THE IMPERIAL COLLEGE OF SCIENCE,
TECHNOLOGY AND MEDICINE

DOCTORAL THESIS

Hybridizable compatible finite element discretizations for numerical weather prediction: implementation and analysis

---

Author:
Thomas H. GIBSON

Supervisors:
David A. HAM
Colin J. COTTER

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Applied Mathematics and Mathematical Physics Section
Department of Mathematics

Declaration of Authorship

I declare that this dissertation titled, "Hybridizable compatible finite element discretizations for numerical weather prediction: implementation and analysis" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at Imperial College London.
- Where I have consulted the published work of others, this is always clearly referenced.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- Any images or figures obtained from outside sources are clearly referenced, along with the access dates. Any tables, figures, or diagrams without such a reference are produced entirely by myself.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Thomas H. GIBSON

THE IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

Abstract

Faculty of Natural Sciences

Department of Mathematics

Doctor of Philosophy

Hybridizable compatible finite element discretizations for numerical weather prediction: implementation and analysis

by Thomas H. GIBSON

There is a current explosion of interest in new numerical methods for atmospheric modeling. A driving force behind this is the need to be able to simulate, with high efficiency, large-scale geophysical flows on increasingly more parallel computer systems. Many current operational models, including that of the UK Met Office, depend on orthogonal meshes, such as the latitude–longitude grid. This facilitates the development of finite difference discretizations with favorable numerical properties. However, such methods suffer from the “pole problem," which prohibits the model to make efficient use of a large number of computing processors due to excessive concentration of grid-points at the poles.

Recently developed finite element discretizations, known as “compatible" finite elements, avoid this issue while maintaining the key numerical properties essential for accurate geophysical simulations. Moreover, these properties can be obtained on arbitrary, non-orthogonal meshes. However, the efficient solution of the resulting discrete systems depend on transforming the mixed velocity-pressure (or velocity-pressure-buoyancy) system into an elliptic problem for the pressure. This is not so straightforward within the compatible finite element framework due to inter-element coupling.

This thesis supports the proposition that systems arising from compatible finite element discretizations can be solved efficiently using a technique known as “hybridization." Hybridization removes inter-element coupling while maintaining the desired numerical properties. This permits the construction of sparse, elliptic problems, for which fast solver algorithms are known, using localized algebra. We first introduce the technique for compatible finite element discretizations of simplified atmospheric models. We then develop a general software abstraction for the rapid implementation and composition of hybridization methods, with an emphasis on preconditioning. Finally, we extend the technique for a new compatible method for the full, compressible atmospheric equations used in operational models.

Acknowledgements

Before arriving at Imperial College London, I had the pleasure of meeting David Ham at my undergraduate institution, Baylor University, in Waco, Texas. The meeting was spontaneously arranged by one of my mentors, Dr. Robert C. Kirby. Both David and I spoke in great detail about PhD opportunities in Europe, and I was made aware of a new PhD program at Imperial College London supported by The Engineering and Physical Sciences Research Council (EPSRC): The Centre for Doctoral Training in the Mathematics of Planet Earth (MPECDT). Having been interested in weather and mathematics at a young age, I eagerly applied specifically to work with David Ham.

Through David, I met Colin Cotter, and both agreed to take me on as a PhD student. I am deeply grateful for their support throughout my studies. I feel that I have grown as an individual and scientist due to their guidance. When self-doubt was an all time high, their patience and enthusiasm for the work we do was incredibly motivating. They are both a constant source of inspiration; for every new suggestion, a plethora of new ideas emerge which unfortunately could not fit within the time-frame of my PhD. However, they both have inspired my own research agenda. I hope to have many opportunities for collaboration in the future.

While technically a student in the Mathematics Department, I spent a significant portion of my PhD interacting with members from the Department of Computing. I wish to thank my colleagues and friends from this time: Dr. Lawrence Mitchell (now an Associate Professor at Durham University), Dr. Miklos Homolya (now in seminary school), and Dr. Fabio Luporini (now a Research Associate at Imperial College London in Earth Sciences). Each of them helped me develop as a rigorous programmer and accelerated my understanding of software development.

I wish to also express my gratitude to fellow cohort members in the MPECDT and PhD students at Imperial: Thomas Bendall, Goodwin “Goody" Gibbins, Matthew Garrod, Tasmin Symons, Jemima Tabeart, Michael Haigh, Josephine “Josie" Park, Darije Čustović, and Andreas Bock. We have endured the stresses of research together, and have developed a strong sense of camaraderie.

Lastly, I want to thank my family and fiancée, Fátima Millán Cañas, for their support from the very beginning. To my father (Dr. Thomas Michael Gibson), mother (Melody Jane Gibson), sister (Janel Joy Gibson), and Fátima: I am grateful and dedicate this work you all.

My funding was provided by The EPSRC Centre for Doctoral Training in the Mathematics of Planet Earth (http://mpecdt.org/). Thank you, Prof. Dan Crisan, for pushing to secure funding for me as an overseas student.

Copyright Declaration

"I seek strength, not to be greater than my brother, but to fight my greatest enemy: Myself."

A Sioux prayer: "The Great Spirit Prayer."

Contents

List of Figures

List of Tables

1 Introduction

Computational science and engineering has emerged as a powerful tool for scientific inquiry, industrial manufacturing, and predictive technologies. Often referred to as "the third pillar" of the scientific method, the use of computer simulations of complex physical systems has accelerated our understanding of natural phenomena which would otherwise be impossible to directly reproduce in a laboratory setting. Such examples include large-scale geophysical flows (atmosphere and ocean circulation) and the rheology of the Earth's interior (mantel convection). A driving force behind the advancement in both computational power and robust numerical methods is the need to efficiently simulate systems which contain a vast range of dynamics, from the smallest scales (atomic and molecular) to the continuum level (solid-bodies and fluids). What many of these physical systems share is their representation as a system of tightly-coupled partial differential equations, derived from constitutive relations and physical laws (conservation of mass, momentum, and energy).

For weather and climate in particular, the benefits of improving our existing computational models translates across many facets of scientific research, governmental policy development, and the private sector. Having more accurate forecasts and climate projections in a timely-manner leads to a better understanding of the future-state of our own planet; leaders in government can make informed policy decisions driven by scientific-understanding; and business users from both insurers and the energy-sector can quantify risks associated with natural disasters driven by weather and climate.

Interest in simulating large-scale atmospheric dynamics on next-generation supercomputers has galvanized a massive effort around the globe to design new numerical methods capable of producing more accurate forecasts under strict operational time constraints. More accurate simulations (meaning more grid-points or degrees of freedom) directly implies the use of more computational power in order to meet scientific and industrial demands. For high-resolution (high-detail) models to be run in the same wall-clock time (execution time on a computer) or less, the numerics must be able to make efficient use of increasingly more parallel supercomputers. This is due to the fact that recent trends in modern super-computing systems opt for an increase in on-chip parallelism rather than processor speed. In other words, processing power is obtained by increasing parallel efficiency.

The previous generation of numerics depends on structured, orthogonal meshes (latitude-longitude) which suffer from the "pole problem": as model resolution is increased the resolution converges to a singularity at one or more special points on the grid. This leads to a breakdown of parallel scalability as a result of increased processor communication at the poles. The current dynamical core of the UK Met Office, ENDGame (Wood et al., 2014; Thuburn, 2016), has managed to extend its scalability on latitude-longitude grids by changing how variables are stored at the poles. However, this is not sustainable in the long-term. Recognizing this, the Met

Office/NERC/STFC UK dynamical core project, codenamed "Gung-Ho," was established. The project was tasked with finding suitable numerical discretizations that can be used on psuedo-uniform grids (avoiding the pole problem entirely) while maintaining the desired numerical properties of ENDGame that made it one of the most accurate numerical models in the world.

This effort has resulted in adopting the framework of "compatible" finite element methods (Cotter and Shipton, 2012; Cotter and Thuburn, 2014; Natale, Shipton, and Cotter, 2016; Shipton, Gibson, and Cotter, 2018). These special variations of traditional mixed finite element methods are grounded in a deep and rigorous mathematical framework which generalizes the numerics of ENDGame to more general, psuedo-uniform computational grids. The resulting new dynamical core in development, known as "LFRic" (in honor of Lewis Fry Richardson), is the culmination of extensive research through the Gung-Ho project and the application of compatible finite elements to large-scale geophysical systems (Adams et al., 2019). With this in mind, compatible finite element shows great promise.

Strong scalability (decrease in execution time per time-step as the number of processors increases for a fixed-size problem) is promising. However, the overall time-to-solution needs to be reduced dramatically to meet the operational constraints. At each time-step, a mixed saddle-point system coupling the prognostic variables (wind speed, density, temperature, and pressure) must be solved. For staggered finite difference methods employed by the UK Met Office model ENDGame, the saddle-point system is avoided by performing point-wise eliminations to arrive at an elliptic pressure-correction equation for which efficient methods are known (Müller and Scheichl, 2014). This is a challenge within finite element discretizations, as the saddle-point system cannot be decoupled in this manner and requires computationally expensive numerical algorithms to solve. This ultimately forms the computational bottleneck of the new dynamical core. Despite the elegant mathematics of compatible finite elements, the search for better solution algorithms for the complex discrete systems arising from such methods is far from over.

Fortunately, the problem of solving discrete systems arising from complex physical processes is nothing new. Engineers have been working on this problem for decades, long before finite element methods became popular for computational fluid dynamics. During the inception of the finite element method, previously known as the "framework method," (Hrennikoff, 1941) it was a numerical process for relating stress and strain from constitutive laws in continuum mechanics and structural engineering. The realization made by Courant (1943) brought the modern notion of a "mesh": a collection or cells or "elements" in which PDEs can be discretized over.

Throughout the 1950s and 1960s, the modern interpretation of the "finite element method" was realized due to the contributions of Argyris (1955), Clough (1960), and Zienkiewicz and Cheung (1965). As computers advanced, interest in more efficient computational methods became more of a priority. This lead to the co-developments of Guyan (1965) and Irons (1965), where they realized that certain discrete systems arising from finite element discretizations can be algebraically manipulated to produce condensed systems. This procedure is now known as *static condensation*: the process of eliminating unknowns to produce a smaller global linear system.

De Veubeke (1965) proposed the first application of static condensation to a mixed system derived from an elasticity model. Due to the strong coupling between unknowns, he introduced Lagrange multipliers on cell interfaces which imposed the

coupling conditions independently. This de-coupled the original unknowns, allowing for the static condensation of all but the variables lying on cell boundaries. Considered an "implementation trick" at the time, the method was rediscovered by Arnold and Brezzi (1985) for the mixed formulation of elliptic equations. They showed that the solutions of the "hybrid" formulation with the Lagrange multipliers *coincides* with the solutions of the original mixed problem. This reformulation became what is known today as *hybridization*. Rigorized further by Cockburn and Gopalakrishnan (2004) and Cockburn, Gopalakrishnan, and Lazarov (2009), the hybridization technique can permit the transformation of indefinite saddle-point systems into condensed elliptic equations on cell boundaries. The eliminated unknowns can then be recovered by solving small linear systems in each mesh cell.

In this dissertation, we investigate the application of hybridization techniques to the compatible finite element discretizations of geophysical models. While we apply many of the same arguments and methods to derive hybridizable formulations, we explore the use of hybridization as a method of preconditioning mixed equations, an alternative to standard solution methods for saddle-point systems like the Schur-complement approach (Benzi, Golub, and Liesen, 2005). Ultimately, we aim to develop a new hybridization method for the systems which appear in semi-implicit linearizations of the compressible atmospheric equations.

1.1 Thesis statement

The hybridization of compatible finite element discretizations provides a promising mechanism for the efficient solution of the saddle-point systems arising from atmospheric models.

1.2 Technical contributions

The novel technical contributions presented in this dissertation includes:

- Analysis of the discrete system arising from the hybridization of a simplified geophysical model: the linear rotating shallow water system. A characterization theorem for the solution of the hybridizable mixed finite element method is presented which reveals important properties of the discrete operator arising from static condensation. The analysis mirrors standard arguments from Cockburn and Gopalakrishnan (2004) and Cockburn, Gopalakrishnan, and Lazarov (2009), which has previously been applied to simplified elliptic systems.
- Two new hybridizable methods: one for the discretization of the hydrostatic pressure equation, and another for a linear compressible model.
- A domain-specific language, known as "Slate," for the symbolic representation of dense linear algebra on the element-wise systems appearing from finite element discretizations.
- A domain-specific compiler for the Slate language which translate the symbolic linear algebra operations into compiled C++ code.
- Implementations of preconditioners utilizing the Python-interface to PETSc (Balay et al., 1997): petsc4py (Dalcin et al., 2011). These preconditioners use

the Slate language and its compiler to automate the hybridization and static condensation of suitable finite element discretizations.

- Lastly, a new hybridization technique is presented for the linearized systems appearing in semi-implicit discretizations of the compressible atmospheric using compatible finite elements.

1.3 Dissertation outline

This dissertation is structured as follows.

- Chapter 2 provides a range of background material for the topics covered in subsequent chapters. This includes the following:
  - An overview of geophysical fluid dynamics relevant for the atmospheric dynamical core research. This will be presented as a hierarchy of models of varying complexity, starting from the full compressible atmosphere model (rotating Euler equations) down to a simplified two-dimensional shallow water system.
  - A summary of desirable numerical properties for atmospheric dynamical cores is presented to further motivate the use of "compatible" methods.
  - Background on the finite element method. Specifically, various element families are defined which are particularly relevant for the compatible finite element framework.
  - We end by summarizing compatible finite element discretizations for two- and three-dimensional systems. Moreoever, we discuss the construction of tensor-product finite element systems, which exploits the structure of standard atmospheric meshes.
- In Chapter 3, we introduce the hybridization technique for simplified atmospheric models, starting with the shallow water system. Analysis is provided for the hybridizable formulation which characterizes both the derivation and construction of the "hybridized" system. We also introduce the reader to a quasi-Newton/Picard method for solving the nonlinear shallow water system and how hybridization fits within that. We then apply the method to standard test cases on the sphere. The chapter concludes by providing two new hybridizable formulations of the hydrostatic pressure equation and a simplified three-dimensional gravity wave system.
- Chapter 4 centers around the Firedrake Project (www.firedrakeproject.org), an automatic code-generation library for finite element methods. We present an original contribution in the form of software, which enables the rapid implementation of hybridization and static condensation solvers. This is accomplished through the use of domain-specific abstractions and compilers for automatic code-generation. We further argue that the hybridization of mixed methods (and static condensation in general) can be interpreted as *preconditioners* for solving certain linear systems. Preconditioner interfaces composing with the PETSc solver library (Balay et al., 2016) are detailed as well. Numerical examples are provided demonstrating composability, correctness, and performance.

- In Chapter 5, we apply the methods of previous chapters to derive a new hybridization method for the quasi-Newton/Picard linearization of the Euler equations. We introduce Gusto (www.firedrakeproject.org/gusto/), a compatible finite element dynamical core built on top of Firedrake, and summarize how the hybridization method is used within a finite element dynamical core. Standard test cases for atmospheric dynamical cores are presented for model verification. We then conclude with remarks on iterative solvers for the resulting multiplier system and provide a three-dimensional test.
- Chapter 6 will summarize our results and discuss open questions and avenues for future research.

1.4 Dissemination

The work that will be presented in this dissertation has already been disseminated through journal publications, preprints, conference presentations, seminar talks, external collaboration (UK MET Office), a monograph with Springer-Verlag, and as an implementation in open-source software through the Firedrake Project (Rathgeber et al., 2017). Here, we summarize the various works that contribute to content of this dissertation.

Two scientific papers (one published and one in-review) contribute to core content in the dissertation:

- Shipton, Gibson, Cotter (2018), *Higher-order compatible finite element schemes for the nonlinear rotating shallow water equations on the sphere* (Shipton, Gibson, and Cotter, 2018).
- Gibson, Mitchell, Ham, Cotter (2019), *Slate: extending Firedrake's domain-specific abstraction to hybridized solvers for geoscience and beyond* (Gibson et al., 2019b).

The monograph, accepted for publication in the 2019 Springerbrief series in the Mathematics of Planet Earth, is the culmination of several contributions based on the research conducted in this dissertation:

- Gibson, McRae, Cotter, Mitchell, Ham (2019), *Compatible finite element methods for geophysical flows: automation and implementation using Firedrake* (Gibson et al., 2019a).

This Springerbrief is a pedagogical overview of simulating geophysical flows. The book introduces concepts and key results for building compatible finite element discretizations of geophysical flow equations. An extensive portion of the book is designed to demonstrate how to use Firedrake to design, build, and execute geophysical models. Discussions on building efficient solvers and utilizing hybridization is also included. Therefore, some of its content will provide some background and examples for this dissertation.

2 Preliminaries

2.1 Hierarchy of geophysical models

The first computer weather forecast on the ENIAC (one of the world's first digital computers) by Charney, von Neumann and co-workers solved a single-layer quasi-geostrophic model (Lynch, 2008).[1] Subsequently, as computer power increased, it became possible to move up the hierarchy, making fewer model approximations in more numerically intensive calculations. Numerical weather predictions were made with multilayer shallow water models, hydrostatic compressible Euler models, and finally the non-hydrostatic compressible Euler models which represent the state of the art today (Kalnay, 2002; Bauer, Thorpe, and Brunet, 2015; Staniforth and Wood, 2008). Model hierarchies are also very useful in the development of numerical models. One can move down the hierarchy to isolate specific aspects of the model to examine their numerical treatment in a less computationally intensive setting, and then move back up to use this insight. In the development of atmospheric dynamical cores, it is very standard to start with the shallow water equations to focus on horizontal discretisation aspects before moving up the hierarchy.

The hierarchy of models we will build up in this chapter is shown in Figure 2.1. This is just one choice of hierarchy, since for example one can apply the hydrostatic approximation directly to the compressible Euler equations without making Boussinesq/anelastic approximations. Similarly, we do not consider quasi-geostrophic models which are valid for rapidly rotating systems; these approximations can be made at any step of our hierarchy.

2.1.1 The compressible Euler system

We start by presenting the compressible Euler equations, which have the fewest approximations within our hierarchy. Before we present the full equations of motion, we introduce the notion of a *material derivative*. That is, the time rate of change of some field following the motion of a fluid.

Given some velocity field $\boldsymbol{u}$, we denote the material derivative by $\frac{d}{dt}$. For a field F, $\frac{dF}{dt}$ is given by the instantaneous time rate of change of F, plus a contribution from the spatial variation of F as a result of being moved by $\boldsymbol{u}$,

$$\frac{dF}{dt} := \frac{\partial F}{\partial t} + (\boldsymbol{u} \cdot \nabla) F. \tag{2.1}$$

If the material derivative is zero, then the field is materially conserved following the motion of the fluid.

[1] The quasi-geostrophic model is an approximation that filters out acoustic and internal gravity waves, hence allowing a larger time-step. This made the computations feasible on the ENIAC.

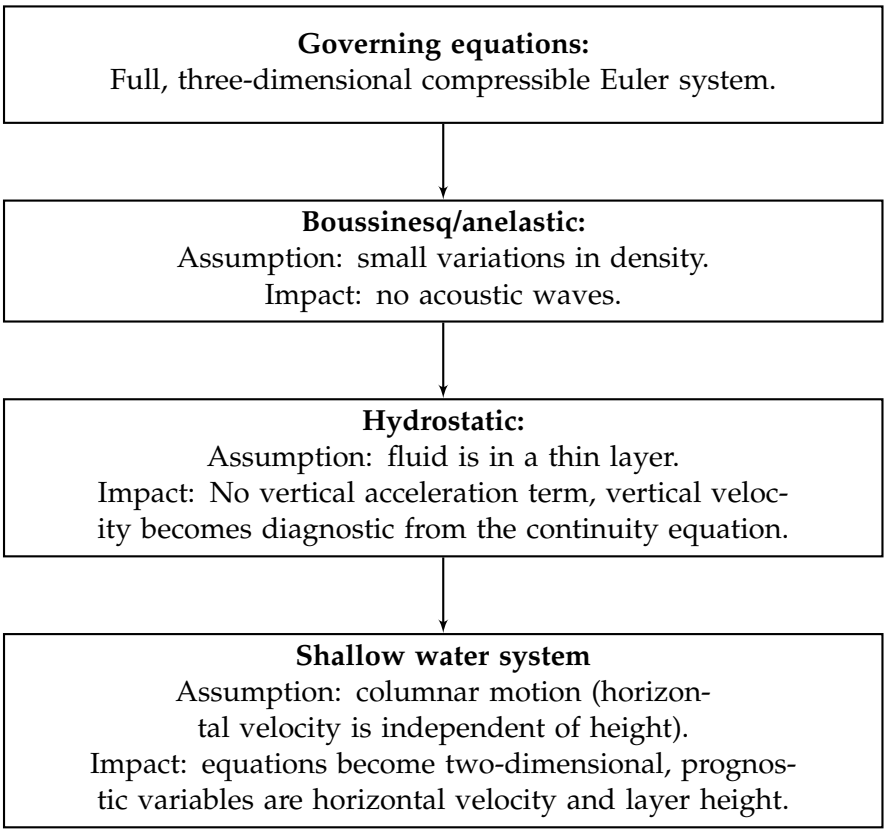


FIGURE 2.1: A hierarchy of models illustrating a successive application of various approximations to yield simplified models.

For global atmospheric models, the choice of domain is typically a spherical annulus in $\mathbb{R}^3$ with radius $R = 6,371$ km and a uniform height ranging between $70 - 100$ km. For regional or vertical slice simulations, the equations are defined in a Cartesian box. We assume that the air is dry (no moisture), inviscid (no viscous forces), and adiabatic (no sources or diffusion of temperature). The governing equations for a dry, inviscid, adiabatic, compressible fluid in a rotating reference frame with angular velocity $\boldsymbol{\Omega}$ may be written in the form

$$\frac{d\boldsymbol{u}}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\frac{1}{\rho}\nabla p - \nabla\Phi, \tag{2.2}$$

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\boldsymbol{u}\rho) = 0, \tag{2.3}$$

$$\frac{d\theta}{dt} = 0, \tag{2.4}$$

$$p = P(\rho, T), \tag{2.5}$$

where $\boldsymbol{u} = (u, v, w)$ is the fluid velocity, ρ is the fluid density, p is the pressure, and Φ is the geopotential comprising the gravitational and centrifugal potentials (often neglected as it is small compared to the gravitational potential). For our purposes, we will only consider the case for a geopotential of the form $\Phi = gz$, where g is the acceleration due to gravity. We use the potential temperature θ, defined as the temperature an air parcel would attain if moved adiabatically to a reference pressure

p_R. The ideal gas laws allow us to compute this explicitly:

$$\frac{T}{p^\kappa} = \frac{\theta}{p_R^\kappa} \implies \theta = T\left(\frac{p_R}{p}\right)^\kappa, \tag{2.6}$$

where p_R is a chosen reference pressure, and $\kappa = R/c_p$ is the ratio of the gas constant R and the specific heat at constant pressure c_p. The pay-off for this rather complicated formulation is that the potential temperature is constant along Lagrangian trajectories in the absence of diabatic processes.

