changes in mixed layer stratification driven by submesoscale instabilities and winds. *Journal of Geophysical Research: Oceans*, 115(C3), 2010. Publisher: Wiley Online Library.

[172] Amala Mahadevan. Modeling vertical motion at ocean fronts: Are nonhydrostatic effects relevant at submesoscales? *Ocean Modelling*, 14(3-4):222–240, January 2006.

[173] Amala Mahadevan, Eric A D'Asaro, John T Allen, Pablo Almaraz García, Eva Alou-Font, Harilal Meenambika Aravind, Pau Balaguer, Isabel Caballero, Noemi Calafat, Andrea Carbornero, et al. Calypso 2019 cruise report: field campaign in the mediterranean. 2020.

[174] Amala Mahadevan, Ananda Pascual, Daniel L Rudnick, Simón Ruiz, Joaquín Tintoré, and Eric D'Asaro. Coherent pathways for vertical transport from the surface ocean to interior. *Bulletin of the American Meteorological Society*, 101(11):E1996–E2004, 2020.

[175] Jan Mandel. Efficient implementation of the ensemble Kalman filter.

[176] Jan Mandel. A brief tutorial on the ensemble Kalman filter. *arXiv preprint arXiv:0901.3725*, 2009.

[177] Emilie Marchandise, Jean-Francois Remacle, and Nicolas Chevaugeon. A quadrature-free discontinuous Galerkin method for the level set equation. *Journal of Computational Physics*, 212(1):338–357, February 2006.

[178] John Marshall, Chris Hill, Lev Perelman, and Alistair Adcroft. Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. *Journal of Geophysical Research: Oceans*, 102(C3):5733–5752, March 1997.

[179] Youssef M Marzouk and Habib N Najm. Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems. *Journal of Computational Physics*, 228(6):1862–1902, 2009. Publisher: Elsevier.

[180] James C McWilliams. Submesoscale currents in the ocean. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 472(2189):20160117, 2016.

[181] Chris Mirabito, Patrick J. Haley, Jr., Manan Doshi, Kyprianos Gkirgkis, Wael Hajj Ali, Chinmay Kulkarni, Abhinav Gupta, Pierre F. J. Lermusiaux, Amala Mahadevan, Mara Freilich, Mathieu Dever, Shaun Johnston, Tamay Özgökmen, Larry Pratt, Irina Rypina, Dan Rudnick, Andrey Shcherbina, Craig McNeil, Ruth Musgrave, Sutanu Sarkar, Luca Centurioni, Eric D'Asaro, James McWilliams, Peter Sullivan, Helga Huntley, Denny Kirwan, Tom Farrar, Annalisa Griffa, Uwe Send, Matthias Lankhorst, Michael Allshouse, Thomas Peacock,

Pierre-Marie Poulain, Simón Ruiz, Ananda Pascual, Joaquín Tintoré, John Allen, Baptiste Mourre, Eva Alou-Font, Nikolaos Zarakanellos, Cristian Muñoz, Inma Ruiz, and Alexandra Z. Worden. Real-time high-resolution probabilistic Eulerian and Lagrangian forecasting in the Alboran Sea: Skill, subduction, and multi-downscaling ensemble predictions. *Ocean Modelling*, 2023. In preparation.

[182] Volodymyr Mnih, Adriá Puigdoménech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv*, 2016.

[183] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, 2013.

[184] A. M. Moore, M. Martin, S. Akella, H. Arango, M. Balmaseda, L. Bertino, S. Ciavatta, B. Cornuelle, J. Cummings, S. Frolov, P. Lermusiaux, P. Oddo, P. R. Oke, A. Storto, A. Teruzzi, A. Vidard, and A. T. Weaver. Synthesis of ocean observations using data assimilation for operational, real-time and reanalysis systems: A more complete picture of the state of the ocean. *Frontiers in Marine Science*, 6(90):1–6, March 2019.

[185] Pedro Morin, Ricardo H Nochetto, and Kunibert G Siebert. Data oscillation and convergence of adaptive FEM. *SIAM Journal on Numerical Analysis*, 38(2):466–488, 2000. Publisher: SIAM.

[186] MSEAS Group. MSEAS Software, 2013.