Equation (2.5) closes the system of equations by relating pressure to the other thermodynamic variables. In the case of the atmosphere, the equation of state for an ideal gas is typically used, *i.e.*,

$$P(\rho, T) = \rho R T. \tag{2.7}$$

It is fairly common to use an alternative formulation of the pressure gradient term

$$\frac{1}{\rho}\nabla p = c_p \theta \nabla \Pi, \tag{2.8}$$

where Π is the Exner pressure given by the equation of state

$$\Pi = \left(\frac{R\rho\theta}{p_r}\right)^{\frac{\kappa}{1-\kappa}}. \tag{2.9}$$

One of the reasons for making this change is that it is then fairly simple to incorporate the thermodynamic effects of moisture into the pressure gradient term. The situation is more complicated in the case of the ocean, where the equation of state additionally depends on salinity. In this dissertation, we shall concentrate on the dry adiabatic case, but the techniques developed here are directly extensible to model the moist atmosphere.

These equations also need boundary conditions. As is the case for atmospheric dynamical cores, we shall mostly restrict ourselves to slip boundary conditions $\boldsymbol{u}\cdot\boldsymbol{n} = 0$ where $\boldsymbol{n}$ is the vector normal to the boundary. These are typically imposed on the upper and lower boundaries (top/bottom surfaces of the spherical annulus or Cartesian box). Another important boundary condition is the free surface boundary condition, $p = p_A$ (where p_A is an external reference pressure). In this case, the boundary surface must move with the velocity $\boldsymbol{u}$ at the boundary.

We note for later that (2.3) is an expression of local mass conservation; integration over a control volume V leads to

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V \rho \,\mathrm{d}\boldsymbol{x} + \int_{\partial V} \boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}S = 0, \tag{2.10}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ is the usual time derivative (not to be confused with the material derivative $\frac{d}{dt}$) and $\boldsymbol{n}$ is the outward pointing unit normal vector to the boundary ∂V of V. This shows that the rate of change of total mass in V is balanced by fluxes through the boundary ∂V.

At this point, it is worth discussing various approximations to the Coriolis term $2\boldsymbol{\Omega} \times \boldsymbol{u}$. For a model on the sphere, $\boldsymbol{\Omega}$ is aligned with the polar axis and has the form

$$\boldsymbol{\Omega} = (0, \Omega \cos\phi, \Omega \sin\phi)\,, \tag{2.11}$$

where $\Omega \approx 7.2921 \times 10^{-5}$ rad/s is the angular rotation rate of the Earth, and ϕ is the latitude. A common approximation is the *traditional approximation*, under which the vertical part of this force is neglected (since it is small compared to the gravitational force). For a coordinate system whose origin is at the center of the sphere, we write $\hat{\boldsymbol{r}} = \boldsymbol{x}/|\boldsymbol{x}|$ for the unit vector pointing away from the origin. Then the traditional approximation replaces the Coriolis term with

$$2\boldsymbol{\Omega} \times \boldsymbol{u} \approx f\hat{\boldsymbol{r}} \times \boldsymbol{u}, \tag{2.12}$$

where $f = 2\Omega \sin\phi$. Under this approximation, the Coriolis term vanishes at the equator and is maximal at the poles. We call f the *Coriolis parameter*.

For both mathematical simplicity and in the study of various models, a patch of the planetary surface can further be approximated by a Cartesian plane that is tangent at some longitude and latitude, λ_0 and ϕ_0 respectively. The motion can then be expressed in local (x, y, z) coordinates centered at λ_0 and ϕ_0, using the relation (see Vallis (2017, §2.3)):

$$(x, y, z) \approx (R\,(\lambda - \lambda_0)\cos\phi, R\,(\phi - \phi_0)\,, z)\,, \tag{2.13}$$

where R is the planet radius. We consider two common treatments of the Coriolis parameter here.

1. *f-plane approximation*: $f = f_0 = 2\Omega \sin\phi_0$ is taken to be constant-valued at a given latitude ϕ_0. This approximation is used frequently in the case of highly idealized flows. A notable consequence is that Rossby waves, which depend on variations in f, do not occur in models using this approximation.

2. *β-plane approximation*: Since f depends on variations in latitude, an f-plane approximation may not be appropriate when considering flows over large length scales. The β-plane approximation improves on this by considering linear variations via a Taylor expansion around a given latitude ϕ_0:

$$\begin{aligned} f = 2\Omega \sin\phi &= 2\Omega \sin\phi_0 + 2\Omega\,(\phi - \phi_0)\cos\phi_0 + \cdots \\ &\approx f_0 + \beta y, \end{aligned} \tag{2.14}$$

   where f_0 is the Coriolis parameter at ϕ_0, $y = R\,(\phi - \phi_0)$ is the distance in the meridional (north-south directions) from ϕ_0, and $\beta = \frac{2\Omega\cos\phi_0}{R} = \frac{\mathrm{d}f}{\mathrm{d}y}|_{\phi_0}$ is the Rossby parameter.

Under the f- or β-plane approximations, the momentum equation (2.2) becomes

$$\frac{du}{dt} - fv = -\frac{1}{\rho}\frac{\partial p}{\partial x}, \tag{2.15}$$

$$\frac{dv}{dt} + fu = -\frac{1}{\rho}\frac{\partial p}{\partial y}, \tag{2.16}$$

$$\frac{dw}{dt} = -\frac{1}{\rho}\frac{\partial p}{\partial z} - g. \tag{2.17}$$

2.1.2 Anelastic and Boussinesq approximations

The next level down in our model hierarchy is distinguished by the lack of acoustic waves that are present in the compressible Euler model. The anelastic approximation was first introduced in Ogura and Phillips (1962) to filter out acoustic modes without needing to assume hydrostatic balance. It has since been well-studied and modified (Durran, 1989; Durran, 2008). The approximation is based upon the assumption that there are only small variations in density relative to a reference field. We begin by writing

$$\rho = \tilde{\rho} + \delta\rho(\boldsymbol{x}, t), \tag{2.18}$$

where $\tilde{\rho}$ is a reference density field that only depends on the height (i.e., the radial direction on the sphere or the z direction in the plane).

After noticing that $\frac{\partial \tilde{\rho}}{\partial t} = 0$, we rewrite (2.3) as

$$\frac{\partial \delta\rho}{\partial t} + \nabla \cdot (\boldsymbol{u}\tilde{\rho}) + \nabla \cdot (\boldsymbol{u}\delta\rho) = 0. \tag{2.19}$$

Since $\delta\rho$ is much smaller than $\tilde{\rho}$, we neglect those terms to obtain the anelastic continuity equation

$$\nabla \cdot (\tilde{\rho}\boldsymbol{u}) = 0. \tag{2.20}$$

We now consider the pressure gradient term under this approximation:

$$p = \tilde{p} + \delta p(\boldsymbol{x}, t), \tag{2.21}$$

where $\tilde{p}$ is chosen such that it is the hydrostatic pressure corresponding to $\tilde{\rho}$, i.e., satisfying the balance equation

$$\hat{\boldsymbol{k}} \cdot \nabla \tilde{p} = -g\tilde{\rho}, \tag{2.22}$$

where $\hat{\boldsymbol{k}}$ is the unit vector in the upward direction (either $\hat{\boldsymbol{r}}$ for the sphere or $\hat{\boldsymbol{z}}$ for a planar geometry). In other words, the effects due to gravity are balanced by the pressure gradient. We will revisit this in Section 2.1.3 in more detail.

After this choice, the combination of the pressure gradient and the gravity force becomes

$$-\nabla p - g\hat{\boldsymbol{k}}\rho = -\nabla \delta p - g\delta\rho\hat{\boldsymbol{k}}. \tag{2.23}$$

The momentum equation (2.2) is then

$$\frac{d\boldsymbol{u}}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\frac{1}{\rho}\nabla \delta p - g\frac{\delta\rho}{\rho}. \tag{2.24}$$

The anelastic approximation first replaces ρ with $\tilde{\rho}$ everywhere while leaving $\delta\rho$ in the gravity term, and we get

$$\frac{d\boldsymbol{u}}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\frac{1}{\tilde{\rho}}\nabla \delta p - g\frac{\delta\rho}{\tilde{\rho}}. \tag{2.25}$$

In the case where $\tilde{\rho}$ is constant, say $\tilde{\rho} \equiv \rho_0$, we obtain the Boussinesq equations. This is particularly relevant for ocean modeling, since the density in the ocean does not deviate far from a constant reference value ρ_0. The complete set of Boussinesq

equations are

$$\frac{d\boldsymbol{u}}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -\nabla \bar{p} - b\hat{\boldsymbol{k}}, \tag{2.26}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{2.27}$$

$$\frac{d\theta}{dt} = 0, \tag{2.28}$$

$$b = B(\theta), \tag{2.29}$$

where $\bar{p} = \delta p/\rho_0$, and B is an equation of state either derived from (2.5), or more frequently, a linear approximation $B(\theta) = B_0 + \alpha\theta$, where B_0 and α are constants.

Another useful modification leads to the compressible Boussinesq equations, in which (2.27) is replaced by

$$\frac{\partial p}{\partial t} + c^2 \nabla \cdot \boldsymbol{u} = 0, \tag{2.30}$$

where c is an acoustic wave speed. Taking the limit $c \to \infty$ recovers the standard Boussinesq equations. This modification allows the introduction of compressibility effects without the full nonlinear pressure gradient term of the compressible Euler equations. This is particularly useful for testing implicit solver approaches in a simplified setting.

To complete the anelastic approximation for non-constant $\tilde{\rho}$, we turn our attention to just the vertical part of (2.25). With z as the coordinate in the vertical direction, we have

$$\frac{dw}{dt} = -\frac{1}{\tilde{\rho}}\frac{\partial \delta p}{\partial z} - g\frac{\delta\rho}{\tilde{\rho}} = -\frac{\partial}{\partial z}\left(\frac{\delta p}{\tilde{\rho}}\right) - \frac{\delta p}{\tilde{\rho}}\frac{\partial \ln(\tilde{\rho})}{\partial z} - g\frac{\delta\rho}{\tilde{\rho}}, \tag{2.31}$$

where the latter equality comes from the chain rule. Recalling the definition of potential temperature in (2.6), it can be shown using the ideal gas law that

$$c_p \ln(\theta) = c_v \ln(p) - c_p \ln(\rho) - \ln(R) - \frac{R}{c_p}\ln(p_R), \tag{2.32}$$

implying that

$$\frac{\delta\theta}{\theta} \approx \frac{1}{\gamma}\frac{\delta p}{\tilde{p}} - \frac{\delta\rho}{\tilde{\rho}}, \tag{2.33}$$

where $\gamma = c_p/c_v$ and $\delta\theta$ denotes variations in potential temperature (Vallis, 2017, §2.5). We can now eliminate $\delta\rho$ in the gravitational term of (2.31), using (2.32) and (2.33) to get

$$\begin{aligned}\frac{dw}{dt} &\approx g\left(\frac{\delta\theta}{\theta} - \frac{1}{\gamma}\frac{\delta p}{\tilde{p}}\right) - \frac{\partial}{\partial z}\left(\frac{\delta p}{\tilde{\rho}}\right) + \frac{\partial}{\partial z}\left(\ln(\theta) - \frac{1}{\gamma}\ln(\tilde{p})\right)\frac{\delta p}{\tilde{\rho}} \\ &= g\frac{\delta\theta}{\theta} - \frac{\partial}{\partial z}\left(\frac{\delta p}{\tilde{\rho}}\right) + \frac{\partial \ln(\theta)}{\partial z}\frac{\delta p}{\tilde{\rho}},\end{aligned} \tag{2.34}$$

where the final equality in (2.34) is derived by applying (2.22). We now make a final observation by recognizing that the term $\partial \ln(\theta)/\partial z = \theta^{-1}\partial\theta/\partial z$ actually defines a length scale in potential temperature. In the atmosphere, the vertical length scale in potential temperature (roughly 100 km) is much larger than the density vertical length scale (about 10 km), so the last term in (2.34) can be neglected under the

anelastic approximation. This gives us a vertical momentum equation of the form

$$\frac{dw}{dt} = b_a - \frac{\partial \bar{p}}{\partial z}, \tag{2.35}$$

where $b_a = g\delta\theta/\theta$ is the anelastic buoyancy and $\bar{p} = \delta p/\tilde{\rho}$. The horizontal momentum equations simplify in a far more direct manner by simply neglecting density variations. The complete set of anelastic equations are given by the system

$$\frac{d\boldsymbol{u}}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = b_a \hat{\boldsymbol{k}} - \nabla \bar{p}, \tag{2.36}$$

$$\nabla \cdot (\tilde{\rho}\boldsymbol{u}) = 0, \tag{2.37}$$

$$\frac{db_a}{dt} = 0. \tag{2.38}$$

These equations are similar in presentation to that of Ogura and Phillips (1962) and Vallis (2017). However, various forms of the equations exist (often referred to as pseudo-incompressible or modified-anelastic sets) such as the set proposed by Durran (1989).

Both the Boussinesq equations for the ocean and the anelastic equations for the atmosphere (together with the local Cartesian approximation) provide another level of models which approximate the more general Euler equations. They do not permit acoustic modes, and both have been very productive in limited applications, such as large-eddy or localized flow simulations. However, particularly for the atmosphere, they are not generally sufficient for global-scale circulation modeling. A more in-depth discussion of the relative merits of different approximations for global-scale modeling can be found in Davies et al. (2003).

2.1.3 Hydrostatic approximation

The hydrostatic approximation is valid under the assumption of a thin layer of fluid, i.e., the equations are being solved in a domain of horizontal scale L and vertical scale H, with $H \ll L$.[2] Under this approximation, acceleration of the fluid velocity in the vertical direction is neglected. We briefly mentioned the notion of hydrostatic balance in (2.22). Here, we describe the conditions under which this approximation is acceptable. Revisiting the vertical component of the momentum equation (2.36), the hydrostatic approximation yields

$$\frac{dw}{dt} + \frac{\partial \bar{p}}{\partial z} = \frac{1}{\tilde{\rho}} \frac{\partial \delta p}{\partial z} = b_a, \tag{2.39}$$

where $\frac{dw}{dt} = 0$ is a consequence of the hydrostatic approximation. Using the anelastic-Boussinesq equations as our example, we now derive the condition under which $\frac{dw}{dt}$ can be neglected in terms of length-scales.

We assume that the vertical velocity has a typical scale W, the horizontal velocity has a typical scale U, and the time-scale of fluid movement is of the order L/U. We

[2] This is a primary characteristic of atmospheric motion, as the dynamics is dominated by large-scale horizontal motion.

write the advection equation (2.38) in the form

$$\frac{d_H b_a}{dt} + wN^2 = 0, \tag{2.40}$$

where $\frac{d_H b_a}{dt} = \frac{\partial b_a}{\partial t} + \boldsymbol{u}_H \cdot \nabla b_a$ and $\boldsymbol{u}_H$ is the horizontal component of the velocity, and $N^2 = -\frac{\partial b_a}{\partial z}$ is the Brunt–Väisälä frequency. Performing scaling analysis on (2.40) gives

$$\frac{b_a U}{L} \sim WN^2 \implies W \sim \frac{b_a U}{N^2 L}. \tag{2.41}$$

If $\frac{dw}{dt} \ll b_a$, then we have from (2.41) that

$$\frac{WU}{L} \ll \frac{LN^2 W}{U} \implies \frac{U^2}{LN^2} \ll 1. \tag{2.42}$$

Introducing the aspect ratio of the domain $\epsilon = H/L$ and the Richardson number $\mathrm{Ri} = N^2 H^2/U^2$, (2.42) implies that

$$\frac{\epsilon^2}{\mathrm{Ri}} \ll 1. \tag{2.43}$$

This means that we can neglect the vertical component of the acceleration $\frac{du}{dt}$ if the aspect ratio ϵ is small or the Richardson number Ri is large. We call this the *hydrostatic approximation*. Under this approximation, we simply neglect the vertical acceleration in the governing equations:

$$\frac{d\boldsymbol{u}_H}{dt} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = b_a \hat{\boldsymbol{k}} - \nabla \bar{p}, \tag{2.44}$$

$$\nabla \cdot (\tilde{\rho} \boldsymbol{u}) = 0, \tag{2.45}$$

$$\frac{db_a}{dt} = 0, \tag{2.46}$$

where $\boldsymbol{u}_H = (u, v, 0)$.

The small-aspect ratio condition is a primary characteristic of atmospheric models. Therefore, the hydrostatic approximation is not limited to the Boussinesq equations and can be applied to, for example, the compressible Euler system in Section 2.1.1. The result is very much the same; motion in the vertical direction is dominated by the balancing effects of gravity and the pressure gradient, resulting in the vertical component of (2.2) reducing to:

$$\frac{1}{\rho}\frac{\partial p}{\partial z} = -g. \tag{2.47}$$

2.1.4 Single-layer rotating shallow water system

Thus far, we have consider three-dimensional models. Since the domains we are interested in geophysical modeling typically have small aspect ratios (the atmosphere is tens of kilometers high, but thousands of kilometers across), significant insight into large scale behaviour can be obtained by considering vertically-integrated models.

One of the simplest vertically-integrated models is the single-layer rotating shallow water equations. It allows for the analysis of flows under rotational effects within

a relatively simple framework. It can be described as the culmination of all previous approximations mentioned in this section. That is, it describes a thin layer of constant density (akin to the Boussinesq/anelastic approximation) in hydrostatic balance. Under the shallow water approximation, we assume that the horizontal velocity is independent of depth, i.e., the fluid is moving as vertical columns. Thus, the system is inherently a two-dimensional approximation. The shallow water equations consider the case where the upper surface of the fluid is free, so the pressure is constant there, and the surface moves at the speed of the total velocity $\boldsymbol{u}_H + (0, 0, w)$.

We start by summarizing the continuity equation of a fluid with horizontal velocity $\boldsymbol{u}_H$ and variable depth $D = D(x, y, t)$. In line with common notation for shallow water models, we write $\boldsymbol{u}$ in place of $\boldsymbol{u}_H$ (in a slight abuse of notation). This can be derived from three-dimensional mass conservation. Since the fluid has constant density, it is therefore incompressible and, in components, we have

$$\frac{\partial w}{\partial z} = -\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = -\nabla \cdot \boldsymbol{u}, \tag{2.48}$$

where $\nabla = (\partial_x, \partial_y, 0)$ now denotes the horizontal gradient. If we consider a vertical column of fluid with bottom depth $z = \eta_b$ and top $z = \eta_t$, integrating the left-hand side of (2.48), using the fact that $\boldsymbol{u}$ is independent of z, and applying the fundamental theorem of calculus gives

$$\int_{\eta_b}^{\eta_t} \frac{\partial w}{\partial z}\, \mathrm{d}z = w(\eta_t) - w(\eta_b) = -D\nabla \cdot \boldsymbol{u}, \tag{2.49}$$

where $D(x, y, t) = \eta_t(x, y, t) - \eta_b(x, y)$. We assume that no mass is lost at the free surface η_t. That is, no fluid crosses the surface (which is analogous to the ocean). Therefore, we have

$$\frac{d\eta_t}{dt} = w(\eta_t). \tag{2.50}$$

Similarly, the motion of the fluid must follow the bottom topography:

$$\frac{d\eta_b}{dt} = w(\eta_b). \tag{2.51}$$

Using (2.50) and (2.51), (2.49) becomes

$$\frac{d}{dt}(\eta_t - \eta_b) \equiv \frac{dD}{dt} = -D\nabla \cdot \boldsymbol{u}, \tag{2.52}$$

resulting in the continuity equation for the shallow water system,

$$\frac{dD}{dt} + D\nabla \cdot \boldsymbol{u} \equiv \frac{\partial D}{\partial t} + \nabla \cdot (\boldsymbol{u}D) = 0. \tag{2.53}$$

This equation has the same form as (2.3), with a different (but related) interpretation. Here it describes conservation of the total volume of fluid beneath the free surface.

Momentum balance can be derived directly from the hydrostatic balance equation, as previously discussed in Section 2.1.3,

$$\frac{\partial p}{\partial z} = -\rho g, \tag{2.54}$$

and since ρ is assumed to be constant, we may vertically integrate (2.54) to obtain

$$p(x,y,z,t) = -\rho g z + p_0. \tag{2.55}$$

At the top surface $z = \eta_t = D + \eta_b$, the pressure is determined by the weight of the fluid above, which is assumed to be negligible. Therefore we set $p(x,y,\eta_t) = 0$. The pressure inside the fluid layer is then

$$p(x,y,z,t) = \rho g(D(x,y,t) + \eta_b(x,y,t) - z). \tag{2.56}$$

Consequently, the horizontal pressure gradient is z-independent, with $\nabla p = \rho g \nabla(D + \eta_b)$. Using this relation, together with the traditional approximation for the Coriolis force, the u-component of the momentum equation becomes

$$\begin{aligned} \frac{du}{dt} - fv &= -\frac{1}{\rho}\frac{\partial p}{\partial x} \\ &= -g\frac{\partial}{\partial x}\left(D + \eta_b\right), \end{aligned} \tag{2.57}$$

where f is the Coriolis parameter. Similarly, the v-component of the momentum equation becomes

$$\frac{dv}{dt} + fu = -g\frac{\partial}{\partial y}\left(D + \eta_b\right). \tag{2.58}$$

Therefore, the rotating single-layer shallow water equations are

$$\frac{d\boldsymbol{u}}{dt} + f\boldsymbol{u}^\perp = -g\nabla\left(D + \eta_b\right), \tag{2.59}$$

$$\frac{\partial D}{\partial t} + \nabla \cdot (\boldsymbol{u}D) = 0, \tag{2.60}$$

where we have introduced the notation $\boldsymbol{u}^\perp = \hat{\boldsymbol{k}} \times \boldsymbol{u}$, a rotation of the two-dimensional velocity $\boldsymbol{u}$ in the plane. No additional thermodynamic equations are needed to close this system of equations, as thermodynamic effects were eliminated in the various assumptions that led to this model.