[187] Carlos Muñoz Royo, Thomas Peacock, Matthew H. Alford, Jerome A. Smith, Arnaud Le Boyer, Chinmay S. Kulkarni, Pierre F. J. Lermusiaux, Patrick J. Haley, Jr., Chris Mirabito, Dayang Wang, E. Eric Adams, Raphael Ouillon, Alexander Breugem, Boudewijn Decrop, Thijs Lanckriet, Rohit B. Supekar, Andrew J. Rzeznik, Amy Gartman, and Se-Jong Ju. Extent of impact of deep-sea nodule mining midwater plumes is influenced by sediment loading, turbulence and thresholds. *Nature Communications Earth & Environment*, 2(148):1–16, July 2021.

[188] Fabio Naddei, Marta de la Llave Plata, Vincent Couaillier, and Frédéric Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods. *Journal of Computational Physics*, 376:508–533, 2019. Publisher: Elsevier.

[189] Ramachandran D Nair. Quadrature-Free Implementation of a Discontinuous Galerkin Global Shallow–Water Model via Flux Correction Procedure. *Monthly Weather Review*, 143(4):1335–1346, 2015.

[190] Christopher Nemeth and Paul Fearnhead. Stochastic gradient Markov chain Monte Carlo, July 2019. arXiv:1907.06986 [stat].

[191] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for linear convection–diffusion equations. *Journal of Computational Physics*, 228(9):3232–3254, May 2009.

[192] N. C. Nguyen, J. Peraire, and B. Cockburn. A hybridizable discontinuous Galerkin method for Stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 199(9):582–597, January 2010.

[193] N. C. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 230(4):1147–1170, February 2011.

[194] N.C. Nguyen and J. Peraire. Hybridizable discontinuous Galerkin methods for partial differential equations in continuum mechanics. *Journal of Computational Physics*, 231(18):5955–5988, July 2012.

[195] N.C. Nguyen, J. Peraire, and B. Cockburn. High-order implicit hybridizable discontinuous Galerkin methods for acoustics and elastodynamics. *Journal of Computational Physics*, 230(10):3695–3718, May 2011.

[196] Ngoc Cuong Nguyen and Jaume Peraire. Lecture Notes for MIT course 16.930, 2017.

[197] Ngoc Cuong Nguyen, Jaume Peraire, and Bernardo Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for nonlinear convection–diffusion equations. *Journal of Computational Physics*, 228(23):8841–8855, 2009. Publisher: Elsevier.

[198] Tien-Tai Nguyen. Nonlinear Rayleigh-Taylor instability of the viscous surface wave in an infinitely deep ocean, May 2023. arXiv:2211.14888 [math].

[199] Reiner Onken, Alberto Álvarez, Vicente Fernández, Guillermo Vizoso, Gotzon Basterretxea, Joaquín Tintoré, Patrick Haley Jr, and Elvio Nacini. A forecast experiment in the Balearic Sea. *Journal of Marine Systems*, 71(1-2):79–98, 2008.

[200] Reiner Onken, Allan R. Robinson, Pierre F. J. Lermusiaux, Patrick J. Haley, and Larry A. Anderson. Data-driven simulations of synoptic circulation and transports in the Tunisia-Sardinia-Sicily region. *Journal of Geophysical Research: Oceans*, 108(C9), 2003.

[201] Jie Pan, Jingwei Huang, Gengdong Cheng, and Yong Zeng. Reinforcement learning for automatic quadrilateral mesh generation: a soft actor-critic approach. *arXiv preprint arXiv:2203.11203*, 2022.

[202] Wei Pan, Stephan C. Kramer, Tuomas Kärnä, and Matthew D. Piggott. Comparing non-hydrostatic extensions to a discontinuous finite element coastal ocean model. *Ocean Modelling*, 151:101634, July 2020.

[203] Wei Pan, Stephan C. Kramer, and Matthew D. Piggott. Multi-layer non-hydrostatic free surface modelling using the discontinuous Galerkin method. *Ocean Modelling*, 134:68–83, February 2019.

[204] Wei Pan, Stephan C. Kramer, and Matthew D. Piggott. A sigma-coordinate non-hydrostatic discontinuous finite element coastal ocean model. *Ocean Modelling*, 157:101732, January 2021.

[205] Ivo Pasmans, Yumeng Chen, Alberto Carrassi, and Chris K. R. T. Jones. Tailoring data assimilation to discontinuous Galerkin models, May 2023. arXiv:2305.02950 [physics].