To gain some further insight into these equations, we perform some further manipulations to (2.59)–(2.60). Introducing the relative vorticity as the (two-dimensional) curl of the fluid velocity: $\zeta = \nabla^\perp \cdot \boldsymbol{u}$, where $\nabla^\perp = \hat{\boldsymbol{k}} \times \nabla$, we can rewrite the momentum equation in *vector-invariant* form:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\zeta + f)\boldsymbol{u}^\perp + \nabla\left(g(D + \eta_b) + \frac{1}{2}|\boldsymbol{u}|^2\right) = 0. \tag{2.61}$$

Taking the two-dimensional curl (also called rot) of (2.61) produces the following equation for the relative vorticity:

$$\frac{\partial \zeta}{\partial t} + (\boldsymbol{u} \cdot \nabla)(\zeta + f) + (\zeta + f)\nabla \cdot \boldsymbol{u} = 0, \tag{2.62}$$

where we have used the vector calculus identities $\nabla^\perp \cdot \nabla \equiv 0$ and

$$\nabla^\perp \cdot \left(a\boldsymbol{b}^\perp\right) = \nabla \cdot (a\boldsymbol{b}) = (\boldsymbol{b} \cdot \nabla)\, a + a\nabla \cdot \boldsymbol{b}, \tag{2.63}$$

for suitable scalar field a and vector $\boldsymbol{b}$. Since the Coriolis parameter f is independent of time, we can rewrite (2.62) as

$$\frac{d\left(\zeta+f\right)}{dt} = -\left(\zeta+f\right)\nabla\cdot\boldsymbol{u}. \tag{2.64}$$

Now, using the continuity equation (2.53), we multiply both sides of the expression by $\zeta+f$ and rearrange to obtain

$$-\left(\zeta+f\right)\nabla\cdot\boldsymbol{u} = \left(\frac{\zeta+f}{D}\right)\frac{dD}{dt}. \tag{2.65}$$

Comparing (2.64) with (2.65) gives the relation

$$\frac{d\left(\zeta+f\right)}{dt} = \left(\frac{\zeta+f}{D}\right)\frac{dD}{dt}. \tag{2.66}$$

Defining the shallow water potential vorticity $q = \frac{\zeta+f}{D}$, (2.66) can be further rewritten using the quotient rule to arrive at a transport equation for q:

$$\frac{1}{D}\frac{d\left(\zeta+f\right)}{dt} - \left(\frac{\zeta+f}{D^2}\right)\frac{dD}{dt} = \frac{d}{dt}\left(\frac{\zeta+f}{D}\right) = \frac{dq}{dt} = 0, \tag{2.67}$$

implying that q remains constant and is materially conserved moving with the fluid velocity.

In a boundary-free domain (such as surface of the sphere) or one with a rigid boundary (Cartesian box with slip boundary conditions $\boldsymbol{u}\cdot\boldsymbol{n}=0$), the shallow water equations have a conserved total energy E, and mass M:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{1}{2}\frac{\partial}{\partial t}\int_\Omega D|\boldsymbol{u}|^2 + g((D+\eta_b)^2-\eta_b^2)\,\mathrm{d}\boldsymbol{x} = 0, \tag{2.68}$$

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \frac{\partial}{\partial t}\int_\Omega D\,\mathrm{d}\boldsymbol{x} = 0. \tag{2.69}$$

To show energy conservation, we perform a sequence of algebraic manipulations of both the momentum and continuity equations. First, multiplying (2.59) by $\boldsymbol{u}$ produces

$$\frac{d}{dt}\left(\frac{|\boldsymbol{u}|^2}{2}\right) + g\boldsymbol{u}\cdot\nabla\left(D+\eta_b\right) = 0. \tag{2.70}$$

Now we multiply (2.70) by D to arrive at

$$\frac{\partial}{\partial t}\left(D\frac{|\boldsymbol{u}|^2}{2}\right) + \nabla\cdot\left(\boldsymbol{u}D\frac{|\boldsymbol{u}|^2}{2}\right) + g\boldsymbol{u}\cdot\left(\nabla\frac{D^2}{2} + D\nabla\eta_b\right) = 0, \tag{2.71}$$

where we used the continuity equation and the resulting identity

$$D\frac{d\Psi}{dt} = \frac{\partial D\Psi}{\partial t} + \nabla\cdot\left(D\Psi\boldsymbol{u}\right), \tag{2.72}$$

for some scalar-field Ψ. Equation (2.71) describes the evolution of the kinetic energy: $KE = D\frac{|\boldsymbol{u}|^2}{2}$. The potential energy PE of the fluid is obtained by vertically integrating

the geopotential:

$$PE = \int_{\eta_b}^{D+\eta_b} gz\mathrm{d}z = \frac{1}{2}g[(D+\eta_b)^2 - \eta_b^2]. \tag{2.73}$$

To derive an expression for the evolution of potential energy, we first multiply the continuity equation by gD and use (2.72) to get

$$\frac{\partial}{\partial t}\left(\frac{gD^2}{2}\right) + \nabla\cdot\left(\boldsymbol{u}\frac{gD^2}{2}\right) + \frac{gD^2}{2}\nabla\cdot\boldsymbol{u} = 0. \tag{2.74}$$

Multiplying the continuity equation by $g\eta_b$ and adding the result to (2.74), we obtain an equation for the evolution of the potential energy:

$$\frac{\partial}{\partial t}\left(\frac{1}{2}g[(D+\eta_b)^2 - \eta_b^2]\right) + \nabla\cdot\left(\boldsymbol{u}D\frac{|\boldsymbol{u}|^2}{2}\right) + \frac{gD^2}{2}\nabla\cdot\boldsymbol{u} + g\eta_b\nabla\cdot(D\boldsymbol{u}) = 0. \tag{2.75}$$

Adding (2.71) and (2.75) yields an evolution equation for the total energy: $PE + KE = \frac{1}{2}\left(D|\boldsymbol{u}|^2 + g((D+\eta_b)^2 - \eta_b^2)\right)$,

$$\begin{aligned}\frac{\partial}{\partial t}(PE+KE) &= -\nabla\cdot\left(\frac{\boldsymbol{u}}{2}[D|\boldsymbol{u}|^2 + gD^2]\right)\\ &\quad - g[D\boldsymbol{u}\cdot\nabla\eta_b + \eta_b\nabla\cdot(D\boldsymbol{u})] - \nabla\cdot\left(\boldsymbol{u}\frac{gD^2}{2}\right)\\ &= -\nabla\cdot\left[\frac{\boldsymbol{u}}{2}\left(D|\boldsymbol{u}|^2 + gD^2 + gD^2\right)\right] - g\nabla\cdot(D\eta_b\boldsymbol{u})\\ &= -\nabla\cdot\left[\frac{\boldsymbol{u}}{2}\left(D|\boldsymbol{u}|^2 + gD^2 + g[(D+\eta_b)^2 - \eta_b^2]\right)\right],\end{aligned} \tag{2.76}$$

or equivalently:

$$\frac{\partial}{\partial t}(PE+KE) + \nabla\cdot\boldsymbol{F} = 0, \tag{2.77}$$

where $\boldsymbol{F} = \frac{\boldsymbol{u}}{2}\left(D|\boldsymbol{u}|^2 + gD^2 + g[(D+\eta_b)^2 - \eta_b^2]\right)$ is the energy flux. Then, using (2.77), integrating over the domain, and applying the divergence theorem, we see that

$$\begin{aligned}\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{\partial}{\partial t}\int_\Omega PE + KE\,\mathrm{d}\boldsymbol{x} &= \frac{1}{2}\int_\Omega \frac{\partial}{\partial t}\left(D|\boldsymbol{u}|^2 + g((D+\eta_b)^2 - \eta_b^2)\right)\mathrm{d}\boldsymbol{x}\\ &= -\int_\Omega \nabla\cdot\boldsymbol{F}\,\mathrm{d}\boldsymbol{x} = 0,\end{aligned} \tag{2.78}$$

which holds for when the fluid is bounded by rigid walls, with $\boldsymbol{u}\cdot\boldsymbol{n} = 0$. In the case where Ω is the surface of a sphere, the domain is a manifold with no boundary ($\partial\Omega$ is the empty set) and the resulting surface integral arising from the divergence theorem evaluates to zero. Conservation of total mass in (2.69) can be verified via direct computation and applying the divergence theorem.

Linearized shallow water equations

Sometimes, it is useful to linearize the shallow water equations. In the absence of topography $\eta_b = 0$, a steady-state solution is a constant layer depth H at rest ($\boldsymbol{u} = \boldsymbol{0}$). Then the nonlinear fields $\boldsymbol{u}$ and D can be expressed in terms of *small* perturbations

$\delta \boldsymbol{u}$ and δD around this state:

$$\boldsymbol{u}(x,y,t) = \delta \boldsymbol{u}(x,y,t), \tag{2.79}$$
$$D(x,y,t) = H + \delta D(x,y,t). \tag{2.80}$$

Substituting (2.79) and (2.80) into (2.59) and (2.60), the momentum and continuity equations then become:

$$\frac{\partial \delta \boldsymbol{u}}{\partial t} + (\delta \boldsymbol{u} \cdot \nabla)\, \delta \boldsymbol{u} + f \delta \boldsymbol{u}^{\perp} = -g \nabla \delta D, \tag{2.81}$$
$$\frac{\partial \delta D}{\partial t} + (H + \delta D)\, \nabla \cdot \delta \boldsymbol{u} + \delta \boldsymbol{u} \cdot \nabla \delta D = 0. \tag{2.82}$$

After neglecting the second-order terms, we arrive at the linear equations:

$$\frac{\partial \delta \boldsymbol{u}}{\partial t} + f \delta \boldsymbol{u}^{\perp} = -g \nabla \delta D, \tag{2.83}$$
$$\frac{\partial \delta D}{\partial t} + H \nabla \cdot \delta \boldsymbol{u} = 0. \tag{2.84}$$

It is easy to see that these equations conserve the linearized energy,

$$E = \frac{1}{2} \int_{\Omega} H |\delta \boldsymbol{u}|^2 + g \delta D^2 \, \mathrm{d}\boldsymbol{x}, \tag{2.85}$$

by direct calculation:

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= \int_{\Omega} H \delta \boldsymbol{u} \cdot \frac{\partial \delta \boldsymbol{u}}{\partial t} + g \delta D \frac{\partial \delta D}{\partial t} \, \mathrm{d}\boldsymbol{x}, \\
&= \int_{\Omega} H \delta \boldsymbol{u} \cdot \left(-g \nabla \delta D - f \delta \boldsymbol{u}^{\perp} \right) - g \delta D H \nabla \cdot \delta \boldsymbol{u} \, \mathrm{d}\boldsymbol{x}, \\
&= \int_{\Omega} -g H \delta \boldsymbol{u} \cdot \nabla \delta D - g H \delta D \nabla \cdot \delta \boldsymbol{u} \, \mathrm{d}\boldsymbol{x}, \\
&= \int_{\Omega} -g H \nabla \cdot (\delta D \delta \boldsymbol{u}) \, \mathrm{d}\boldsymbol{x}, \\
&= 0,
\end{aligned} \tag{2.86}$$

using the divergence theorem as previously done for the nonlinear equations. A similar computation can be done to show conservation of mass. We will return to the linear system (2.83)–(2.84) in Chapter 3, as it is an ideal toy model for the introduction of the solution methods outlined in this dissertation.

Shallow water waves on the f-plane

These equations provide the simplest system that exhibits the slow-fast separation in geophysical fluid dynamics. To see this split, consider the f-plane approximation ($f \equiv f_0$, constant) with periodic boundary conditions. Differentiating the linear mass conservation equation (2.83) in time produces:

$$\frac{\partial^2 \delta D}{\partial t^2} - g H \nabla^2 \delta D + H f \left(\frac{\partial \delta v}{\partial x} - \frac{\partial \delta u}{\partial y} \right) = 0, \tag{2.87}$$

where we have used the momentum equation (2.84) and δu and δv denote the horizontal components of $\delta \boldsymbol{u}$ in the x- and y-direction respectively. Then taking the time-derivative of (2.87) and using the $\delta u/\delta v$ components of (2.84), we arrive at

$$\frac{\partial}{\partial t}\left(\frac{\partial^2 \delta D}{\partial t^2} + \left(f^2 - gH\nabla^2\right)\delta D\right) = 0. \tag{2.88}$$

Equation (2.88) supports two main types of wave solutions. To see this clearly, we substitute the wave ansatz in (2.88):

$$\delta D = \mathrm{Re}(\delta D_0) e^{i(\boldsymbol{k}\cdot\boldsymbol{x} - \omega t)}, \tag{2.89}$$

where $\mathrm{Re}(\delta D_0)$ denotes the real-part of the complex wave amplitude δD_0, i is the imaginary unit, $\boldsymbol{x} = (x, y)$, $\boldsymbol{k} = (k, l)$ is the horizontal wave number, and ω is the wave frequency. This leads to a characteristic polynomial in terms of ω to determine the dispersion relation:

$$\omega\left(\omega^2 - f^2 - c^2\left(k^2 + l^2\right)\right) = 0, \tag{2.90}$$

where $c = \sqrt{gH}$ is the wave speed. Immediately, we can see two possible solution branches.

"Slow" solutions arise when $\omega = 0$; this corresponds to time-independent flows which are in a state of *geostrophic balance*. That is, flows which have velocity $\delta \boldsymbol{u}_g = (\delta u_g, \delta v_g)$ (often referred to as the *geostrophic wind*), satisfying

$$f\delta \boldsymbol{u}_g^{\perp} = -g\nabla\delta D. \tag{2.91}$$

Such flows are divergence-free, and can be expressed as the curl of the stream function $\psi = g\delta D/f$ (hence $\delta \boldsymbol{u}_g = \nabla^{\perp}\psi$ and $\nabla \cdot \delta \boldsymbol{u}_g = 0$). The set of "fast" solutions correspond to the dispersion relation:

$$\omega^2 = f^2 + c^2\left(k^2 + l^2\right), \tag{2.92}$$

which are known as *Poincaré* waves. Note that in the special case when $(k^2 + l^2) \ll f^2/c^2$, then the dispersion relation can be approximated by $\omega = f$. Such waves are called *inertial oscillations*, which corresponds to flows satisfying

$$\frac{\partial \delta u}{\partial t} - f\delta v = 0, \quad \frac{\partial \delta v}{\partial t} + f\delta u = 0. \tag{2.93}$$

That is, internal oscillations are described by flow velocities which are spatially constant, unrestrained by the pressure gradient force, and oscillate with frequency f. For more information and in-depth discussion on the slow-fast separation of shallow water waves, we refer the reader to Vallis (2017, §3.7). The important point to keep in mind is that it is critical for the numerical scheme to maintain this slow/fast separation at the discrete level.

Shallow water waves on the β-plane

When $f = f_0 + \beta y$, the spatial variation in the Coriolis parameter introduces an additional mechanism that results in slowly propagating divergence-free solutions. To aid our discussion, we make the following definitions. We denote the time, length,

and velocity scales by T, L and U respectively. Then the Rossby number is defined as the ratio between inertial forces and the Coriolis force:

$$\mathrm{Ro} = \frac{U}{f_0 L}. \tag{2.94}$$

Now, we make the *quasi-geostrophic assumptions* (Vallis, 2017, §5.3):

1. The Rossby number is small: $\mathrm{Ro} \ll 1$, i.e. the flow is in near-geostrophic balance. More precisely, the fluid velocity takes the form: $\delta \boldsymbol{u} = \delta \boldsymbol{u}_g + \delta \boldsymbol{u}_{ag}$, where $\delta \boldsymbol{u}_g$ is the geostrophic wind satisfying (2.91), and $\delta \boldsymbol{u}_{ag}$ is the *ageostrophic wind*, where
$$\frac{|\delta \boldsymbol{u}_{ag}|}{|\delta \boldsymbol{u}_g|} \sim \frac{U_{ag}}{U} \sim \mathcal{O}(\mathrm{Ro}), \tag{2.95}$$
where U_{ag} is a velocity scale for the ageostrophic wind.

2. The scale of motion is not significantly larger than the Rossby deformation radius:
$$\frac{L^2}{L_D^2} \sim \mathcal{O}(1), \tag{2.96}$$
where $L_D = \sqrt{gH}/f_0$ is the Rossby deformation radius. In a single-layer shallow water model, this is equivalent to variations in the depth being small compared to the mean depth: $|\delta D| \ll H$. Note that this has implicitly been assumed since we shall start from the linearized equations.

3. Linear variations in the Coriolis term are small: $|\beta y| \sim |\beta L| \ll |f_0|$. More precisely:
$$\frac{\beta L}{f_0} \sim \mathcal{O}(\mathrm{Ro}). \tag{2.97}$$

4. Time scales advectively. That is, the scaling for time is given as $T = L/U$.

Revisiting the momentum equation (2.83), we rewrite using $\delta \boldsymbol{u} = \delta \boldsymbol{u}_g + \delta \boldsymbol{u}_{ag}$ and $f = f_0 + \beta y$:

$$\frac{\partial \delta \boldsymbol{u}_g}{\partial t} + \frac{\partial \delta \boldsymbol{u}_{ag}}{\partial t} + f_0 \delta \boldsymbol{u}_g^\perp + f_0 \delta \boldsymbol{u}_{ag}^\perp + \beta y \delta \boldsymbol{u}_g^\perp + \beta y \delta \boldsymbol{u}_{ag}^\perp = -g \nabla \delta D. \tag{2.98}$$

Performing scaling analysis of (2.98) (summarized in Table 2.1), we consider only terms of order $\mathcal{O}(\mathrm{Ro})$ (order 1 terms cancel due to the geostrophic balance relation), which yields:

$$\frac{\partial \delta \boldsymbol{u}_g}{\partial t} + f_0 \delta \boldsymbol{u}_{ag}^\perp + \beta y \delta \boldsymbol{u}_g^\perp = 0. \tag{2.99}$$

We perform a similar substitution for the linear mass conservation equation (2.84) to arrive at

$$\frac{\partial \delta D}{\partial t} + H \nabla \cdot \left(\delta \boldsymbol{u}_g + \delta \boldsymbol{u}_{ag} \right) = \frac{\partial \delta D}{\partial t} + H \nabla \cdot \delta \boldsymbol{u}_{ag} = 0, \tag{2.100}$$

where the final equality in (2.100) comes from the fact that the geostrophic wind is a divergence-free velocity field. Now we introduce (again) the stream function $\psi = g \delta D / f_0$, with $\delta \boldsymbol{u}_g = \nabla^\perp \psi$, to rewrite equations (2.98) and (2.100) as the following

TABLE 2.1: Scaling analysis of the terms in equation (2.98). The order 1 terms correspond to the geostrophically balanced part of the flow. The remaining terms are determined by orders of Ro and Ro^2. Equation (2.99) is obtained by neglecting the small $\mathcal{O}(\mathrm{Ro}^2)$ terms and observing that the order 1 terms cancel due to equation (2.91).

	$\frac{\partial \delta \boldsymbol{u}_g}{\partial t}$	$\frac{\partial \delta \boldsymbol{u}_{ag}}{\partial t}$	$f_0 \delta \boldsymbol{u}_g^\perp$	$f_0 \delta \boldsymbol{u}_{ag}^\perp$	$\beta y \delta \boldsymbol{u}_g^\perp$	$\beta y \delta \boldsymbol{u}_{ag}^\perp$	$-g \nabla \delta D$
	$\frac{U}{T}$	$\frac{U_{ag}}{T}$	$f_0 U$	$f_0 U_{ag}$	$\beta L U$	$\beta L U_{ag}$	$f_0 U$
$T = \frac{L}{U}$	$\frac{U^2}{L}$	$\frac{U U_{ag}}{L}$	$f_0 U$	$f_0 U_{ag}$	$\beta L U$	$\beta L U_{ag}$	$f_0 U$
Scale by $\frac{1}{f_0 U}$	$\frac{U}{f_0 L}$	$\frac{U_{ag}}{f_0 L}$	1	$\frac{U_{ag}}{U}$	$\beta \frac{L}{f_0}$	$\beta \frac{L U_{ag}}{f_0 U}$	1
(2.95), (2.97)	$\mathcal{O}(\mathrm{Ro})$	$\mathcal{O}(\mathrm{Ro}^2)$	1	$\mathcal{O}(\mathrm{Ro})$	$\mathcal{O}(\mathrm{Ro})$	$\mathcal{O}(\mathrm{Ro}^2)$	1

linear system:

$$\frac{\partial \nabla^\perp \psi}{\partial t} + f_0 \delta \boldsymbol{u}_{ag}^\perp + \beta y \nabla \psi = 0, \tag{2.101}$$

$$\frac{f_0}{g} \frac{\partial \psi}{\partial t} + H \nabla \cdot \delta \boldsymbol{u}_{ag} = 0. \tag{2.102}$$

To derive our desired equation, we first take the two-dimensional curl ($\nabla^\perp \cdot$) of (2.101) in order to obtain:

$$\frac{\partial}{\partial t} \nabla^2 \psi + \beta \frac{\partial \psi}{\partial x} + f_0 \nabla \cdot \delta \boldsymbol{u}_{ag} = 0. \tag{2.103}$$

And finally, we can use (2.102) to obtain

$$\frac{\partial}{\partial t} \left(\nabla^2 - \frac{1}{L_D^2} \right) \psi + \beta \frac{\partial \psi}{\partial x} = 0. \tag{2.104}$$

A dispersion relation for (2.104) can again be derived by substituting the wave ansatz, $\psi = \mathrm{Re}(\psi_0) e^{i(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t)}$, which leads to a single branch of solutions with a dispersion relation given by:

$$\omega \left(k^2 + l^2 + \frac{1}{L_D^2} \right) + \beta k = 0 \implies \omega = \frac{-\beta k}{\left(k^2 + l^2 + \frac{1}{L_D^2} \right)}. \tag{2.105}$$

These are known as Rossby waves, which are waves whose restoring force is the conservation of linear vorticity under y-variations in f (corresponding to the linear term βy). These are propagating divergence-free solutions that are slow compared to the unbalanced Poincaré waves on the f-plane. See Vallis (2017, §5.7) for a more general overview of Rossby waves and their dispersion relations for both linear and nonlinear models.

2.1.5 Numerical modeling of atmospheric flows

The use of computational models based on the numerical solution of PDEs to simulate physical processes is a powerful tool which complements experimentation, observation and theory. As the dynamics governing large-scale weather, ocean, and

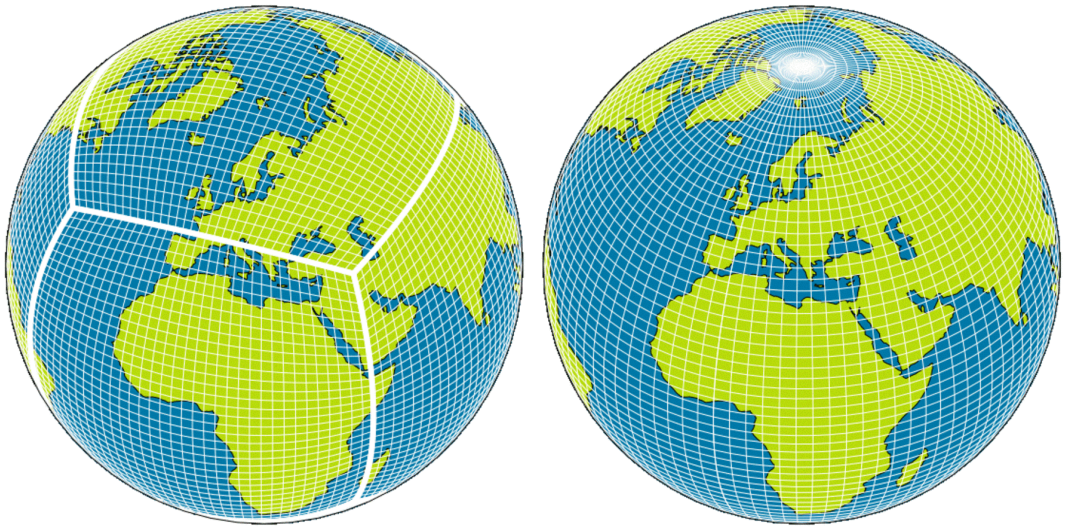


FIGURE 2.2: Examples of two horizontal grids: a quasi-uniform cubed sphere mesh (left) and traditional latitude–longitude grid (right). The polar singularity in the latitude–longitude mesh is clearly visible. Both grids were directly accessed from the UK Met Office 2019 blog post, "Next generation atmospheric model development" (Met Office, 2019).

climate processes cannot be replicated in a laboratory, numerical simulation is often the only mode of scientific inquiry available to researchers aiming to test, build, and improve existing models. Here, we summarize some important developments in the computational models used in large-scale atmospheric flows.

Structured orthogonal grids, such as the latitude–longitude grid (see Figure 2.2), are among the most widely-used grids in global forecast and ocean circulation models. Due to the convergence of the meridians at the poles, the resolution at or near the poles are vastly different than the rest of the mesh. For an explicit time-integration method with an Eulerian advection scheme, the Courant–Friedrichs–Lewy (CFL) condition (Courant, Friedrichs, and Lewy, 1928), which controls the maximum the time-step size given a particular resolution, imposes unbearably small time-steps for model convergence. To circumvent this, semi-implicit time-integration combined with semi-Lagrangian advection schemes (Staniforth and Côté, 1991; Williamson, 2007) are used to remove the severe time-step restriction. However, semi-implicit time-integrators require the solution of a globally-defined elliptic PDE at every time-step. Significant data communication among the grid points at the poles is required and presents an inescapable computational bottleneck, as processor communication is slow and can leave other processors waiting for data (Staniforth and Thuburn, 2011).

In contrast to typical engineering applications, where viscous and dissipative effects combined with outflow boundary conditions can alleviate the catastrophic accumulation of grid-scale errors, large-scale atmospheric modeling requires that the balanced forces of the continuous equations are exactly captured in the discrete model. The accurate and stable representation of large-scale force balances in a finite difference model demands a staggered variable arrangement, with pressure and velocity unknowns stored at different locations on the grid. The canonical taxonomy of variable staggerings for finite difference models is given in Arakawa and Lamb (1977). In that nomenclature, finite difference atmosphere models almost exclusively use C-grid staggering, as visualized in Figure 2.3. In this case, the components of velocity

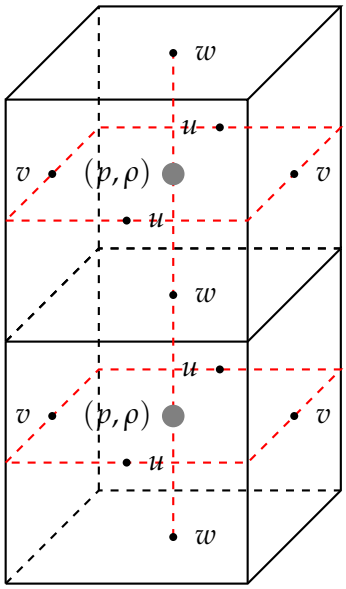


FIGURE 2.3: The "C-grid" proposed by Arakawa and Lamb (1977), which shows the finite difference staggering of the velocity $\boldsymbol{u} = (u, v, w)$, the pressure p, and fluid density ρ. The velocity components (u, v, w) are positioned in the center of cell faces, while the pressure/-density are collocated in the center of the cell.

(u, v, w) are positioned in the mid-point/center of cell boundaries (facets), and scalar variables like the pressure are stored in center of each cell.

In the context of horizontal discretizations, Staniforth and Thuburn (2011) considered a number of essential and desirable properties for an atmospheric dynamical core (the component of the full numerical model which solves the dynamical equations), which motivate the choice of compatible finite element discretizations discussed in this thesis. These properties are as follows:

1. Mass conservation of both active (dry air) and passive fields (trace species). This is desirable, but not essential for short-term weather prediction. It is more important when considering long-term climate simulations.

2. Accurate representation of flow close to hydrostatic and geostrophic balance is crucial. Slowly evolving, balanced flows should be accurately represented upon discretization; failure to do so results in a spontaneous loss of balance.

3. Computational modes should be absent, or at the very least well-controlled. As we have seen with the shallow water system in Section 2.1.4, after linearization around some reference profile, the continuous equations will support a spectrum of wave modes. This is also true for the discretized equations. Therefore, it is reasonable that the discrete wave modes will approximate those of the continuous equations. However, in some cases the discrete modes behave "unphysically" (zero frequency modes); we refer to these as "computational modes."

4. Grid imprinting should be minimal. Any non-uniform meshes will contain grid regions which are locally different to the vast majority of the mesh. This can lead to different error patterns, which may influence the resolved solution by coupling through nonlinear terms. For example, the latitude–longitude grid has very different structure at the poles (see Figure 2.2).

5. The geopotential term and pressure gradient should not produce unphysical sources of vorticity.
6. Terms involving the pressure should be energy conserving.
7. The discretization of the Coriolis force should be energy conserving.
8. Rossby waves should not propagate unrealistically fast.
9. Axial angular momentum should be conserved.

Properties (5)–(9) all rely on the discretization being "mimetic" in the sense that the discrete equations should mimic the basic geometric properties of the continuous system. In particular, discrete analogues of the calculus identities need to hold:

- $\nabla \times (\nabla \psi) = \mathbf{0}$, for a scalar function ψ.
- $\nabla \cdot (\nabla \times \boldsymbol{\omega}) = 0$, for a vector field $\boldsymbol{\omega}$.
- For a scalar field ψ and vector $\boldsymbol{\omega}$, then $\nabla \cdot (\psi \boldsymbol{\omega}) = \boldsymbol{\omega} \cdot \nabla \psi + \psi \nabla \cdot \boldsymbol{\omega}$.

These properties are all achievable with a finite difference C-grid staggering of the dynamic variables (Arakawa and Lamb, 1977), and hence this lays down the gauntlet for other discretizations. Hence, maintaining these mimetic properties will be a primary motivator for the use of compatible finite element methods.

There are two strong reasons why the latitude–longitude grid is used with the C-grid. The first is that the latitude–longitude grid is formed from quadrilaterals. This means that, globally, there are twice as many horizontal velocity values as pressure (each cell contains one pressure value, and four horizontal velocity values that are each shared with a neighbouring cell, so there are effectively two horizontal velocity values per cell). This mirrors the physical situation where each point has two horizontal velocity values and one pressure value, and helps to avoid spurious branches in the linear solution. Using a mesh constructed from other polygons alters the global ratio of velocity and pressure values. One possibility is to use triangular grids, which allow a pseudo-uniform coverage of the sphere from icosahedral triangulation, as used in the ICON model (Zängl et al., 2014), for example. With triangular cells, the shortage of velocity values (the ratio is 3:2 instead of 2:1) means that when $f = 0$ there are two branches of solutions to the Poincaré wave equation, one physical branch and one spurious branch. When $f \neq 0$, the situation is worse, with the two branches intertwined so that for some parts of the spectrum, one branch is the physical one, and for different parts of the spectrum, the other branch is unphysical. This issue causes serious problems when modeling the shallow water equations, and is also observed when modeling baroclinic flows in three dimensions (Danilov, 2010; Gassmann, 2011).

The second reason is that the latitude–longitude is used is that it is orthogonal. An orthogonal grid is one where the line joining the centres of two cells that share an edge crosses that edge at right-angles. This is a critical part of the classical C-grid formulation and was extended to arbitrary polygonal meshes in Ringler et al. (2010). The extension to non-orthogonal grids was provided in Thuburn and Cotter (2012).

To avoid the parallel scalability issues of the latitude–longitude grid, one possibility is to find other grids that cover the sphere in a more uniform manner. For the C-grid, a couple options include the icosahedral grid (triangles) and the cubed sphere (quadrilaterals) made by tiling the faces of a cube and deforming to a sphere (see Figure 2.2). The icosahedral grid has the spurious Poincaré waves mentioned above.

The orthogonal version of the cubed sphere grid has a slightly less severe version of the problem of the latitude–longitude grid, in that the ratio of the largest and smallest cell areas goes to infinity as the mesh is refined, with clustering of grid points around the cube vertices (Putman and Lin, 2007). The non-orthogonal cubed sphere version of the C-grid was also demonstrated to be inconsistent in Thuburn, Cotter, and Dubos (2014). Due to this inconsistency, The C-grid finite difference discretization is not being considered for the new atmospheric model of the UK Met Office (Adams et al., 2019).

In contrast, compatible finite element methods present a way of simultaneously satisfying the properties listed above and avoiding polar singularities in the problem domain (Cotter and Shipton, 2012; Cotter and McRae, 2014; Gibson et al., 2019a). The mimetic properties can be obtained without relying on an underlying orthogonal mesh structure. This allows for the use of far more general meshes, including the refined icosahedral sphere (triangular) and non-orthogonal cubed sphere (quadrilateral) meshes. Furthermore, the number of velocity and pressure degrees of freedom is less tightly coupled to the connectivity of the underlying mesh, and the finite element method presents a systematic mechanism for constructing higher order discretizations. This dissertation with focus purely on aspects of compatible finite elements. As a result, we first review basic terms and concepts relevant for the development of finite element discretizations.

2.2 Background on the finite element method

The finite element method (FEM) is one particular method for finding numerical solutions to PDEs. The method was first proposed by engineers within the context of solving structural analysis problems on complex domains, and its origins can be traced back to the work of Hrennikoff (1941). A couple years later, Courant developed the idea of the minimization of a functional using linear approximations over sub-domains, with values being specified at discrete nodes (Courant, 1943). In essence, this seminal paper provides the rigorous underpinning of finite element analysis (FEA), where approximation techniques are applied to solve variational formulations of PDEs. Subsequent papers by Argyris, Clough, Zienkiewicz and Cheung develop the theory associated with the matrices which arise from these approximation techniques (Argyris, 1955; Clough, 1960; Zienkiewicz and Cheung, 1965). Due to the contributions of numerous authors throughout the 20^{th} Century, the finite element method is a well-established and mathematically rigorous framework for generating discrete algorithms for computing solutions to well-posed boundary value problems.

For the purpose of this section, we assume some basic familiarity with FEM. However, we shall quickly present basic terminology and some universally less well-known concepts which provides fundamental information for the rest of this dissertation.

2.2.1 The finite element

We first define the notion of a finite element and a finite element space, which is the fundamental backbone of describing discretizations of PDEs based on FEM. Following Ciarlet (2002), the *finite element* is defined as the triple, $(K, V, \mathcal{N})$, where

- K is a non-empty domain with a Lipschitz continuous boundary ∂K;
- $V = V(K)$ is a finite dimensional space on K, with $\dim V = N$, consisting of real-valued functions; and
- $\mathcal{N}$ is a space of linear functionals over P. More precisely, $\mathcal{N} = \{n_i\}_{i=1}^N$ is a basis for the dual of V, V'.

The space V is often referred to as the space of local shape functions. For most finite element applications, V is simply taken to be an appropriate polynomial space of a specified degree, commonly denoted as $\mathcal{P}_k(K)$ (polynomials of degree $\leq k$). This applies to vector-valued quantities as well, using instead the d-vector polynomial space $[\mathcal{P}_k(K)]^d$. The set $\mathcal{N}$ are often referred to as the *nodes* (or degrees of freedom) of a finite element. The nodes n_i can take many forms depending on the type of finite element and are defined on some *geometric entity* of K (vertices, edges, faces, cell interiors). The choice of n_i heavily depends on the desired smoothness properties of the global approximation. We summarize below some common examples:

- (Point evaluation): $v(\boldsymbol{x}), \boldsymbol{x} \in K$.
- (Evaluation of all first derivatives): $\frac{\partial v(\boldsymbol{x})}{\partial x_i}, i = 1, \cdots, d$.
- (Evaluation of the normal component): $v(\boldsymbol{x}) \cdot \boldsymbol{n}$, where $\boldsymbol{n}$ is the outward-pointing normal vector on ∂K. The normal components are often evaluated through integration against polynomials defined on the facets.
- (Evaluation of the tangential component): $v(\boldsymbol{x}) \cdot \boldsymbol{t}$, where $\boldsymbol{t}$ is the tangential normal vector on ∂K. Similarly, these are typically evaluated by integrating $v(\boldsymbol{x}) \cdot \boldsymbol{t}$ against polynomials defined on cell edges.

A finite element is said to be V-unisolvent if, for any scalars $\{c_i\}_{i=1}^N \subset \mathbb{R}$, there exists a unique $v \in V$ such that

$$n_i(v) = c_i. \tag{2.106}$$

More simply put, a finite element being V-unisolvent is equivalent to the following conditions being satisfied:

1. $\dim P = \text{card}\mathcal{N} = N$;
2. There exists a set of function $\{\phi_i\}_{i=1}^N \subset V$ with

$$n_i(\phi_j) = \delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0 & \text{otherwise;} \end{cases} \tag{2.107}$$

3. The functions $\{\phi_i\}_{i=1}^N$ form a basis for V; and
4. Given any $v \in V$, we can write

$$v = \sum_{i=1}^N n_i(v)\phi_i. \tag{2.108}$$

Note that this relation is a direct consequence of items (1)–(3). We call the basis $\{\phi_i\}_{i=1}^N$ the *nodal basis*.

We now define the (local) *interpolant* for a particular finite element $(K, V, \mathcal{N})$. Let $\{\phi_i\}_{i=1}^N$ be the basis of V which is dual to $\mathcal{N}$. In other words, $\{\phi_i\}_{i=1}^N$ is the basis satisfying (2.107). If v is a function for which all $n_i \in \mathcal{N}$ are well-defined, then we define the local interpolant, $\mathcal{I}_K$, as the linear operator:

$$\mathcal{I}_K v = \sum_{i=1}^{N} n_i(v)\phi_i. \tag{2.109}$$

Note that (2.108) is precisely the result of interpolating v whenever $v \in V$ ($\mathcal{I}_K$ is the identity operator in this case). Now that we have defined the local interpolant, it is time to piece everything together.

Given some computational domain $\Omega \subset \mathbb{R}^n$, we wish to cover Ω with a finite number of *elements* in a way that varies with a resolution parameter h. To do this, we make the following definitions. We say $\mathcal{T}_h = \{K_i\}$ is a *subdivision* or *mesh* of Ω if:

- The interiors of each K_i are disjoint, i.e., $\text{Interior}(K_i) \cap \text{Interior}(K_j) = \emptyset$ whenever $i \neq j$.
- $\bigcup_i K_i = \overline{\Omega}$, where $\overline{\Omega}$ denotes the closure of Ω. In our context, this can best be understood as the union of Ω with its boundary.

In this case, we define the mesh-resolution parameter h as the maximal diameter of $K \in \mathcal{T}_h$ over all elements, $h = \max_{K\in\mathcal{T}_h}\{\text{diam}(K)\}$. Throughout this dissertation, we restrict ourselves to subdivisions/meshes without hanging nodes. That is, no vertex lies on the interior edge-or-facet of any other cell. Unless stated otherwise, all meshes $\mathcal{T}_h$ should be interpreted in this way.

Having established the notion of a mesh, we now define the *global interpolant*. Let $\mathcal{T}_h$ be a mesh of elements K for the computational domain Ω, where each K are equipped with a space of local shape functions $V(K)$. Let r denote the highest integer order of derivatives present in $\mathcal{N}$. Then the global interpolant for functions $v \in C^r(\overline{\Omega})$ (the space of functions with continuous derivatives up to order r) is defined via:

$$\mathcal{I}_{\mathcal{T}_h} v|_K = \mathcal{I}_K v, \quad \forall K \in \mathcal{T}_h. \tag{2.110}$$

The global interpolant is said to be of continuity order $m \leq r$ if $\mathcal{I}_{\mathcal{T}_h} v \in C^m(\overline{\Omega})$ for all $v \in C^r(\overline{\Omega})$. Following Brenner and Scott (2008), we can define a global C^m-finite element space, V_h, as the image of the global interpolant:

$$V_h = \{\mathcal{I}_{\mathcal{T}_h} v : v \in C^r(\overline{\Omega})\}. \tag{2.111}$$

Note that when strong (Dirichlet) boundary conditions are present, (2.111) must be modified accordingly. For example, if v must satisfy $v = 0$ on $\Gamma \subseteq \partial\Omega$, then we define the finite element space as: $V_{h,0} = \{\mathcal{I}_{\mathcal{T}_h} v : v \in C^r(\overline{\Omega}), v|_\Gamma = 0\}$.

The *global basis* $\{\Phi_{\hat{i}}\}$ of V_h is defined in terms of the nodal basis on each cell K. First, note that every node n_i of a finite element $(K, V, \mathcal{N})$, defined on some geometric entity of K, there is a corresponding *global enumeration* over the geometric entities of $\mathcal{T}_h$, say $\hat{i}$. Then the global basis function $\Phi_{\hat{i}}$ is defined via

$$\Phi_{\hat{i}}|_K = \phi_i^K, \forall K \in \mathcal{T}_h, \tag{2.112}$$

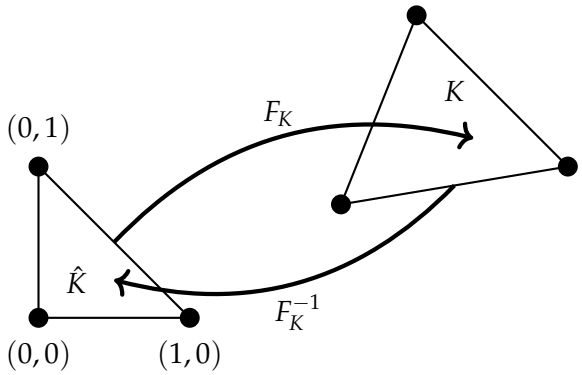


FIGURE 2.4: Illustration of the change-of-coordinates to and from the reference element (right triangle with vertices $(0,0)$, $(1,0)$, and $(1,0)$). While the diagram shows F_K for triangular cells, constructing F_K for general polygons is straightforward.

where ϕ_i^K is the local basis function of $V(K)$ satisfying (2.107), and $\Phi_{\hat{i}}|_K = 0$ if $n_{\hat{i}}$ is *not* a node associated with K.

In an implementation, it is common to construct a single *reference* finite element $(\hat{K}, \hat{V}, \hat{\mathcal{N}})$ and a mapping from physical cell K to the reference cell $\hat{K}$. The geometry of $\hat{K}$ is usually simpler, such as a right-triangle (or tetrahedra) for simplicial meshes or a unit square (cube) for hexahedra meshes. Let $F_K : \hat{K} \to K$ be an affine change-of-coordinates map. Then the *pullback* associated with F_K is defined as $F^*(v)(\hat{x}) = v(F_K(\hat{x}))$, for $\hat{x} \in \hat{K}$, $v \in V$. For a node $\hat{n} \in \hat{\mathcal{N}}$, the *pushforward* is defined as $F_*(\hat{n})(v) = \hat{n}(F^*(v)(\hat{x}))$. We say the finite elements $(\hat{K}, \hat{V}, \hat{\mathcal{N}})$ and $(K, V, \mathcal{N})$ are *affine equivalent* if $F^*(V) = \hat{V}$ and $F_*(\hat{\mathcal{N}}) = \mathcal{N}$. See Figure 2.4 for an illustration of the mapping from $\hat{K}$ to K.

A direct consequence of affine equivalence is that only a single nodal basis is required; all computations are performed on the reference cell and mapped to each physical cell in the mesh. We remark here that not all finite elements have an affine equivalent construction. Fortunately, the families we consider in this section do support such a construction. For some vector-valued spaces we consider, slightly more care is needed when applying pullbacks. We shall discuss this explicitly when the relevant family is presented. For more information on the construction of finite elements, we refer the interested reader to Brenner and Scott (2008) and Kirby et al. (2012). For the cases when the change-of-coordinate mapping is not affine, more care and details are needed. We elaborate on this further in Section 2.3.

Abstractly, solutions to PDEs belong to a particular Sobolev space. Finite element approximations are constructed by generating fields contained in a finite-dimensional subspace of the relevant Sobolev space. The choice of $\mathcal{N}$ determines, for example, whether or not taking gradients is mathematically well-posed. In this section, we summarize some common and perhaps less-common finite element families for approximating fields in various function spaces.

2.2.2 Finite element spaces

Sobolev spaces

The notion of a Sobolev space is ubiquitous in functional analysis and the theory of PDEs. Such spaces concisely characterize the smoothness of solutions to a PDE, or system of PDEs. Here, we present a few relevant spaces that appear when analyzing the equations of motion for geophysical systems.

We begin by defining the space of square-integrable functions. For some compact domain $\Omega \subset \mathbb{R}^d$, we define the space $L^2(\Omega)$ to be the set

$$L^2(\Omega) = \left\{ f : \Omega \to \mathbb{R} : \int_\Omega |f|^2 \mathrm{d}x < \infty \right\}. \tag{2.113}$$

When the domain is clear from the context, we abbreviate as simply L^2. For any two functions $f, g \in L^2$, we have that the operation:

$$\langle f, g \rangle_{L^2} = \int_\Omega fg \mathrm{d}x \tag{2.114}$$

defines a inner-product on L^2. Moreover, L^2 is *complete* with respect to the induced norm:

$$\|f\|_{L^2} = \sqrt{\langle f, f \rangle_{L^2}} = \sqrt{\int_\Omega |f|^2 \mathrm{d}x}, \tag{2.115}$$

making L^2 a *Hilbert space*.

The notion of a "derivative" when dealing within the context of Sobolev spaces is not defined in the classical sense. The traditional calculus definition of a derivative involves evaluating the limit of a difference quotient: for a given function f, the derivative of f with respect to x is given by

$$Df = \frac{\mathrm{d}f}{\mathrm{d}x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{2.116}$$

This is a *local* definition of a derivative, involving information about f only within a neighborhood of the points x. This is often called the *strong derivative*. Within our contexts, we are interested in the *global* behavior of functions, therefore we require extending the notion of a derivative in a more general setting.

We denote the differential operator in d-dimensions of order $|\alpha|$ to be D^α, where

$$D^\alpha = \left(\frac{\partial}{\partial x_1} \right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_d} \right)^{\alpha_d} = \frac{\partial^{|\alpha|}}{\partial_{x_1}^{\alpha_1} \ldots \partial_{x_d}^{\alpha_d}}, \tag{2.117}$$

$\alpha = (\alpha_1, \ldots, \alpha_d)$ is a multi-index, and $|\alpha| = \sum_{i=1}^d \alpha_i$. Now, suppose we have a functions f which is *locally integrable* on Ω ($\int_K |f| \, \mathrm{d}\boldsymbol{x} < \infty$ for all bounded, open sets $K \subset \Omega$). Suppose also that there exists a function g which is locally integrable on Ω with

$$\int_\Omega gv \, \mathrm{d}\boldsymbol{x} = (-1)^{|\alpha|} \int_\Omega f D^\alpha v \, \mathrm{d}\boldsymbol{x}, \tag{2.118}$$

for all functions v which are infinitely differentiable with bounded support in Ω, i.e., $v \in C_0^\infty(\Omega)$. Then when say g is the order $|\alpha|$ *weak derivative* of f, written as $g = D_w^\alpha f$.