[206] Will Pazner and Per-Olof Persson. Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations. *Journal of Computational Physics*, 335:700–717, 2017. Publisher: Elsevier.

[207] Jaime Peraire, Ngoc Nguyen, and Bernardo Cockburn. A hybridizable discontinuous Galerkin method for the compressible Euler and Navier-Stokes equations. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 363, 2010.

[208] Nadia Pinardi, L. Cavaleri, G. Coppini, P. De Mey, C. Fratianni, J. Huthnance, P. F. J. Lermusiaux, A. Navarra, R. Preller, and S. Tibaldi. From weather to ocean predictions: an historical viewpoint. *Journal of Marine Research*, 75(3):103–159, May 2017. Special issue: The Science of Ocean Prediction, vol. 17 of The Sea.

[209] Nadia Pinardi, P. F. J. Lermusiaux, K. H. Brink, and R. Preller. The sea: The science of ocean prediction. *Journal of Marine Research*, 75(3):101–102, May 2017. Special issue: The Science of Ocean Prediction, vol. 17 of The Sea.

[210] Tomasz Plewa, Timur Linde, V Gregory Weirs, and others. *Adaptive mesh refinement-theory and applications*. Springer, 2005.

[211] Andrei D Polyanin and Vladimir E Nazaikinskii. *Handbook of linear partial differential equations for engineers and scientists*. CRC press, 2015.

[212] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable Baselines3, 2019.

[213] Ellery Rajagopal, Anantha N. S. Babu, Tony Ryu, Patrick J. Haley, Jr., Chris Mirabito, and Pierre F. J. Lermusiaux. Evaluation of deep neural operator models toward ocean forecasting. In *OCEANS 2023 IEEE/MTS Gulf Coast*, Biloxi, MS, September 2023. IEEE. In press.

[214] Ali Ramadhan, Gregory Wagner, Chris Hill, Jean-Michel Campin, Valentin Churavy, Tim Besard, Andre Souza, Alan Edelman, Raffaele Ferrari, and John Marshall. Oceananigans. jl: Fast and friendly geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2020.

[215] Steven R. Ramp, Pierre F. J. Lermusiaux, Igor Shulman, Yi Chao, Rebecca E. Wolf, and Frederick L. Bahr. Oceanographic and atmospheric conditions on the continental shelf north of the Monterey Bay during August 2006. *Dynamics of Atmospheres and Oceans*, 52(1–2):192–223, September 2011. Special issue of Dynamics of Atmospheres and Oceans in honor of Prof. A. R. Robinson.

[216] Karl Johan Åström. Optimal control of Markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965. Publisher: Academic Press.

[217] J. Ray, S. A. McKenna, B. van Bloemen Waanders, and Y. M. Marzouk. Bayesian reconstruction of binary media with unresolved fine-scale spatial structures. *Advances in Water Resources*, 44:1–19, August 2012.

[218] William H Reed and TR Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[219] Bruce A. Remington, R. Paul Drake, Hideaki Takabe, and David Arnett. A review of astrophysics experiments on intense lasers. *Physics of Plasmas*, 7:1641–1652, 2000.

[220] Beatrice Riviere. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[221] A. R. Robinson and P. F. J. Lermusiaux. Data assimilation (physical/interdisciplinary). In J. H. Steele, S. Thorpe, and K. Turekian, editors, *Encyclopedia of Ocean Sciences*, pages 302–308. Academic Press, London, 2001.

[222] A. R. Robinson, P. F. J. Lermusiaux, and N. Q. Sloan III. Data assimilation. In Kenneth H. Brink and Allan R. Robinson, editors, *The Global Coastal Ocean-Processes and Methods*, volume 10 of *The Sea*, chapter 20, pages 541–594. John Wiley and Sons, New York, 1998.

[223] Allan R. Robinson and Pierre F. J. Lermusiaux. Data assimilation for modeling and predicting coupled physical–biological interactions in the sea. In Allan R. Robinson, James J. McCarthy, and Brian J. Rothschild, editors, *Biological-Physical Interactions in the Sea*, volume 12 of *The Sea*, chapter 12, pages 475–536. John Wiley and Sons, New York, 2002.

[224] Xevi Roca, Cuong Nguyen, and Jaime Peraire. Scalable parallelization of the hybridized discontinuous Galerkin method for compressible flow. In *21st AIAA Computational Fluid Dynamics Conference*, page 2939, 2013.

[225] Xevi Roca, Ngoc Cuong Nguyen, and Jaime Peraire. GPU-accelerated sparse matrix-vector product for a hybridizable discontinuous Galerkin method. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 687, 2011.

[226] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, April 2016. Publisher: PeerJ.

[227] R. L. Sani and P. M. Gresho. Résumé and remarks on the open boundary condition minisymposium. *International Journal for Numerical Methods in Fluids*, 18(10):983–1008, 1994.

[228] Aditya Karthik Saravanakumar. Towards coupled nonhydrostatic-hydrostatic hybridizable discontinuous Galerkin method. Master's thesis, Massachusetts Institute of Technology, Computational Science and Engineering, Cambridge, Massachusetts, June 2023.

[229] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[230] Nico Schlömer. quadpy: Numerical integration (quadrature, cubature) in Python (2018).

[231] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

[232] David Silver. Lectures on Reinforcement Learning, 2015. Published: URL: https://www.davidsilver.uk/teaching/.

[233] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017. Publisher: Elsevier.

[234] Walter A Strauss. *Partial differential equations: An introduction.* John Wiley & Sons, 2007.

[235] D. N. Subramani, P. F. J. Lermusiaux, P. J. Haley, Jr., C. Mirabito, S. Jana, C. S. Kulkarni, A. Girard, D. Wickman, J. Edwards, and J. Smith. Time-optimal path planning: Real-time sea exercises. In *Oceans '17 MTS/IEEE Conference*, Aberdeen, June 2017.

[236] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[237] Nobuhiro Suzuki, Baylor Fox-Kemper, Peter E. Hamlington, and Luke P. Van Roekel. Surface waves affect frontogenesis. *Journal of Geophysical Research: Oceans*, 121(5):3597–3624, 2016. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/2015JC011563.

[238] Hitoshi Tamura, Peter Bacopoulos, Dingbao Wang, Scott C. Hagen, and Ethan J. Kubatko. State estimation of tidal hydrodynamics using ensemble Kalman filter. *Advances in Water Resources*, 63:45–56, January 2014.

[239] Qiang Tang, Xiaomeng Huang, Lei Lin, Wei Xiong, Dong Wang, Mingqing Wang, and Xing Huang. MERF v3.0, a highly computationally efficient non-hydrostatic ocean model with implicit parallelism: Algorithms and validation experiments. *Ocean Modelling*, 167:101877, November 2021.

[240] John R Taylor and Andrew F Thompson. Submesoscale dynamics in the upper ocean. *Annual Review of Fluid Mechanics*, 55:103–127, 2023.

[241] Ioannis Toulopoulos and John A Ekaterinaris. High-order discontinuous Galerkin discretizations for computational aeroacoustics in complex domains. *AIAA journal*, 44(3):502–511, 2006.

[242] M. P. Ueckermann. Towards next generation ocean models: Novel discontinuous Galerkin schemes for 2D unsteady biogeochemical models. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, MA, September 2009.

[243] M. P. Ueckermann. *High Order Hybrid Discontinuous Galerkin Regional Ocean Modeling.* PhD thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, MA, February 2014.

[244] M. P. Ueckermann and P. F. J. Lermusiaux. High order schemes for 2D unsteady biogeochemical ocean models. *Ocean Dynamics*, 60(6):1415–1445, December 2010.

[245] M. P. Ueckermann and P. F. J. Lermusiaux. 2.29 Finite Volume MATLAB Framework Documentation. MSEAS Report 14, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 2012.

[246] M. P. Ueckermann and P. F. J. Lermusiaux. Hybridizable discontinuous Galerkin projection methods for Navier–Stokes and Boussinesq equations. *Journal of Computational Physics*, 306:390–421, 2016.

[247] Mattheus P. Ueckermann, Chris Mirabito, Patrick J. Haley, Jr., and Pierre F. J. Lermusiaux. High order hybridizable discontinuous Galerkin projection schemes for non-hydrostatic physical-biogeochemical ocean modeling. *Ocean Dynamics*, 2023. In preparation.