Remark 1. *(Notation for differential operators): Clearly if f is sufficiently smooth, say $f \in C^m(\Omega)$, then the weak derivative $D^\alpha_w f$ of order $|\alpha| \leq m$ coincides with its strong derivative. For this reason, we simply write D^α to denote both strong and weak derivatives (in a slight abuse of notation). Within our contexts, functions will typically be polynomial expressions, which are sufficiently smooth for differentiation. Therefore it will be clear by the context which notion of the derivative will be considered.*

Having established a notion of a square-integrable function and weak derivatives, we now define a particular class of Sobolev spaces with integer order. With Ω as before and $m \in \mathbb{N}$, we define the set $H^m(\Omega)$ as

$$H^m(\Omega) = \{f : \Omega \to \mathbb{R} : D^\alpha f \in L^2(\Omega), \forall \alpha \text{ with } |\alpha| \leq m\}, \tag{2.119}$$

For all m, the spaces $H^m(\Omega)$ (henceforth abbreviated as H^m) are equipped with the inner product:

$$\langle f, g \rangle_{H^m} = \int_\Omega fg + \sum_{|\alpha| \leq m} D^\alpha f D^\alpha g \mathrm{d}x. \tag{2.120}$$

These spaces are also Hilbert, as they are complete under the induced norm $\|f\|_{H^m} = \sqrt{\langle f, f \rangle_{H^m}}$. The case when $m = 0$ ($H^0 = L^2$) and $m = 1$ are particularly relevant. The space

$$H^1(\Omega) = \{f : \Omega \to \mathbb{R} : \nabla f \in L^2(\Omega)\}, \tag{2.121}$$

is fundamental in the analysis of the weak formulations of second-order elliptic PDEs. Finite element subspaces of H^1 are one of the best-known spaces used in finite element modeling.

Given a vector field $\boldsymbol{u}$, we could also say $\boldsymbol{u} \in [H^1]^d$ if all spatial components are in H^1. However, there are varying levels of smoothness for vector quantities. For example, $\boldsymbol{u}$ may not be in $[H^1]^d$, but its divergence is square-integrable: $\nabla \cdot \boldsymbol{u} \in L^2$. In this case, we say that such vector fields belong to the Sobolev space

$$H(\mathrm{div}; \Omega) = \{\boldsymbol{u} : \Omega \to \mathbb{R}^d : \nabla \cdot \boldsymbol{u} \in L^2(\Omega)\}. \tag{2.122}$$

As before, $H(\mathrm{div})$ is a complete inner-product space, with inner product:

$$\langle \boldsymbol{u}, \boldsymbol{w} \rangle_{H(\mathrm{div})} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{w} \mathrm{d}x + \int_\Omega \nabla \cdot \boldsymbol{u} \nabla \cdot \boldsymbol{w} \mathrm{d}x. \tag{2.123}$$

We can define a similar Sobolev space for vectors with square-integrable curl (or rot in two-dimensions: $\nabla^\perp \cdot = -\frac{\partial}{\partial y} + \frac{\partial}{\partial x}$)

$$H(\mathrm{rot}; \Omega) = \{\boldsymbol{u} : \Omega \to \mathbb{R}^2 : \nabla^\perp \cdot \boldsymbol{u} \in L^2(\Omega)\}, \tag{2.124}$$

$$H(\mathrm{curl}; \Omega) = \{\boldsymbol{u} : \Omega \to \mathbb{R}^3 : \nabla \times \boldsymbol{u} \in [L^2(\Omega)]^3\}, \tag{2.125}$$

equipped with the inner products:

$$\langle \boldsymbol{u}, \boldsymbol{w} \rangle_{H(\mathrm{rot})} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{w} \mathrm{d}x + \int_\Omega \nabla^\perp \cdot \boldsymbol{u} \nabla^\perp \cdot \boldsymbol{w} \mathrm{d}x, \tag{2.126}$$

$$\langle \boldsymbol{u}, \boldsymbol{w} \rangle_{H(\mathrm{curl})} = \int_\Omega \boldsymbol{u} \cdot \boldsymbol{w} \mathrm{d}x + \int_\Omega \nabla \times \boldsymbol{u} \cdot \nabla \times \boldsymbol{w} \mathrm{d}x. \tag{2.127}$$

For simplicity, we write $H(\mathrm{curl})$ for both two- and three-dimensional cases.

Weak formulations of PDEs

The weak form of a PDE, also called the integral or variational formulation, is an alternate form of the PDE in which solutions are no longer required to hold in a pointwise sense. Instead, solutions are required to hold in an *integral sense* by means of testing against so-called *test functions* and integrating over the domain. For example, consider the model problem:

$$-\nabla^2 u + u = f, \text{ in } \Omega \tag{2.128}$$

$$u = 0, \text{ on } \partial\Omega_D, \tag{2.129}$$

$$\nabla u \cdot \boldsymbol{n} = g, \text{ on } \partial\Omega_N, \tag{2.130}$$

with $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$. We shall assume that $f \in L^2(\Omega)$ and $g \in L^2(\partial\Omega)$. Now, introducing the Sobolev space

$$H_0^1 = \{v \in H^1 : v|_{\partial\Omega_D} = 0\}, \tag{2.131}$$

the *weak formulation* reads as follows: find $u \in H_0^1$ such that

$$a(u,v) := \int_\Omega \nabla u \cdot \nabla v \, \mathrm{d}\boldsymbol{x} + \int_\Omega uv \, \mathrm{d}\boldsymbol{x} = \int_\Omega fv \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_N} gv \, \mathrm{d}s := L(v), \tag{2.132}$$

for all $v \in H_0^1$. All derivatives in (2.132) should be interpreted in the weak sense as defined in (2.118). Here, $a(u,v)$ is called a *bilinear form* and $L(v)$ is a linear functional (i.e. linear form). Applying the Lax-Milgram theorem (Lax and Milgram, 1955), the existence and uniqueness of a solution to (2.132) can easily be shown. We call such a solution the *weak solution*.

The weak form of the PDE is obtained after integrating by parts. The surface integral containing the directional derivative appears as a forcing term on the right-hand side due to the Neumann condition (2.130). In this sense, the Neumann condition is automatically satisfied after obtaining the weak-solution (hence the common nomenclature *natural boundary condition*). The Dirichlet condition (2.129), however, is an explicit condition on the solution, which requires modifying the solution space (2.131). For this reason, Dirichlet conditions are often referred to as *essential boundary conditions*.

Notice that the strong form of the equation (2.128) requires *at least* C^2-continuity of the solution. In the weak form, we need only enough continuity to take a single derivative. Since the equations are interpreted in an integral-sense, seeking solutions in H_0^1 is still meaningful. In fact, it can be shown that if u is a *strong* solution to (2.128), then it is also a weak solution to (2.132). The converse, however, is *not* true in general. Despite this, weak solutions are important since the vast majority of PDEs describing real-world phenomena do not produce smooth solutions.

The finite element method approximates solutions to the weak formulation of a given PDE. This is accomplished by constructing a finite-dimensional space V_h over a mesh of Ω, from which a piecewise approximation can be assembled by satisfying the discretized weak form over each element. We will elaborate on how finite element computations are performed in Section 2.3.

H^1 finite elements

Finite element subspaces of H^1 (that is, an H^1-finite element space) are typically the first family of elements introduced to students and researchers using finite element modeling. Among the most widely-used elements are the *Lagrange* family on simplices and hexahedra. These elements are globally C^0, with functions that are single-valued on cell vertices, edges, and faces. For our application focus, the Lagrange elements will be the element family with the most continuity requirements. Note that there exist element families which produce subspaces of higher-order Sobolev spaces, like the Argyris for H^2 (Argyris, Fried, and Scharpf, 1968). There are also other finite elements for H^1, such as the one proposed by Morley (1971). However, we will not be covering these families and instead focus on elements which are relevant for our purposes.

The Lagrange elements on simplices. Let $\mathcal{T}_h$ denote a mesh of cells $K \subset \mathbb{R}^d$, where K is a d-simplex (interval, triangle, or tetrahedron). Then the Lagrange element of order q, $\mathrm{P}_q(K)$, is defined using $V = \mathcal{P}_q(K)$, the space of polynomials up to degree q on K, and nodes:

$$n_i(p) = v(x^i), \quad i \in \{1, \cdots N_q\}, \quad p \in V. \tag{2.133}$$

Here, x^i are enumerated points on K such that points located on cell vertices, edges, and faces ensure C^0-continuity. There are a number of possible ways to generate such a set of points. The simplest construction are set of points defined by:

$$\{x^i\}_{i=1}^{N_q} = \begin{cases} i/q, & 0 \le i \le q, \text{ for } d = 1, \\ (i/q, j/q), & 0 \le i + j \le q, \text{ for } d = 2, \\ (i/q, j/q, k/q), & 0 \le i + j + k \le q, \text{ for } d = 3. \end{cases} \tag{2.134}$$

The integer N_q denotes the dimension of the function space $\mathcal{P}_q$ on the simplex element, which is $N_q = q + 1$ on intervals, $N_q = \frac{1}{2}(q+1)(q+2)$ on triangles, and $N_q = \frac{1}{6}(q+1)(q+2)(q+3)$ on tetrahedra.

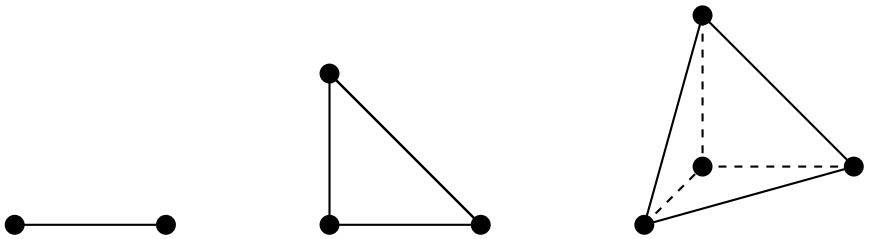
When mapping from physical to reference cells, a simple affine change of coordinates is sufficient. Let $\hat{v}(\hat{x})$ be a function defined on the reference cell $\hat{K}$ and $F_K : \hat{K} \to K$ be the affine change-of-coordinate mapping as previously defined (illustrated in Figure 2.4). Then the function on the physical cell K is simply

$$v(x) = v(F_K(\hat{x})) = \hat{v}(\hat{x}), \tag{2.135}$$

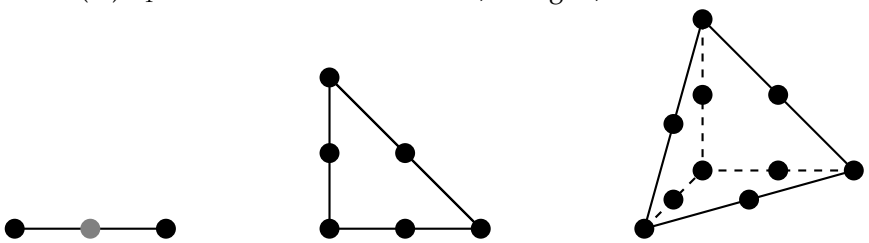
which we denote as $v = \mathcal{F}(\hat{v})$. Linear and quadratic Lagrange finite elements are illustrated in Figure 2.5.

The Lagrange elements on quadrilaterals and regular hexahedra. The Lagrange elements on quadrilaterals and cubes follow a very similar construction; we use the same continuity constraints as before, ensuring a globally C^0 function space. The key difference lies in the construction of a polynomial basis for $\mathcal{P}_q(K)$, which admits a *tensor product* structure. As a result, these elements are often referred to as the "continuous tensor product element" of order q, commonly abbreviated as $\mathrm{Q}_q(K)$.

We first remark that a unit square can be interpreted as the region defined by the bounded box of the Cartesian product: $[0,1]^2 = [0,1] \times [0,1]$. Similarly for the



(A) P_1 finite elements on intervals, triangles, and tetrahedra.



(B) P_2 finite elements on intervals, triangles, and tetrahedra.

FIGURE 2.5: Linear and quadratic Lagrange finite elements on simplicial cells. The filled discs denote point-wise evaluations on specified locations on the cell. Black discs denote degrees of freedom which couple to adjacent cells and impose continuity; gray discs denote degrees of freedom which are associated with the cell only.

unit cube, we have $[0,1]^3 = [0,1] \times [0,1] \times [0,1]$. A function space defined on $K = [0,1]^d$ can be constructed by taking a polynomial space defined on the unit interval, $\mathcal{P}_q([0,1])$ and taking the tensor product with itself to create a function space of tensor product polynomial functions:

$$\mathcal{P}_{q_1,\cdots,q_d}(K) = \bigotimes_{i=1}^{d} \mathcal{P}_{q_i}([0,1]) = \mathrm{Span}\{v_1 \cdot v_2 \cdots v_d : v_i \in \mathcal{P}_{q_i}([0,1])\}. \tag{2.136}$$

Note that, in this definition, we allow the possibility for the q_i to vary, meaning that different polynomial degrees can be used in different spatial directions.

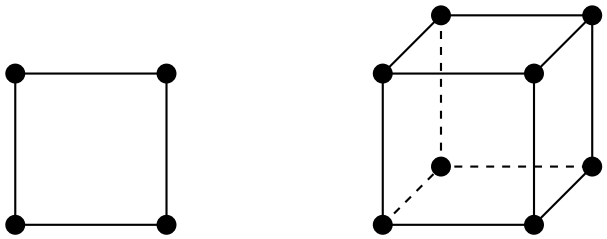
The finite element $\mathrm{Q}_q(K)$ is therefore defined by taking V as in (2.136), with $d = 2$ or $d = 3$ in most circumstances, $q_i = q$ for all i, and defining $\mathcal{N}$ similarly to the degrees of freedom for P_q; if P_q is a finite element defined on an interval, say $[0,1]$, with nodes $\mathcal{N}$, then the nodes of Q_k on the unit square are:

$$\mathcal{N}_{\mathrm{Q}_q} = \{n_i(v) \cdot n_j(v), \text{ with } n_i, n_j \in \mathcal{N}\}. \tag{2.137}$$

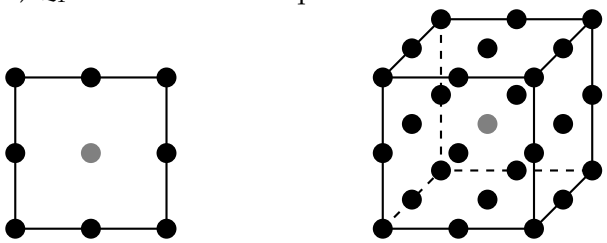
The construction of V and $\mathcal{N}_{\mathrm{Q}_q}$ on a unit cube follows similarly. Moreover, pulling back to a reference element (typically the unit square or cube) follows similarly to the Lagrange elements on simplices. However, the transformation F_K is no longer affine in this case. This will only affect how finite element integrals are evaluated numerically, which we cover in Section 2.3.

In general, the dimension of Q_q on $\mathbb{R}^d$ is $(q+1)^d$, which is a result of taking the product of dimension of one-dimensional P_q finite element spaces. Therefore, $\mathrm{Q}_q(K)$, $K = [0,1]^d \subset \mathbb{R}^d$ is more concisely defined as

$$\mathrm{Q}_q(K) = \bigotimes_{i=1}^{d} \mathrm{P}_q([0,1]). \tag{2.138}$$



(A) Q_1 finite elements on quadrilateral and cube cells.



(B) Q_2 finite elements on quadrilateral and cube cells.

FIGURE 2.6: Linear and quadratic Lagrange finite elements on quadrilateral and cube cells.

Diagrams of the Q_q family are presented in Figure 2.6.

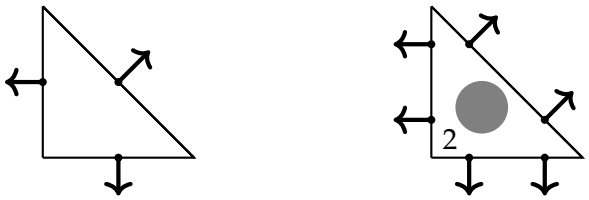
We will return to the notion of taking the tensor product of finite elements in Section 2.4. For a more general discussion, we refer the reader to Arnold, Boffi, and Bonizzoni (2015) and McRae et al. (2016). Lastly, we remark here that to create a vector-valued Lagrange element (simplices or cuboids) can be done by taking the components to be elements of the scalar Lagrange family.

$H(\mathrm{div})$ finite elements

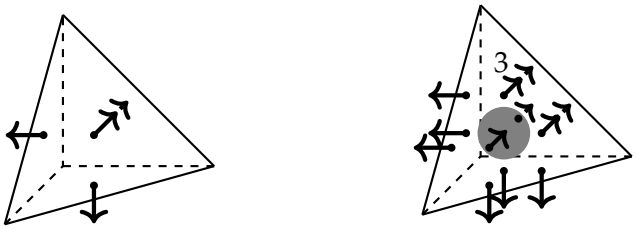
As previously mentioned, the Sobolev space $H(\mathrm{div})$ consists of vector fields whose divergence is square-integrable. These elements consist of vector polynomials with continuous *normal* components through element edges/faces. The tangential components are not required to be continuous. They are particularly important for discretizing geophysical flow equations and will therefore be used extensively throughout this dissertation.

The first $H(\mathrm{div})$ finite element was presented by Raviart and Thomas (1977) on triangular meshes for mixed formulations of second-order elliptic problems. It was later extended by Nédélec (1980) to tetrahedra and cubical meshes. For this reason, these elements are sometimes referred to as the *Raviart-Thomas-Nédélec* elements. More commonly, they are simply referred to as the Raviart-Thomas elements in most of the literature. Using notation inspired by the "Periodic Table of Finite Elements" (Arnold and Logg, 2014), we will denote the two-dimensional family order q as RT_q. Similarly, the three-dimensional family will be written as $\mathrm{N1}^f_q$.[3]

[3] Some choose to enumerate the RT_q and $\mathrm{N1}^f_q$ elements by the lowest-order polynomial space that completely spans the element, resulting the in lowest-order elements being denoted as RT_0 and $\mathrm{N1}^f_0$. This was original convention by Raviart and Thomas (1977). However, we use the second convention by enumerating RT_q (and $\mathrm{N1}^f_q$) by the highest-degree polynomial contained in the space, resulting in the lowest-order element being labeled as RT_1 and $\mathrm{N1}^f_1$. This is also the convention used by Nédélec (1980).



(A) RT_1 and RT_2 finite elements on triangular cells.



(B) $\mathrm{N1}_1^f$ and $\mathrm{N1}_2^f$ finite elements on tetrahedral cells.

FIGURE 2.7: The lowest-order and next-to-lowest-order $H(\mathrm{div})$ elements on simplicial cells. The outward pointing arrows denote normal component evaluations and the large shaded circles denote interior moment evaluations. The numerical values near the shaded circles display the total number of interior degrees of freedom.

Lastly, we carefully mention that Nédélec presented a second construction of $H(\mathrm{div})$ (and $H(\mathrm{curl})$) elements on tetrahedra, referred to as Nédélec elements of the *second kind*, denoted as $\mathrm{N2}_q^f$ ($\mathrm{N2}_q^e$ for $H(\mathrm{curl})$) (Nédélec, 1980). We will not be presenting these, but they are useful to be aware of. We will, however, define the family these elements are derived from: the Brezzi-Douglas-Marini $H(\mathrm{div})$ element in this section. As before, we start by defining $H(\mathrm{div})$ elements on simplicial cells.

The Raviart-Thomas-Nédélec $H(\mathrm{div})$ elements on simplices. The finite elements $\mathrm{RT}_q(K)$ and $\mathrm{N1}_q^f(K)$ are defined by taking V to be the smallest subspace of $[\mathcal{P}_q(K)]^d$ for which the divergence operator $\nabla\cdot$ maps *surjectively* to $\mathcal{P}_{q-1}(K)$. Construction of such a V takes the form:

$$V = [\mathcal{P}_{q-1}(K)]^d + \boldsymbol{x}\mathcal{P}_{q-1}(K), \tag{2.139}$$

and the nodal degrees of freedom $\mathcal{N}$ are defined as

$$\mathcal{N} = \begin{cases} \int_f \boldsymbol{v}\cdot\boldsymbol{n}\psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(f), \forall f \in \partial K, \\ \int_K \boldsymbol{v}\cdot\boldsymbol{\phi}\mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in [\mathcal{P}_{q-2}(K)]^d, q \geq 2. \end{cases} \tag{2.140}$$

The degrees of freedom on the faces (edges in two-dimensions) are *normal component evaluations*, which are present in all orders of the RT_q family. When $q \geq 2$, additional degrees of freedom are introduced in element interiors as *interior moment evaluations*. The dimension of RT_q and $\mathrm{N1}_q^f$ are given by $q(q+2)$ and $\frac{1}{2}q(q+1)(q+3)$ respectively. See Figure 2.7 for an illustration of the RT_q and $\mathrm{N1}_q^f$ elements on simplex cells.

The Brezzi-Douglas-Marini $H(\mathrm{div})$ element on simplices. The element $\mathrm{BDM}_q(K)$ was first introduced in Brezzi, Douglas, and Marini (1985) as an alternative to the Raviart-Thomas elements using a complete polynomial space. It was extended by

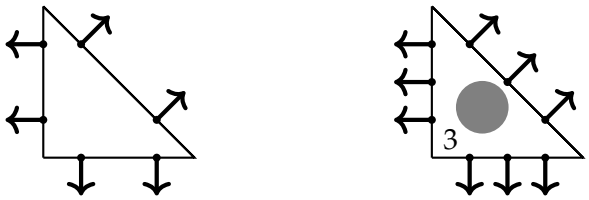


FIGURE 2.8: The BDM_q element for orders $q = 1$ (left) and $q = 2$ (right) on triangular cells.

Nédélec (1986) to tetrahedra, prisms, and cubes, giving rise to the Nédélec elements of the second kind: $\mathrm{N2}_q^f$ and $\mathrm{N2}_q^e$. The definition we present here is based on the construction by Nédélec (1986).

With K a triangle or tetrahedron, the $\mathrm{BDM}_q(K)$ element is defined by taking $V = [\mathcal{P}_q(K)]^d$ and degrees of freedom:

$$\mathcal{N} = \begin{cases} \int_f \boldsymbol{v} \cdot \boldsymbol{n} \psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_q(f), \forall f \in \partial K, \\ \int_K \boldsymbol{v} \cdot \boldsymbol{\phi} \mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in \mathrm{RT}_{q-1}^e(K), \text{ if } d = 2, \\ & \forall \text{ basis functions } \boldsymbol{\phi} \in \mathrm{N1}_{q-1}^e(K), \text{ if } d = 3, \\ & q \geq 2, \end{cases} \tag{2.141}$$

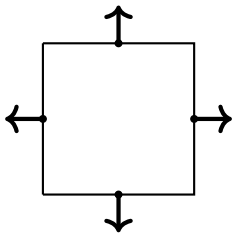
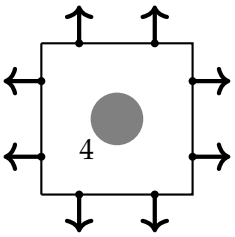
where RT_q^e and $\mathrm{N1}_q^e$ are the Raviart-Thomas edge elements and the Nédélec first kind edge elements, defined later for $H(\mathrm{curl})$ spaces. Figure 2.8 illustrates the degrees of freedom on triangular cells. The dimension of BDM_q is $(q+1)(q+2)$ for triangles and $\frac{1}{2}(q+1)(q+2)(q+3)$ for tetrahedra. These elements are presented for quadrilaterals and cubes in more detail in (Brezzi et al., 1987a). Another variation of the BDM_q element is the so-called BDFM_q family (Brezzi-Douglas-Fortin-Marini) on triangles and quadrilaterals. The element was introduced on quadrilaterals (Brezzi et al., 1987a) and later extended to triangular elements (Brezzi and Fortin, 1991). They are not typically used on triangles due to the larger number of degrees of freedom compared with BDM_q or RT_q of the same order.

The Raviart-Thomas-Nédélec $H(\mathrm{div})$ elements on cuboids. The Raviart-Thomas family was extended to cuboids by Nédélec (1980), and follows a similar construction. As we have done for the $H(\mathrm{div})$ elements, we will distinguish between two- and three-dimensional families. Similarly with the Q_q elements, the choice of V will consist of tensor product polynomials. Henceforth abbreviated as RTC_q^f and NC_q^f for two- and three-dimensions respectively, the $H(\mathrm{div})$ elements are constructed by taking V defined as

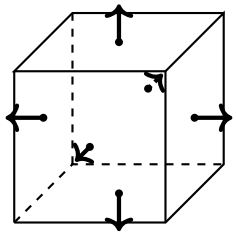
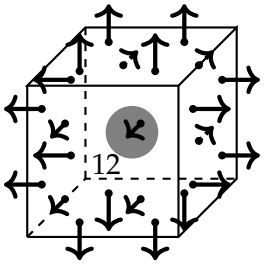
$$V = \begin{cases} \mathcal{P}_{q,q-1}(K) \times \mathcal{P}_{q-1,q}(K) & \text{for quadrilaterals,} \\ \mathcal{P}_{q,q-1,q-1}(K) \times \mathcal{P}_{q-1,q,q-1}(K) \times \mathcal{P}_{q-1,q-1,q}(K) & \text{for cubes,} \end{cases} \tag{2.142}$$

where the spaces $\mathcal{P}_{q,r}$ (and its three-dimensional variants) are as defined in (2.136). Similar to the construction of RT_q, the space V is constructed such that $\nabla\cdot$ maps surjectively to $\mathcal{P}_{q-1,q-1}$ (or $\mathcal{P}_{q-1,q-1,q-1}$ in three-dimensions). In two-dimensions, the degrees of freedom for the RTC_q^f element is nearly identical to RT_q. Setting

$$\Psi_q(K) = \begin{cases} \mathcal{P}_{q-1,q}(K) \times \mathcal{P}_{q,q-1}(K), & d = 2, \\ \mathcal{P}_{q-1,q,q}(K) \times \mathcal{P}_{q,q-1,q}(K) \times \mathcal{P}_{q,q,q-1}(K), & d = 3, \end{cases} \tag{2.143}$$

(A) RTC_1^f and RTC_2^f finite elements on quadrilateral cells.

(B) NC_1^f and NC_2^f finite elements on cube cells.

FIGURE 2.9: The lowest-order and next-to-lowest-order $H(\mathrm{div})$ elements on quadrilateral cells. Normal component evaluations are denoted by outward arrows, and interior moments by the shaded gray circle.

we define $\mathcal{N}$ as the set of functionals:

$$\mathcal{N}_{\mathrm{RTC}_q^f} = \begin{cases} \int_f \boldsymbol{v}\cdot\boldsymbol{n}\psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(e), \forall \text{ edges } e, \\ \int_K \boldsymbol{v}\cdot\boldsymbol{\phi}\mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in \Psi_{q-2}(K), q \geq 2, \end{cases} \tag{2.144}$$

for the RTC_q^f element. The NC_q^f element follows similarly:

$$\mathcal{N}_{\mathrm{NC}_q^f} = \begin{cases} \int_f \boldsymbol{v}\cdot\boldsymbol{n}\psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1,q-1}(f), \forall \text{ faces } f, \\ \int_K \boldsymbol{v}\cdot\boldsymbol{\phi}\mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in \Psi_{q-2}(K), q \geq 2. \end{cases} \tag{2.145}$$

The dimensions of RTC_q^f and NC_q^f are given by $2(q+1)(q+2)$ and $3(q+1)^2(q+2)$ respectively. See Figure 2.9 for element diagrams showing the layout of the degrees of freedom.

The contravariant Piola transform. When mapping from reference to physical cell, it is imperative to preserve the orientation of normal components. Otherwise, we are not guaranteed to preserve the continuity of normal components after mapping to the physical cell. The standard pullback defined previously for H^1 elements is not sufficient.

Let $\boldsymbol{x} = (x_1, \cdots, x_d)$ denote the physical coordinates and $\hat{\boldsymbol{x}} = (\hat{x}_1, \cdots, \hat{x}_d)$ be the coordinates on the reference cell. With J denoting the Jacobian of the change-of-coordinates mapping $F_K : \hat{K} \to K$, i.e. $J_K = DF_K(\hat{\boldsymbol{x}}) = \frac{\partial F_K(\hat{\boldsymbol{x}})}{\partial \hat{\boldsymbol{x}}}$, we define the *contravariant Piola transform* of a vector function $\boldsymbol{v}(\boldsymbol{x}) : K \to \mathbb{R}^d$ to be

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{J_K}{|J_K|}\boldsymbol{v}(F_K(\hat{\boldsymbol{x}})) = \frac{J_K}{|J_K|}\hat{\boldsymbol{v}}(\hat{\boldsymbol{x}}) =: \mathcal{F}_{\mathrm{div}}(\hat{\boldsymbol{v}}), \tag{2.146}$$

where $\hat{\boldsymbol{v}}(\hat{\boldsymbol{x}}) : \hat{K} \to \mathbb{R}^d$ is a vector function defined on the reference cell $\hat{K}$, and $|J_K|$ is the Jacobian determinant. This mapping is applied as the appropriate pullback for the $H(\mathrm{div})$ finite elements presented here.

$H(\mathbf{curl})$ finite elements

The Sobolev space $H(\mathrm{curl})$ frequently appears in electromagnetism applications and fluid dynamics problems (especially when using vorticity-velocity formulations of fluid flow motion). One variant of an $H(\mathrm{curl})$ finite element was first presented by Nédélec (1980) and a second family was constructed in (Nédélec, 1986; Brezzi et al., 1987a). They are often referred to as *edge elements* due to the fact that the elements involve degrees of freedom along cell edges. By construction, the tangential components must be continuous along cell edges, while normal components are free to be discontinuous.

We remark here that, while this dissertation will not use these elements extensively, they are useful to mention as they help construct a complete set of finite element families which are useful for approximating solutions to the geophysical flow equations discussed throughout later chapters. We will summarize this in Section 2.4. Lastly, we primarily discuss the elements known as $\mathrm{RT}_q^e/\mathrm{N1}_q^e$ and $\mathrm{RTC}_q^e/\mathrm{NC}_q^e$: the Nédélec $H(\mathrm{curl})$ elements of the *first kind* on simplices and cuboids respectively. Construction for the elements of the *second kind* will be mentioned and references where appropriate.

The Raviart-Thomas-Nédélec $H(\mathbf{curl})$ elements on simplices. To define the Nédélec elements of the first kind on simplices, denoted as RT_q^e and $\mathrm{N1}_q^e$ on triangles/tetrahedra respectively, we first define the set

$$S_q(K) = \{\boldsymbol{s} \in [\mathcal{P}_q(K)]^d : \boldsymbol{s} \cdot \boldsymbol{x} = 0, \forall \boldsymbol{x} \in K\}. \tag{2.147}$$

Then, for $d = 2$ and $d = 3$, we take V to be

$$V = [\mathcal{P}_q(K)]^d + S_q(K). \tag{2.148}$$

The nodal degrees of freedom for RT_q^e in two-dimensions are given as the set

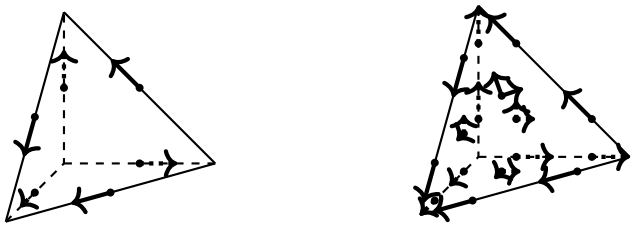
$$\mathcal{N}_{\mathrm{RT}_q^e} = \begin{cases} \int_e \boldsymbol{v} \cdot \boldsymbol{t} \psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(e), \forall \text{ edges } e, \\ \int_K \boldsymbol{v} \cdot \boldsymbol{\phi} \mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in [\mathcal{P}_{q-2}(K)]^2, q \geq 2, \end{cases} \tag{2.149}$$

and in three-dimensions for $\mathrm{N1}_q^e$:

$$\mathcal{N}_{\mathrm{N1}_q^e} = \begin{cases} \int_e \boldsymbol{v} \cdot \boldsymbol{t} \psi \mathrm{d}l, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(e), \forall \text{ edges } e, \\ \int_f (\boldsymbol{v} \times \boldsymbol{n}) \cdot \boldsymbol{\psi} \mathrm{d}s, & \forall \text{ basis functions } \boldsymbol{\psi} \in [\mathcal{P}_{q-2}(K)]^2, \forall \text{ faces } f, \\ & q \geq 2, \\ \int_K \boldsymbol{v} \cdot \boldsymbol{\phi} \mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in [\mathcal{P}_{q-2}(K)]^3, q \geq 3. \end{cases} \tag{2.150}$$

Here, $\boldsymbol{t}$ denotes the vector which is tangent to each edge e of K. Lowest- and next-to-lowest order elements are illustrated in Figure 2.10. These are also interpreted as *rotated* (by 90° along edges/faces) Raviart-Thomas elements. The dimension of $\mathrm{RT}_q^e/\mathrm{N1}_q^e$ are $q\,(q+2)$ and $\frac{1}{2}q\,(q+2)\,(q+3)$ respectively.

(A) RT_1^e and RT_2^e finite elements on triangular cells.



(B) $\mathrm{N1}_1^e$ and $\mathrm{N1}_2^e$ finite elements on tetrahedral cells.

FIGURE 2.10: The lowest-order and next-to-lowest-order $H(\mathrm{curl})$ elements on simplicial cells. The arrows along cell edges denote tangential component evaluations, crossed arrows on the faces (in three-dimensions) denotes evaluating the tangential components on faces, and the large shaded circles denote interior moment evaluations. The numerical values near the shaded circles display the total number of interior degrees of freedom.

We remark here that the construction of the Nédélec elements of the second kind are constructed by rotating BDM_q finite elements. As a result, there is also a two-dimensional $H(\mathrm{curl})$ element derived from BDM_q, written as BDM_q^e. The construction for both simplex and quadrilateral/cube cells are presented in (Nédélec, 1980; Nédélec, 1986; Brezzi et al., 1987a).

The Raviart-Thomas-Nédélec $H(\mathrm{curl})$ elements on cuboids. The construction of $H(\mathrm{curl})$ on quadrilaterals and cubes, written as RTC_q^e and NC_q^e respectively, follows similarly to the construction of the RTC_q^f and NC_q^f families. The function space V on K is as defined in (2.142). The main difference lies in the construction of the degrees of freedom, where tangential components are used in place of normal components. In two-dimensions, we have

$$\mathcal{N}_{\mathrm{RTC}_q^e} = \begin{cases} \int_e \boldsymbol{v}\cdot\boldsymbol{t}\psi \mathrm{d}s, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(e), \forall \text{ edges } e, \\ \int_K \boldsymbol{v}\cdot\boldsymbol{\phi}\mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in \mathrm{RTC}_{q-2}^f(K), q \geq 2, \end{cases} \tag{2.151}$$

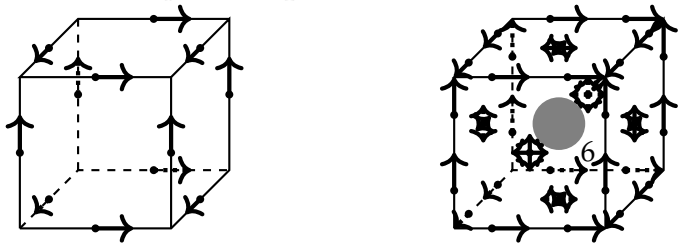
and in three-dimensions:

$$\mathcal{N}_{\mathrm{NC}_q^e} = \begin{cases} \int_e \boldsymbol{v}\cdot\boldsymbol{t}\psi \mathrm{d}l, & \forall \text{ basis functions } \psi \in \mathcal{P}_{q-1}(e), \forall \text{ edges } e, \\ \int_f (\boldsymbol{v}\times\boldsymbol{n})\cdot\boldsymbol{\psi}\mathrm{d}s, & \forall \text{ basis functions } \boldsymbol{\psi} \in \mathrm{RTC}_{q-2}^f(f), \forall \text{ faces } f, \\ & q \geq 2, \\ \int_K \boldsymbol{v}\cdot\boldsymbol{\phi}\mathrm{d}x, & \forall \text{ basis functions } \boldsymbol{\phi} \in \mathrm{NC}_{q-2}^f(K), q \geq 2. \end{cases} \tag{2.152}$$

Dimensions of the finite elements are $2(q+1)(q+2)$ and $3q(q+1)^2$ respectively. Element diagrams are provided in Figure 2.11.

The covariant Piola transform. As with the $H(\mathrm{div})$ case, the pullback derived only from the change-of-coordinates map F_K is not sufficient to preserve tangential components when mapping between reference and physical cells. The *covariant Piola*

(A) RTC_1^e and RTC_2^e elements on quadrilaterals.



(B) NC_1^e and NC_2^e elements on cubes.

FIGURE 2.11: The lowest-order and next-to-lowest-order $H(\mathrm{curl})$ elements on quadrilateral cells.

transform is used instead. With $J_K = DF_K$ denoting the Jacobian, we define the covariant Piola transform of $\boldsymbol{v}(\boldsymbol{x})$ from a field $\hat{\boldsymbol{v}}(\hat{\boldsymbol{x}})$ on the reference cell to be

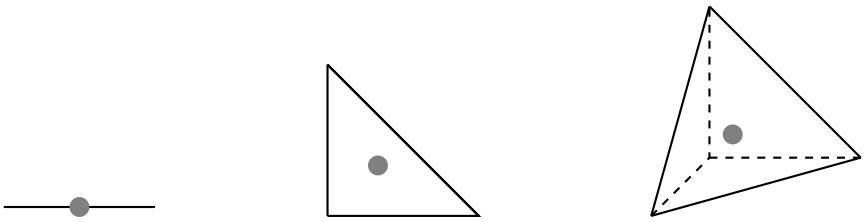
$$\boldsymbol{v}(\boldsymbol{x}) = J_K^{-T}\boldsymbol{v}(F_K(\hat{\boldsymbol{x}})) = J_K^{-T}\hat{\boldsymbol{v}}(\hat{\boldsymbol{x}}) =: \mathcal{F}_{\mathrm{curl}}(\hat{\boldsymbol{v}}). \tag{2.153}$$

More information on Piola transformations and pullbacks for $H(\mathrm{div})$ and $H(\mathrm{curl})$ can be found in (Boffi, Brezzi, and Fortin, 2013; Rognes, Kirby, and Logg, 2009).

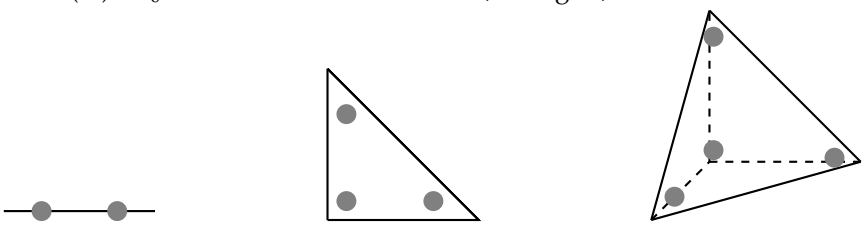
L^2 finite elements

By an L^2-finite element, we mean a finite element space of functions which are not globally C^0-continuous. This commonly appears in mixed formulations of elliptic PDEs, as well as within our context of simulating geophysical flows. Historically, L^2 elements are used in Galerkin methods where continuity is imposed weakly (typically through numerical flux terms appearing on element boundaries) instead of directly via the finite element space. A canonical example is the discontinuous Galerkin (DG) method for elliptic equations (Arnold, 1982; Arnold et al., 2002).

Discontinuous Lagrange elements on simplices and cuboids. Here, we present the discontinuous variants of the Lagrange elements P_q and Q_q as defined earlier. Often called the *discontinuous Lagrange* element, dP_q, the construction is identical to its continuous counterpart. The same holds for the discontinuous quadrilateral/cube element: dQ_q. The main difference is that all degrees of freedom (as defined in (2.133) and (2.137)) are *topologically* associated with the cell (see Figure 2.12 for illustrations of dP_q and Figure 2.13 for dQ_q); there are no constraints for the local polynomial functions to be continuous on cell vertices, edges, or faces. Additionally, the same pullback definition in (2.135) applies to the dP_q and dQ_q elements.

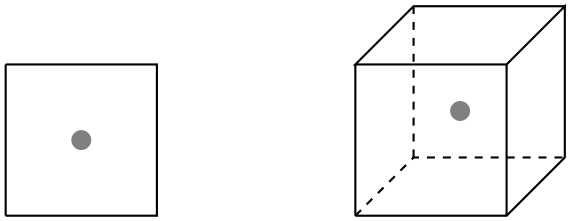


(A) dP_0 finite elements on intervals, triangles, and tetrahedra.

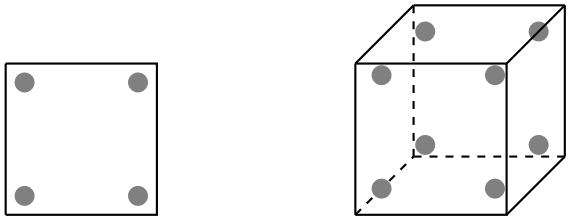


(B) dP_1 finite elements on intervals, triangles, and tetrahedra.

FIGURE 2.12: Constant and linear discontinuous Lagrange finite elements on simplicial cells. Gray discs denote degrees of freedom which are associated with the cell only.



(A) dQ_0 finite elements on quadrilateral and cube cells.



(B) dQ_1 finite elements on quadrilateral and cube cells.

FIGURE 2.13: Constant and linear discontinuous Lagrange finite elements on cuboid cells.

2.3 Finite element computations

Having defined various classes of finite elements, it will be useful to quickly go over how finite element computations are performed within numerical code. As we have previously mentioned in Section 2.2.1, computations for finite elements need only an appropriate pullback to a reference cell $\hat{K}$. For affine-equivalent elements, a single reference finite element $(\hat{K}, \hat{V}, \hat{\mathcal{N}})$ is all we require. The same applies for the Piola-mapped $H(\mathrm{div})/H(\mathrm{curl})$ elements as well (Rognes, Kirby, and Logg, 2009). An analogous result of affine-equivalence for isoparametric finite elements (elements defined on curved domains) is presented by Brenner and Scott (2008, §4.7). A more general theory for mapping finite elements, which includes affine and non-affine transformations, to reference elements is discussed by Kirby (2018). At any rate, the theory presented thus far still applies for more generally mapped finite elements.

Suppose we have a computational domain Ω and a mesh $\mathcal{T}_h$ of Ω consisting of cells

K. Using the weak formulation from Section 2.2.1 as our example (see equation (2.132)), let us take $V_{h,0} \subset H_0^1$ to be the continuous Lagrange finite element space of order q satisfying the homogenous Dirichlet condition $u_h = 0$ on $\Gamma_D = \{\partial K : K \in \mathcal{T}_h\} \cap \partial\Omega_D$. Then the *discrete* finite element problem reads as follows: find $u_h \in V_{h,0}$ such that

$$a_h(u_h, v) := \int_{\mathcal{T}_h} \nabla_h u_h \cdot \nabla_h v + u_h v \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} fv \,\mathrm{d}\boldsymbol{x} + \int_{\Gamma_N} gv \,\mathrm{ds} := L_h(v), \tag{2.154}$$

for all $v \in V_{h,0}$, where $\Gamma_N = \{\partial K : K \in \mathcal{T}_h\} \cap \partial\Omega_N$ and ∇_h denotes the cell-wise gradient: $\nabla_h|_K = \nabla|_K$. Note that since $V_{h,0} \subset H_0^1$, the integral containing gradients of the *trial function* u_h and *test function* v is legal. Equation (2.154) can best be understood as a *Galerkin projection* from H_0^1 onto the finite-dimensional space $V_{h,0}$. For this reason, methods which involve testing the equation with functions in the same space as the solution variable are often called *Galerkin methods*.

Then, expanding $u_h = \sum_i u_i \Phi_i$ into its global finite element basis representation, taking $v = \Phi_j$, and using the fact that $a_h(u_h, v)$ is *linear* in the solution variable u_h, we arrive at a discrete $N \times N$ (where $N = \dim V_{h,0}$) matrix system for the nodal coefficients u_i:

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F}, \tag{2.155}$$

where $\boldsymbol{U} = (u_1, \cdots, u_N)$, and

$$\begin{aligned} \boldsymbol{K}_{ij} = a_h(\Phi_i, \Phi_j) &= \int_{\mathcal{T}_h} \nabla_h \Phi_i \cdot \nabla_h \Phi_j + \Phi_i \Phi_j \,\mathrm{d}\boldsymbol{x} \\ &= \sum_{K \in \mathcal{T}_h} \int_K \nabla \Phi_i \cdot \nabla \Phi_j + \Phi_i \Phi_j \,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{2.156}$$

$$\boldsymbol{F}_j = L_h(\Phi_j) = \int_{\mathcal{T}_h} f \Phi_j \,\mathrm{d}\boldsymbol{x} = \sum_{K \in \mathcal{T}_h} \int_K f \Phi_j \,\mathrm{d}\boldsymbol{x}. \tag{2.157}$$

2.3.1 Evaluating finite element forms

To evaluate $\boldsymbol{K}$ and $\boldsymbol{F}$, we first map the element-wise contributions of (2.156) and (2.157) to the reference element $\hat{K}$. Let J_K be the Jacobian of the change-of-coordinates transformation F_K, with $\boldsymbol{x} = F_K(\hat{\boldsymbol{x}})$. With $\widehat{\Phi}$ denoting scalar functions defined on $\hat{K}$, we can summarize the relationship between functions in physical space and functions on $\hat{K}$ via:

$$\Phi_i(\boldsymbol{x}) = \Phi_i(F_K(\hat{\boldsymbol{x}})) = \widehat{\Phi}_i(\hat{\boldsymbol{x}}) =: \mathcal{F}(\widehat{\Phi}_i). \tag{2.158}$$

Applying (2.158) and the chain-rule, we can easily show that:

$$\nabla \Phi_i(\boldsymbol{x}) = J_K^{-T} \widehat{\nabla} \widehat{\Phi}_i(\hat{\boldsymbol{x}}), \tag{2.159}$$

where $\widehat{\nabla}$ denotes the gradient with respect to the reference coordinates $\hat{\boldsymbol{x}}$ and $J_K^{-T} = \left(J_K^{-1}\right)^T$. Putting everything together and applying the usual change-of-coordinates rules for integration, we can write the element-wise contribution of $\boldsymbol{K}$ as:

$$\boldsymbol{K}_{K,ij} := \int_K \nabla \Phi_i \cdot \nabla \Phi_j + \Phi_i \Phi_j \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \left(J_K^{-T} \widehat{\nabla} \widehat{\Phi}_i \cdot J_K^{-T} \widehat{\nabla} \widehat{\Phi}_j + \widehat{\Phi}_i \widehat{\Phi}_j \right) |J_K| \mathrm{d}\hat{\boldsymbol{x}}. \tag{2.160}$$

Similarly, the local contribution of $\boldsymbol{F}$ is

$$\boldsymbol{F}_{K,j} := \int_K f\Phi_j \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} f(F_K(\hat{\boldsymbol{x}}))\widehat{\Phi}_j |J_K| \mathrm{d}\hat{\boldsymbol{x}}. \tag{2.161}$$

Remark 2. *(The cell Jacobian J_K): For an affine mapping F_K, the Jacobian will be constant over each cell K. However, for non-affine transformations, such as the case for isoparametric or tensor product finite elements, then J_K will vary as a function of $\hat{\boldsymbol{x}}$. In either case, the discussion here still applies. A non-affine F_K only affects our numerical evaluation of the finite element forms, which may result in integrating non-polynomial expressions.*

Evaluating the local contributions of finite elements forms on the reference cell $\hat{K}$ is the bread and butter of finite element codes. In order to evaluate (2.160) and (2.161) numerically, we require a suitable *quadrature method*. For integrating some function, say $f(\hat{\boldsymbol{x}})$, over $\hat{K}$, an n-point quadrature method has the form

$$\int_{\hat{K}} f(\hat{\boldsymbol{x}})\mathrm{d}\hat{\boldsymbol{x}} \approx \sum_{q=1}^{n} f(\hat{\boldsymbol{x}}_q) w_q \tag{2.162}$$

where $\{\hat{\boldsymbol{x}}_q\}_{q=1}^n$ are *quadrature points* with associated *quadrature weights* $\{w_q\}_{q=1}^n$. A scheme comprised of a set of quadrature points and weights is called a *quadrature rule*. Typically Guassian quadrature rules are employed in finite element codes, as well as its variations such as the Gauss-Lobatto (GL) or Gauss-Lobatto-Legendre (GLL) methods (Davis and Rabinowitz, 1984).