[248] Hans van Haren. Instability observations associated with wave breaking in the stable-stratified deep-ocean. *Physica D: Nonlinear Phenomena*, 292-293:62–69, 2015.

[249] Vicky Verma, Hieu T. Pham, and Sutanu Sarkar. The submesoscale, the finescale and their interaction at a mixed layer front. *Ocean Modelling*, 140:101400, August 2019.

[250] Antoni Vidal-Ferràndiz, Amanda Carreno, Damián Ginestar Peiro, and Gumersindo Jesús Verdú Martín. hp-Finite element method for the simplified Pn equations. In *Congreso de Métodos Numéricos en Ingeniería (CMN 2017). Actas*, pages 208–220. International Center for Numerical Methods in Engineering (CIMNE), 2017.

[251] Sean Vitousek and Oliver B. Fringer. Physical vs. numerical dispersion in nonhydrostatic ocean modeling. *Ocean Modelling*, 40(1):72–86, January 2011.

[252] Sean Vitousek and Oliver B. Fringer. A nonhydrostatic, isopycnal-coordinate ocean model for internal waves. *Ocean Modelling*, 83:118–144, November 2014.

[253] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, April 2003. Google-Books-ID: wE0NrHkrqRAC.

[254] Jingbo Wang and Nicholas Zabaras. A Bayesian inference approach to the inverse heat conduction problem. *International Journal of Heat and Mass Transfer*, 47(17):3927–3941, August 2004.

[255] Yufei Wang, Ziju Shen, Zichao Long, and Bin Dong. Learning to discretize: solving 1D scalar conservation laws via deep reinforcement learning. *arXiv preprint arXiv:1905.11079*, 2019.

[256] Shiyin Wei, Xiaowei Jin, and Hui Li. General solutions for nonlinear differential equations: a rule-based self-learning approach using deep reinforcement learning. *Computational Mechanics*, 64(5):1361–1374, 2019. Publisher: Springer.

[257] Joannes J Westerink, Richard A Luettich, Jesse C Feyen, John H Atkinson, Clint Dawson, Hugh J Roberts, Mark D Powell, Jason P Dunion, Ethan J Kubatko, and Hasan Pourtaheri. A basin-to channel-scale unstructured grid hurricane storm surge model applied to southern Louisiana. *Monthly weather review*, 136(3):833–864, 2008.

[258] Yulong Xing, Xiangxiong Zhang, and Chi-Wang Shu. Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations. *Advances in Water Resources*, 33(12):1476–1493, 2010. Publisher: Elsevier.

[259] J. Xu, P. F. J. Lermusiaux, P. J. Haley Jr., W. G. Leslie, and O. G. Logutov. Spatial and Temporal Variations in Acoustic propagation during the PLUSNet-07 Exercise in Dabob Bay. In *Proceedings of Meetings on Acoustics (POMA)*, volume 4, page 11. Acoustical Society of America 155th Meeting, 2008.

[260] S. Xu, X. Huang, L.-Y. Oey, F. Xu, H. Fu, Y. Zhang, and G. Yang. POM.gpu-v1.0: a GPU-based Princeton Ocean Model. *Geoscientific Model Development*, 8(9):2815–2827, September 2015. Publisher: Copernicus GmbH.

[261] Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, Robert Anderson, and Daniel Faissol. Reinforcement Learning for Adaptive Mesh Refinement. *arXiv:2103.01342 [cs, math]*, March 2021. arXiv: 2103.01342.

[262] Tony D Young and Rickard Armiento. Strategies for h-adaptive refinement for a finite element treatment of harmonic oscillator Schrödinger eigenproblem. *Communications in Theoretical Physics*, 53(6):1017, 2010. Publisher: IOP Publishing.

[263] David L. Youngs. Numerical simulation of turbulent mixing by Rayleigh-Taylor instability. *Physica D: Nonlinear Phenomena*, 12(1):32–44, 1984.

[264] Mengping Zhang and Chi-Wang Shu. An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations. *Mathematical Models and Methods in Applied Sciences*, 13(03):395–413, 2003. Publisher: World Scientific.

[265] Ye Zhou. Rayleigh–Taylor and Richtmyer–Meshkov instability induced flow, turbulence, and mixing. I. *Physics Reports*, 720-722:1–136, 2017.