It is easy to show that, if f is polynomial in $\hat{\boldsymbol{x}}$ with degree $q \leq n-2$, then using an n-point Gaussian quadrature rule is *exact* (Süli and Mayers, 2003). We say a quadrature rule is a q-degree quadrature method if it integrates polynomials of degree $\leq q$ exactly. Since finite element methods frequently involve integrating polynomial expressions, most integrals could be evaluated exactly by using an appropriate quadrature method.

Remark 3. *(Exact vs inexact quadrature): Typically, higher degree quadrature rules have more quadrature points, hence requiring more function evaluations than lower degree methods. This results in a trade-off between the accuracy of the quadrature scheme and the computational cost of an integration using that rule. Exact quadrature results in lower errors, but if the error due to inexact quadrature is small compared with other errors (such as discretization error), then inexact quadrature may be advantageous. This must be handled with care, as inexact quadrature can result in a general loss of stability as well. Analysis of this falls into the category of "variational crimes" (Strang, 1973).*

After applying a quadrature method, (2.160) and (2.161) become:

$$\boldsymbol{K}_{K,ij} \approxeq \sum_q \left(J_K(\hat{\boldsymbol{x}}_q)^{-T} \widehat{\nabla}\widehat{\Phi}_i(\hat{\boldsymbol{x}}_q) \cdot J_K(\hat{\boldsymbol{x}}_q)^{-T} \widehat{\nabla}\widehat{\Phi}_j + \widehat{\Phi}_i(\hat{\boldsymbol{x}}_q)\widehat{\Phi}_j(\hat{\boldsymbol{x}}_q) \right) |J_K(\hat{\boldsymbol{x}}_q)| w_q, \tag{2.163}$$

$$\boldsymbol{F}_{K,j} \approxeq \sum_q f(F_K(\hat{\boldsymbol{x}}_q))\widehat{\Phi}_j(\hat{\boldsymbol{x}}_q)|J_K(\hat{\boldsymbol{x}}_q)|w_q. \tag{2.164}$$

Note that different degree quadrature rules can be used on different integral forms. Some finite element packages, like Firedrake (Rathgeber et al., 2017), employ sophisticated compiler technology which determines an appropriate quadrature rule using encoded information about the finite element space. Within the context of Firedrake, a more in-depth discussion on determining optimal order quadrature methods is

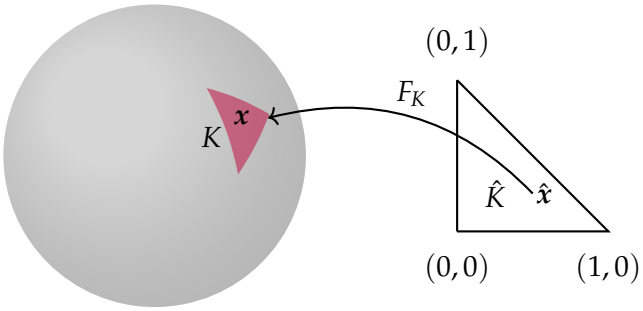


FIGURE 2.14: Change-of-coordinates transformation F_K from the reference triangle to a physical triangle on the surface of a sphere. Coordinates in reference space are mapped to physical space via $\boldsymbol{x} = F_K(\hat{\boldsymbol{x}})$.

presented by Homolya et al. (2018) and Homolya, Kirby, and Ham (2017). More information on Firedrake is provided in Chapter 4.

2.3.2 Finite element computations on immersed manifolds

In this section, we discuss the special case where we want to approximate a PDE on a smooth m-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^n$, $1 \leq m < n$. This is particularly relevant for this dissertation, as we will be solving geophysical equations on the surface of a sphere, a two-dimensional surface embedded in three-dimensions. Therefore, we must introduce relevant material for how such finite element computations are performed. We proceed using a similar presentation to that of Rognes et al. (2013).

We first let $\boldsymbol{x} = (x_1, \cdots, x_n)$ be the coordinates in the *physical* or *geometric* dimension, and $\hat{\boldsymbol{x}} = (\hat{x}_1, \cdots, \hat{x}_m)$ be coordinates in the *manifold* or *topological* dimension. We approximate the manifold $\mathcal{M}$ by some tessellation of cells $\mathcal{T}_h = \{K\}$. That is, each cell K will have topological dimension m and geometric dimension n. We define a fixed reference cell $\hat{K}$ and assume there exists a transformation $F_K : \mathbb{R}^m \to \mathbb{R}^n$ with $K = F_K(\hat{K})$, as illustrated (for triangles) in Figure 2.14. Then the cell Jacobian of the mapping F_K is the $n \times m$ matrix:

$$J_K = \frac{\partial F_K(\hat{\boldsymbol{x}})}{\partial \hat{\boldsymbol{x}}} = \frac{\partial \boldsymbol{x}}{\partial \hat{\boldsymbol{x}}} = \begin{bmatrix} \frac{\partial x_1}{\partial \hat{x}_1} & \frac{\partial x_1}{\partial \hat{x}_2} & \cdots & \frac{\partial x_1}{\partial \hat{x}_m} \\ \frac{\partial x_2}{\partial \hat{x}_1} & \frac{\partial x_2}{\partial \hat{x}_2} & \cdots & \frac{\partial x_2}{\partial \hat{x}_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial \hat{x}_1} & \frac{\partial x_n}{\partial \hat{x}_2} & \cdots & \frac{\partial x_n}{\partial \hat{x}_m} \end{bmatrix}. \tag{2.165}$$

Remark 4. *(A variational crime): In our previous definition of a mesh from Section 2.2.1, we stated the condition that $\bigcup_i K_i = \overline{\Omega}$, for some domain Ω. In the case of an arbitrary manifold, this condition is not always satisfied. For example, consider the case where $\mathcal{M}$ is the surface of a sphere embedded in $\mathbb{R}^3$. Then a piecewise linear tessellation of triangles $\mathcal{T}_h = \{K\}$ breaks this condition:*

$$\bigcup_i K_i \neq \mathcal{M}. \tag{2.166}$$

In this case, $\bigcup_i K_i$ is simply an approximation of $\mathcal{M}$. Then, if $V = H^1(\mathcal{M})$ and $V_h(\mathcal{T}_h)$ is a continuous Lagrange space, we are in a situation where $V_h \not\subset V$. If we are trying to

solve some weak formulation on $\mathcal{M}$ using an H^1 finite element method, we have just broken a crucial rule of traditional Galerkin methods.

This is known as a "variational crime," since functions in V are being approximated by functions in V_h, defined on a different domain. Strang (1973) coined the term "variational crime," and quantified the errors introduced by these "crimes." Brenner and Scott (2008, §10) and Holst and Stern (2012) are useful resources on this topic. In real-world applications, committing these variational crimes are often deemed a practical necessity (due to a finite amount of computational resources).

The *pseudo-determinant* of J_K represents the transformation of a differential volume element on the manifold. For one-dimensional manifolds, $J_K = [J_1]$ and its pseudo-determinant is simply the length of the single column vector J_1 in the Euclidean 2-norm: $|J_K| = \|J_1\|_2$. In general, we define the pseudo-determinant to be

$$|J_K| := \sqrt{|\mathcal{G}|}, \tag{2.167}$$

where $\mathcal{G} = J_K^T J_K$ is the Gram matrix (Horn and Johnson, 2012, §7.2), and $|\mathcal{G}|$ is the determinant of $\mathcal{G}$. In differential geometry, (2.167) defines the volume of an m-dimensional parallelepiped, embedded in n-dimensions, spanned by the columns of the Jacobian J_K. Notice that when $m = n$, then (2.167) reduces to the standard determinant.

Most of the rules for pulling back finite element forms to the reference cell, as discussed in Section 2.3.1, still applies. For example, we have for scalar functions Φ_i, Φ_j:

$$\int_K \Phi_i(\boldsymbol{x})\Phi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \widehat{\Phi}_i(\hat{\boldsymbol{x}})\widehat{\Phi}_j(\hat{\boldsymbol{x}})|J_K|\mathrm{d}\hat{\boldsymbol{x}}, \tag{2.168}$$

where $\Phi_i(\boldsymbol{x}) = \mathcal{F}(\widehat{\Phi}_i)$ from (2.158), and $|J_K|$ from (2.167). Note that this can be extended to vector-valued functions on the manifold (whose function space is the Cartesian product of scalar-valued finite element spaces) by applying the standard change-of-variables to each scalar component of the vector.

Differential operators on manifolds

Most finite element integrals will involve derivatives of basis functions defined over the cells K. Therefore, we must be able to evaluate derivatives of functions defined on the manifold. Now, suppose we have a function $\Phi(\boldsymbol{x})$ defined the cell $K \subset \mathbb{R}^n$ with pullback $\widehat{\Phi}(\hat{\boldsymbol{x}})$ defined on $\hat{K} \subset \mathbb{R}^m$. The gradient $\widehat{\nabla}$ of $\widehat{\Phi}$ in the reference space is simply

$$[\widehat{\nabla}\widehat{\Phi}(\hat{\boldsymbol{x}})]_i = \frac{\partial\widehat{\Phi}(\hat{\boldsymbol{x}})}{\partial\hat{x}_i}, i = 1, \cdots, m. \tag{2.169}$$

We define the *tangent space* $T(K)$ of K to be the image of J_K. Thus, for any $\boldsymbol{v} \in T(K)$, we have $\boldsymbol{v} = J_K\hat{\boldsymbol{v}}$, where $\hat{\boldsymbol{v}}$ is some vector defined in the reference space. Then for any $\boldsymbol{v} \in T(K)$, the gradient of $\Phi(\boldsymbol{x})$ in the direction of $\boldsymbol{v}$ can be defined through the usual Gâteux derivative:

$$\nabla\Phi(\boldsymbol{x}) \cdot \boldsymbol{v} = \lim_{\epsilon\to 0} \frac{\Phi(\boldsymbol{x} + \epsilon\boldsymbol{v}) - \Phi(\boldsymbol{x})}{\epsilon}. \tag{2.170}$$

Assuming that the transformation F_K is non-degenerate, such that the columns of J_K are linearly independent, we have for any $\boldsymbol{v} \in T(K)$:

$$\begin{aligned}\nabla\Phi(\boldsymbol{x}) \cdot \boldsymbol{v} &= \lim_{\epsilon\to 0} \frac{\Phi(\boldsymbol{x}+\epsilon\boldsymbol{v}) - \Phi(\boldsymbol{x})}{\epsilon} \\ &= \lim_{\epsilon\to 0} \frac{\Phi(F_K(\hat{\boldsymbol{x}}+\epsilon\hat{\boldsymbol{v}})) - \Phi(F_K(\hat{\boldsymbol{x}}))}{\epsilon} \\ &= \lim_{\epsilon\to 0} \frac{\widehat{\Phi}(\hat{\boldsymbol{x}}+\epsilon\hat{\boldsymbol{v}}) - \widehat{\Phi}(\hat{\boldsymbol{x}})}{\epsilon} = \widehat{\nabla}\widehat{\Phi}(\hat{\boldsymbol{x}}) \cdot \hat{\boldsymbol{v}}. \end{aligned} \tag{2.171}$$

Next, we introduce the (left) pseudo-inverse of the cell Jacobian J_K, as defined by Penrose (1955):

$$J_K^+ = \left(J_K^T J_K\right)^{-1} J_K^T. \tag{2.172}$$

Then for $\boldsymbol{v} = J_K\hat{\boldsymbol{v}}$, we can easily see that $J_K^+\boldsymbol{v} = \hat{\boldsymbol{v}}$ by direct computation:

$$J_K^+\boldsymbol{v} = J_K^+ J_K\hat{\boldsymbol{v}} = \left(J_K^T J_K\right)^{-1}\left(J_K^T J_K\right)\hat{\boldsymbol{v}} = \hat{\boldsymbol{v}}. \tag{2.173}$$

After substituting (2.173) into (2.171), we see that

$$\nabla\Phi(\boldsymbol{x}) \cdot \boldsymbol{v} = \widehat{\nabla}\widehat{\Phi}(\hat{\boldsymbol{x}}) \cdot J_K^+\boldsymbol{v} = \left(J_K^+\right)^T \widehat{\nabla}\widehat{\Phi}(\hat{\boldsymbol{x}}) \cdot \boldsymbol{v}. \tag{2.174}$$

Notice that $\Phi(\boldsymbol{x})$ is an n-vector in the same physical space where the manifold is immersed. Moreover, $\left(J_K^+\right)^T = J_K\left(J_K^T J_K\right)^{-T} = J_K\left(J_K^T J_K\right)^{-1}$ maps to the tangent space $T(K)$. So from this we can clearly see that $\nabla\Phi(\boldsymbol{x})$ is in the tangent space, as we would expect. Since (2.174) holds for all $\boldsymbol{v} \in T(K)$, we obtain the following relation:

$$\nabla\Phi(\boldsymbol{x}) = \left(J_K^+\right)^T \widehat{\nabla}\widehat{\Phi}(\hat{\boldsymbol{x}}). \tag{2.175}$$

Putting everything together, we can now pullback finite element forms like the weak Laplacian operator:

$$\int_K \nabla\Phi_i \cdot \nabla\Phi_j \,\mathrm{d}\boldsymbol{x}, \tag{2.176}$$

where Φ_i and Φ_j are a pair of finite element basis functions defined in the cell K. Applying (2.175) and the standard rules for changing variables gives:

$$\int_K \nabla\Phi_i \cdot \nabla\Phi_j \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \left(J_K^+\right)^T \widehat{\nabla}\widehat{\Phi}_i \cdot \left(J_K^+\right)^T \widehat{\nabla}\widehat{\Phi}_j |J_K| \mathrm{d}\hat{\boldsymbol{x}}. \tag{2.177}$$

Piola-mapped finite elements on manifolds

Formally, the definition of $\nabla\cdot$ and $\nabla\times$ on a manifold $\mathcal{M}$ is summarized as the limit of flux and circulation integrals:

$$\nabla \cdot \boldsymbol{u}(\boldsymbol{x}) = \lim_{\epsilon\to 0} \frac{1}{|C(\epsilon)|} \oint_{C(\epsilon)} \boldsymbol{u} \cdot \boldsymbol{n} \,\mathrm{d}s, \tag{2.178}$$

$$(\nabla \times \boldsymbol{u}(\boldsymbol{x})) \cdot \boldsymbol{n} = \lim_{\epsilon\to 0} \frac{1}{|C(\epsilon)|} \oint_{C(\epsilon)} \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{s}, \tag{2.179}$$

where $C(\epsilon)$ is an close loop centered at a point $\boldsymbol{x}$ approaching radius ϵ as $\epsilon \to 0$, $\boldsymbol{n}$ is the outward normal along the boundary of $C(\epsilon)$, and $|C(\epsilon)|$ is the area on $\mathcal{M}$ enclosed by $C(\epsilon)$. For notational purposes, we denote the divergence and curl in reference space as $\widehat{\nabla} \cdot \hat{\boldsymbol{u}}$ and $\widehat{\nabla} \times \hat{\boldsymbol{u}}$ respectively, where $\widehat{\nabla}$ denotes the m-vector of partial derivatives in $\hat{\boldsymbol{x}}$.

As previously discussed in Section 2.2.2, $H(\mathrm{div})$ finite element spaces (such as the Raviart-Thomas-Nédélec or Brezzi-Douglas-Marini spaces) consist of vector fields $\boldsymbol{u}$ whose pullbacks are defined via the contraviarant Piola transform. On the manifold, the definition remains largely unchanged. Letting $\widehat{\boldsymbol{\Psi}}$ be a vector function defined on the reference cell $\hat{K}$, we define $\boldsymbol{\Psi}$ on the physical cell K as

$$\boldsymbol{\Psi}(\boldsymbol{x}) = \mathcal{F}_{\mathrm{div}}(\widehat{\boldsymbol{\Psi}}) = \frac{1}{|J_K(\hat{\boldsymbol{x}})|} J_K \widehat{\boldsymbol{\Psi}}(\hat{\boldsymbol{x}}), \tag{2.180}$$

where $|J_K|$ is the Jacobian pseudo-determinate. Under the transformation (2.180), the divergence operator is pulled back via the relation:

$$\nabla \cdot \boldsymbol{\Psi}(\boldsymbol{x}) = \frac{1}{|J_K(\hat{\boldsymbol{x}})|} \widehat{\nabla} \cdot \widehat{\boldsymbol{\Psi}}(\hat{\boldsymbol{x}}). \tag{2.181}$$

In general, $\widehat{\boldsymbol{\Psi}}$ is a vector field with m components and $\mathcal{F}_{\mathrm{div}}(\widehat{\boldsymbol{\Psi}})$ is a vector with n components. Moreover, $\mathcal{F}_{\mathrm{div}}(\widehat{\boldsymbol{\Psi}}) \in T(K)$ by construction: $J_K \widehat{\boldsymbol{\Psi}} \in T(K)$. The sign of $|J_K|$ is positive if K has the same orientation as the manifold $\mathcal{M}$ and negative otherwise. If $\mathcal{M}$ is non-orientable, then (2.180) is not well-defined and thus the Piola transform cannot be applied. Fortunately, this will not be the case for us. For the implementation details of manifold orientation, we refer the reader to Rognes et al. (2013, §3.3.2).

A particularly useful property of the contravariant Piola transform is revealed when mapping between physical and reference cells. Integrals involving Piola-mapped quantities often result in *Jacobian cancellations*, which greatly simplifies integral evaluations. We summarize this with the following lemma from Boffi, Brezzi, and Fortin (2013, §2.1.3).

Lemma 1. *(Properties of the contravariant Piola transformation): For any (contravariant) Piola-mapped vector field $\boldsymbol{\Psi} = \mathcal{F}_{\mathrm{div}}(\widehat{\boldsymbol{\Psi}})$, $\boldsymbol{\Psi} \in H(\mathrm{div}; K)$, and scalar function $\Phi = \mathcal{F}(\widehat{\Phi})$, $\Phi \in H^1(K)$, the following relations hold:*

$$\int_K \nabla\Phi \cdot \boldsymbol{\Psi} \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \widehat{\nabla}\widehat{\Phi} \cdot \widehat{\boldsymbol{\Psi}} \mathrm{d}\hat{\boldsymbol{x}}, \tag{2.182}$$

$$\int_K \Phi\nabla \cdot \boldsymbol{\Psi} \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \widehat{\Phi}\widehat{\nabla} \cdot \widehat{\boldsymbol{\Psi}} \mathrm{d}\hat{\boldsymbol{x}}, \tag{2.183}$$

$$\int_{\partial K} \Phi\boldsymbol{\Psi} \cdot \boldsymbol{n} \,\mathrm{d}s = \int_{\partial \hat{K}} \widehat{\Phi}\widehat{\boldsymbol{\Psi}} \cdot \hat{\boldsymbol{n}} \mathrm{d}\hat{s}. \tag{2.184}$$

This well-known result is proven for affine F_K and extended to non-affine (curved elements) by Thomas (1976). While the identities (2.182)–(2.184) can be shown by direct calculation (similarly for immersed manifolds), albeit through tedious vector calculus computations, the result is more elegantly summarized within the framework of finite element exterior calculus (FEEC). We refer the interested reader to Arnold, Falk, and Winther (2006, §2.1), Arnold, Falk, and Winther (2010, §4), Arnold (2013) and Arnold, Boffi, and Bonizzoni (2015).

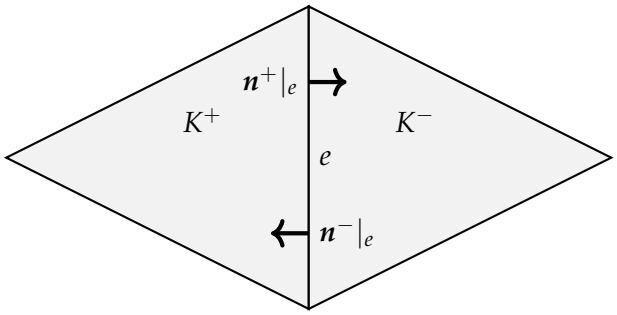


FIGURE 2.15: A shared edge between two adjacent elements. The $+$ and $-$ restrictions denote the different sides corresponding to the direction of the *outward* pointing normal vectors on the edge e.

The usefulness of the identities (2.183)–(2.184) come into play when considering $H(\mathrm{div}) \times L^2$ mixed finite element discretizations on $\mathcal{M}$. A commonly occurring integral for simulating geophysical flows is the pressure gradient term:

$$\int_{\mathcal{T}_h} \boldsymbol{\Psi} \cdot \nabla_h \Phi \, \mathrm{d}\boldsymbol{x}, \tag{2.185}$$

where $\boldsymbol{\Psi}$ and Φ are some vector and scalar fields respectively. Now, taking $\boldsymbol{\Psi} \in U_h$, and $\Phi \in V_h$, where $U_h \times V_h$ is an $H(\mathrm{div}) \times L^2$ pairing of finite element spaces (for example, Raviart-Thomas and discontinuous Lagrange spaces), we can integrate by parts (Green's formula) to obtain:

$$\begin{aligned}\int_{\mathcal{T}_h} \boldsymbol{\Psi} \cdot \nabla_h \Phi \, \mathrm{d}\boldsymbol{x} &= -\int_{\mathcal{T}_h} \Phi \nabla_h \cdot \boldsymbol{\Psi} \, \mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h^\partial} \Phi \boldsymbol{\Psi} \cdot \boldsymbol{n} \, \mathrm{d}s + \underbrace{\int_{\mathcal{E}_h^\circ} \Phi [\![\boldsymbol{\Psi}]\!] \, \mathrm{d}S}_{=0} \\ &= -\int_{\mathcal{T}_h} \Phi \nabla_h \cdot \boldsymbol{\Psi} \, \mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h^\partial} \Phi \boldsymbol{\Psi} \cdot \boldsymbol{n} \, \mathrm{d}s, \end{aligned} \tag{2.186}$$

where $\mathcal{E}_h^\partial$ is the set of exterior facets (empty if the manifold is closed), $\mathcal{E}_h^\circ$ is the set of interior facets, and $[\![\boldsymbol{\Psi}]\!]$ is the jump of $\boldsymbol{\Psi} \cdot \boldsymbol{n}$, defined as follows. On any shared facet e by two elements K^+ and K^-, the jump on e is defined as

$$[\![\boldsymbol{\Psi}]\!] = \boldsymbol{\Psi}|_{K^+} \cdot \boldsymbol{n}^+|_e + \boldsymbol{\Psi}|_{K^-} \cdot \boldsymbol{n}^-|_e, \tag{2.187}$$

where $\boldsymbol{n}^\pm$ denotes the outward-pointing normal on $\partial K^\pm$, as shown in Figure 2.15. Since $\boldsymbol{\Psi}$ comes from an $H(\mathrm{div})$ finite element space, the normal components are continuous across all interior facets, hence the vanishing surface term over $\mathcal{E}_h^\circ$.

Then, for $\boldsymbol{\Psi} = \mathcal{F}_{\mathrm{div}}(\widehat{\boldsymbol{\Psi}})$ and $\Phi = \mathcal{F}(\widehat{\Phi})$, and using (2.183)–(2.184), we can write the cellwise contribution of (2.186) as:

$$-\int_K \Phi \nabla \cdot \boldsymbol{\Psi} \, \mathrm{d}\boldsymbol{x} + \int_{\partial K} \Phi \boldsymbol{\Psi} \cdot \boldsymbol{n} \, \mathrm{d}s = -\int_{\hat{K}} \widehat{\Phi} \widehat{\nabla} \cdot \widehat{\boldsymbol{\Psi}} \mathrm{d}\hat{\boldsymbol{x}} + \int_{\partial \hat{K}} \widehat{\Phi} \widehat{\boldsymbol{\Psi}} \cdot \hat{\boldsymbol{n}} \mathrm{d}\hat{s}. \tag{2.188}$$

Equation (2.188) requires no cell Jacobian evaluations due to the cancellation properties of $\mathcal{F}_{\mathrm{div}}$. This is particularly nice, since otherwise integrals over curved domains will contain non-polynomial expressions after being pulled back to $\hat{K}$ (terms with $1/|J_K(\hat{\boldsymbol{x}})|$). This will make exact quadrature very difficult in this case. Moreover, failure to evaluate the expression (2.186) exactly can cause the discrete model to exhibit spurious computational modes (Cotter and Shipton, 2012). If this occurs, then a

solution with balanced initial condition oscillates rapidly. This is catastrophic when the nonlinear terms are introduced since the solution is polluted by rapid oscillations before the slow balanced state has had time to evolve; this renders a discretization for large-scale atmospheric modeling useless (Staniforth and Thuburn, 2011; Staniforth, Melvin, and Cotter, 2013).

Some integrals do not have such cancellation properties. For example, the mass term of two Piola-mapped vector fields on K becomes:

$$\int_K \boldsymbol{\Psi}_i \cdot \boldsymbol{\Psi}_j \,\mathrm{d}\boldsymbol{x} = \int_{\hat{K}} \frac{J_K \widehat{\boldsymbol{\Psi}_i}}{|J_K(\hat{\boldsymbol{x}})|} \cdot \frac{J_K \widehat{\boldsymbol{\Psi}_j}}{|J_K(\hat{\boldsymbol{x}})|} |J_K(\hat{\boldsymbol{x}})| \mathrm{d}\hat{\boldsymbol{x}} = \int_{\hat{K}} \frac{J_K \widehat{\boldsymbol{\Psi}_i} \cdot J_K \widehat{\boldsymbol{\Psi}_j}}{|J_K(\hat{\boldsymbol{x}})|} \mathrm{d}\hat{\boldsymbol{x}}, \tag{2.189}$$

which is non-polynomial whenever $|J_K(\hat{\boldsymbol{x}})|$ varies as a function of $\hat{\boldsymbol{x}}$. This makes exact integration for expressions like (2.189) very difficult. Fortunately, as mentioned by Cotter and Thuburn (2014), the key properties we desire in a geophysical fluid simulations (the basic energy conservation, balance, and wave propogation properties listed in Section 2.1.5), all rely on exactly integrating the following terms over a cell K (or facet e):

$$\int_K g\boldsymbol{w} \cdot \boldsymbol{u}^\perp \,\mathrm{d}\boldsymbol{x}, \quad \int_K \nabla g \cdot \boldsymbol{u} \,\mathrm{d}\boldsymbol{x}, \quad \int_K g \nabla \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x}, \quad \int_e g\boldsymbol{u} \cdot \boldsymbol{n} \,\mathrm{d}S, \tag{2.190}$$

where $\boldsymbol{u}$, $\boldsymbol{w}$ are Piola-mapped vectors and g is some arbitrary scalar function. All integrands in (2.190) were shown to have cancellations of Jacobian determinants when pulled back to the reference cell (or reference facet in the case of surface integrals). So while terms like (2.189) may appear in a discrete finite element formulation, selecting a quadrature rule of a suitable degree such that all terms in (2.190) are integrated exactly will be sufficient to maintain our desired properties. Direct calculations and analysis within a FEEC framework are provided by Cotter and Thuburn (2014).

Finally, we conclude this section by briefly mentioning how $H(\mathrm{curl})$ finite element functions are handled on manifolds. Similarly with pulling back $H(\mathrm{div})$ functions, the covariant Piola transform (see (2.153)) remains unchanged except for the inclusion of the Jacobian pseudo-inverse:

$$\boldsymbol{\xi}(\boldsymbol{x}) = \mathcal{F}_{\mathrm{curl}}(\widehat{\boldsymbol{\xi}}) = \left(J_K^+\right)^T \widehat{\boldsymbol{\xi}}(\hat{\boldsymbol{x}}), \tag{2.191}$$

where $\boldsymbol{\xi}$ and $\widehat{\boldsymbol{\xi}}$ are vector fields defined on K and $\hat{K}$ respectively. Under the transformation (2.191), we have the following relation for the curl operator:

$$\nabla \times \boldsymbol{\xi}(\boldsymbol{x}) = \frac{J_K}{|J_K(\hat{\boldsymbol{x}})|} \widehat{\nabla} \times \widehat{\boldsymbol{\xi}}(\hat{\boldsymbol{x}}). \tag{2.192}$$

It is worth noting that similar Jacobian cancellations occur for $H(\mathrm{curl})$ vector functions. However, since we do not make use of them in this dissertation, we omit the details and instead refer the interested reader to Boffi, Brezzi, and Fortin (2013, §2, Lemma 2.1.9), Nédélec (1980) and Nédélec (1986).

2.4 Compatible finite element methods

Over the past decade, finite element discretizations of the governing equations for atmospheric motion have become increasing popular. A majority of the focus has

been centered on high-order continuous Galerkin (CG), spectral element, and discontinuous Galerkin (DG) methods. For an overview, see Fournier, Taylor, and Tribbia (2004), Thomas and Loft (2005), Dennis et al. (2011), Kelly and Giraldo (2012), Giraldo, Kelly, and Constantinescu (2013), Bao, Klöfkorn, and Nair (2015), and Marras et al. (2015). More recently, a theory of mixed-finite element discretizations, known as mimetic or *compatible* finite element methods (Cotter and Shipton, 2012; Staniforth, Melvin, and Cotter, 2013; Cotter and Thuburn, 2014; McRae and Cotter, 2014; Natale, Shipton, and Cotter, 2016), has been developed within the context of simulating geophysical fluid dynamics.

Compatible finite element elements are built from discrete spaces that satisfy an L^2 de-Rham co-homology in which differential operators map from one space to another (Arnold, Falk, and Winther, 2006; Arnold, Falk, and Winther, 2010; Arnold and Awanou, 2014). Discretizations arising from this framework have a long history in both numerical analysis and applications ranging from structural mechanics to porous media flows. See Boffi et al. (2008) and Boffi, Brezzi, and Fortin (2013) for a summary of contributions.

The realization made by Cotter and Shipton (2012) that compatible finite element methods respect essential geostrophic balance relations for steady-state solutions of the rotating shallow water equations, combined with other properties making them analogous to the Arakawa C-grid (Thuburn, Cotter, and Dubos, 2014), has galvanized an effort to develop similar techniques for numerical weather prediction. In this section, we introduce key concepts which build the framework of compatible finite element methods for both two- and three-dimensional formulations.

2.4.1 L^2 de-Rham complexes

Let Ω be a bounded domain in $\mathbb{R}^n$. By a *de-Rham complex*, we mean a sequence of spaces with:

$$0 \to V^0(\Omega) \xrightarrow{\mathrm{d}^0} V^1(\Omega) \xrightarrow{\mathrm{d}^1} \cdots \xrightarrow{\mathrm{d}^{n-1}} V^n(\Omega) \to 0, \tag{2.193}$$

where V^k are function spaces on $\mathbb{R}^n$ and d^k are mappings between them satisfying $\mathrm{d}^k \circ \mathrm{d}^{k-1} = 0$. As soon as the function spaces V^k are defined explicitly, the operators d^k take the form of familiar differential operators from vector calculus.

For $k = 0, n$, the spaces V^k consist of scalar-valued functions, and vector-valued for $k = 1, \cdots, n-1$. If the V^k are Hilbert spaces, then (2.193) can be identified with an L^2 de-Rham complex. In $\mathbb{R}^3$, the de-Rham complex takes the form:

$$0 \to H^1 \xrightarrow{\nabla} H(\mathrm{curl}) \xrightarrow{\nabla\times} H(\mathrm{div}) \xrightarrow{\nabla\cdot} L^2 \to 0. \tag{2.194}$$

Note that the well-known vector calculus identities $\nabla \times (\nabla f) = 0$, for a scalar function, and $\nabla \cdot (\nabla \times \boldsymbol{v}) = 0$ for a vector field are encoded in the de-Rham complex via $\mathrm{d}^k \circ \mathrm{d}^{k-1} = 0$. In $\mathbb{R}^2$, there are two identifications one can make which results in two valid L^2 de-Rham complexes. We refer the reader to Arnold, Falk, and Winther (2010) for a discussion on how this identification is made.[4] For brevity, we

[4] This identification branches into *finite element exterior calculus*, which establishes the relationship between spaces of differential forms and finite element spaces. The two complexes in $\mathbb{R}^2$ are a result of selecting an identification of 1-forms with $C^\infty(\mathbb{R}^2)$. This is summarized in Table 10 of Arnold, Falk, and Winther (2010).

just present both complexes below. The first complex in $\mathbb{R}^2$ is:

$$0 \to H^1 \xrightarrow{\nabla^\perp} H(\text{div}) \xrightarrow{\nabla\cdot} L^2 \to 0, \tag{2.195}$$

and the second:

$$0 \to H^1 \xrightarrow{\nabla} H(\text{curl}) \xrightarrow{\nabla^\perp\cdot} L^2 \to 0. \tag{2.196}$$

In one-dimension, there is only the single complex:

$$0 \to H^1 \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}x}} L^2 \to 0. \tag{2.197}$$

For each L^2 complex in $\mathbb{R}^n$, $n \leq 3$, we can identify a discrete complex consisting of finite element subspaces $V_h^k \subset V^k$. Abstractly, a critical requirement of "compatibility" in our context is that the following diagram commutes:

$$\begin{array}{ccccccccc} V^0 & \xrightarrow{\mathrm{d}^0} & V^1 & \xrightarrow{\mathrm{d}^1} & \cdots & \xrightarrow{\mathrm{d}^{n-1}} & V^{n-1} & \xrightarrow{\mathrm{d}^{n-1}} & V^n \\ \downarrow{\scriptstyle \pi^0} & & \downarrow{\scriptstyle \pi^1} & & & & \downarrow{\scriptstyle \pi^{n-1}} & & \downarrow{\scriptstyle \pi^n} \\ V_h^0 & \xrightarrow{\mathrm{d}^0} & V_h^1 & \xrightarrow{\mathrm{d}^1} & \cdots & \xrightarrow{\mathrm{d}^{n-1}} & V_h^{n-1} & \xrightarrow{\mathrm{d}^{n-1}} & V_h^n \end{array} \tag{2.198}$$

where $V_h^k \subset V^k$ are finite element subspaces, and π^k, $k = 1, \cdots, n$ are bounded projection operators between V^k and V_h^k. We remark here that the property $V_h^k \subset V^k$ is not strictly necessary to construct a useful discrete complex; Holst and Stern (2012) handled the more general setting. We shall elaborate in Section 2.4.6.

2.4.2 Compatible finite elements in two-dimensions

We begin with a formal definition for what it means for a two-dimensional finite element discretization to be "compatible." Let $\mathcal{T}_h$ be a mesh of a bounded domain Ω in $\mathbb{R}^2$. Now suppose we have the following L^2 de-Rham complex

$$0 \to H^1 \xrightarrow{\nabla^\perp} H(\text{div}) \xrightarrow{\nabla\cdot} L^2 \to 0. \tag{2.199}$$

Now let $V_h^k(\mathcal{T}_h)$ be finite element subspaces of V^k. The spaces V_h^k are *compatible* if the following conditions are met:

1. $\nabla^\perp \psi \in V_h^1$, for all functions $\psi \in V_h^0$;
2. $\nabla \cdot \boldsymbol{v} \in V_h^2$, for all functions $\boldsymbol{v} \in V_h^1$; and
3. There exists bounded projections π^0, π^1, and π^2 such that the following diagram *commutes*:

$$\begin{array}{ccccc} H^1 & \xrightarrow{\nabla^\perp} & H(\text{div}) & \xrightarrow{\nabla\cdot} & L^2 \\ \downarrow{\scriptstyle \pi^0} & & \downarrow{\scriptstyle \pi^1} & & \downarrow{\scriptstyle \pi^2} \\ V_h^0 & \xrightarrow{\nabla^\perp} & V_h^1 & \xrightarrow{\nabla\cdot} & V_h^2 \end{array} \tag{2.200}$$

The use of these commuting diagrams is used to demonstrate stability and convergence of the finite element method, as discussed by Boffi, Brezzi, and Fortin (2013). The fact that it makes sense to act on functions in V_h^k via the differential operator d^k is

a direct result of $V_h^k \subset V^k$. Within the context of finite exterior calculus (Arnold, Falk, and Winther, 2006; Arnold, Falk, and Winther, 2010; Arnold and Awanou, 2014), this framework is translated into the language of differential forms, where proofs of the existence of the projections π^k are derived from. For clarity, we choose to stay within the context of vector calculus as much as possible.

If the domain Ω has a boundary $\partial\Omega$, then we define $\mathring{H}^1$ and $\mathring{V}_h^0$ as subspaces of H^1 and V_h^0 with vanishing trace on $\partial\Omega$, respectively. Similarly, we define $\mathring{H}(\mathrm{div})$ and $\mathring{V}_h^1$ as the subspaces of $H(\mathrm{div})$ and V_h^1 where the normal components of functions vanish on $\partial\Omega$. Then, the above commutative diagram (2.200) still holds with subspaces appropriately substituted (Natale, Shipton, and Cotter, 2016).

A nearly identical definition can be made for the two-dimensional complex in (2.196). For our purposes, we will only be working with compatible finite element spaces in (2.199). For completion, however, we give some explicit examples of compatible finite element spaces for *both* de-Rham complexes below.

For the first complex in (2.199), the following finite element spaces form a compatible complex for $q \geq 1$:

$$\mathrm{P}_q \xrightarrow{\nabla^\perp} \mathrm{RT}_q \xrightarrow{\nabla\cdot} \mathrm{dP}_{q-1}, \tag{2.201}$$

$$\mathrm{P}_{q+1} \xrightarrow{\nabla^\perp} \mathrm{BDM}_q \xrightarrow{\nabla\cdot} \mathrm{dP}_{q-1}, \tag{2.202}$$

$$\mathrm{Q}_q \xrightarrow{\nabla^\perp} \mathrm{RTC}_q^f \xrightarrow{\nabla\cdot} \mathrm{dQ}_{q-1}. \tag{2.203}$$

Another useful finite element complex relevant for numerical weather prediction was detailed by Cotter and Shipton (2012) on triangles:

$$\mathrm{P}_2 \oplus \mathrm{B}_3 \xrightarrow{\nabla^\perp} \mathrm{BDFM}_2 \xrightarrow{\nabla\cdot} \mathrm{dP}_1, \tag{2.204}$$

where $\mathrm{P}_2 \oplus \mathrm{B}_3$ is the space of quadratic functions enriched by a cubic "bubble" function (a cubic function vanishing on cell boundaries).

For the second complex (2.196), we have for $q \geq 1$:

$$\mathrm{P}_q \xrightarrow{\nabla} \mathrm{RT}_q^e \xrightarrow{\nabla^\perp\cdot} \mathrm{dP}_{q-1}, \tag{2.205}$$

$$\mathrm{P}_{q+1} \xrightarrow{\nabla} \mathrm{BDM}_q^e \xrightarrow{\nabla^\perp\cdot} \mathrm{dP}_{q-1}, \tag{2.206}$$

$$\mathrm{Q}_q \xrightarrow{\nabla} \mathrm{RTC}_q^e \xrightarrow{\nabla^\perp\cdot} \mathrm{dQ}_{q-1}, \tag{2.207}$$

where RT_q^e, BDM_q^e, and RTC_q^e are the "rotated" versions of RT_q, BDM_q, and RTC_q^f respectively for $H(\mathrm{curl})$. These are presented by Brezzi, Douglas, and Marini (1985) and Nédélec (1980) for simplices and Nédélec (1986) and Brezzi et al. (1987a) for quadrilaterals.

2.4.3 Tensor product elements

In preparation for building a three-dimensional finite element complex, it will be helpful to provide constructions of what are known as *tensor product elements*. In fact, we have already encountered one such element explicitly: the $\mathrm{Q}_q(K)$ family. It is the simplest tensor product element by construction, but other elements for the

various Sobolev spaces are indeed possible. Using tensor product elements, we can construct finite elements on more general polytopes, like triangular prisms.

Product cells and function spaces

Let $K_A \subset \mathbb{R}^n$ and $K_B \subset \mathbb{R}^m$ be two polygonal cells. We define the product of K_A and K_B, denoted as $K_A \times K_B$, to simply be the set:

$$K_A \times K_B = \{(x_1, \cdots, x_{n+m}) \in \mathbb{R}^{n+m} : (x_1, \cdots, x_n) \in K_A \text{ and } (x_{n+1}, \cdots x_{n+m}) \in K_B\}. \tag{2.208}$$

This construction can be applied to the product of an arbitrary number of cells $K_i \subset \mathbb{R}^{n_i}$, $i = 1, \cdots, k$:

$$\begin{aligned}\bigtimes_{i=1}^{k} K_i = \{(x_1, \cdots, x_{\sum_{i=1}^k n_i}) \in \mathbb{R}^{\sum_{i=1}^k n_i} : \\ (x_1, \cdots, x_{n_1}) \in \mathbb{R}^{n_1}, \\ (x_{n_1+1}, \cdots, x_{n_1+n_2}) \in \mathbb{R}^{n_2}, \\ \vdots \\ (x_{1+\sum_{i=1}^{k-1} n_i}, \cdots, x_{\sum_{i=1}^k n_i}) \in \mathbb{R}^{n_k}\}.\end{aligned} \tag{2.209}$$

This definition follows naturally, as one could take the product of cells where some of the component cells are tensor product cells themselves, such as cube/box cells.

Let $K = \bigtimes_{i=1}^k K_i$ be a product cell. Then the tensor product function space on K is simply

$$\bigotimes_{i=1}^{k} V_i(K_i) = \text{Span}\{v_1 \cdot v_2 \cdots v_k : v_i \in V_i(K_i)\}, \tag{2.210}$$

where

$$\begin{aligned}&(v_1 \cdot v_2 \cdots v_k)\left(x_1, \cdots, x_{\sum_{i=1}^k n_i}\right) = \\ &v_1(x_1, \cdots, x_{n_1}) \cdot v_2(x_{n_1+1}, \cdots x_{n_1+n_2}) \cdots v_k\left(x_{1+\sum_{i=1}^{k-1} n_i}, \cdots, x_{\sum_{i=1}^k n_i}\right).\end{aligned} \tag{2.211}$$

In our context, we typically construct product cells and tensor product spaces consisting of two cells (hence two function spaces). Namely, $K = K_A \times K_B$ and the resulting product space consists of functions of the form:

$$(v_A \cdot v_B)(x_1, \cdots, x_{n+m}) = v_A(x_1, \cdots, x_n) \cdot v_B(x_{n+1}, \cdots, x_{n+m}), \tag{2.212}$$

where $v_A \in V_A(K_A)$ and $v_B \in V_B(K_B)$. Note that in nearly all cases, at least one of the function spaces will consist of scalar-valued functions, meaning that the product $v_A \cdot v_B$ is unambiguous. Lastly, if $\{\phi_i\}_{i=1}^N$ and $\{\psi_i\}_{i=1}^M$ are bases for V_A and V_B respectively, then a basis for $V_A \otimes V_B$ is given by $\{\Psi_{ij}\}_{1 \le i \le N, 1 \le j \le M}$, where $\Psi_{ij} = \phi_i \cdot \psi_j$.

Focusing on product cells of the form: $K = K_A \times K_B$, where $K_A \subset \mathbb{R}^n$ and $K_B \subset \mathbb{R}^m$, the *topological entities* of K correspond to the product of topological entities of both K_A and K_B. Following the convention outlined by McRae et al. (2016), we label the entities of a cell (say, in $\mathbb{R}^n$) by their dimension (0 for vertices, 1 for edges, $\cdots$, $n-1$

for facets, and n for the cell). Then the topological entities of $K_A \times K_B$ are labeled as follows:

- $(0, 0)$: vertices of $K_A \times K_B$; the product of a vertex of K_A with a vertex of K_B.
- $(0, 1)$: edges of $K_A \times K_B$; the product of a vertex of K_A with an edge of K_B.
- $(1, 0)$: edges of $K_A \times K_B$; the product of an edge of K_A with a vertex of K_B.

$\vdots$

- $(n-1, m)$: facets of $K_A \times K_B$; the product of a facet of K_A with the cell of K_B.
- $(n, m-1)$: facets of $K_A \times K_B$; the product of the cell of K_A with a facet of K_B.
- (n, m): cell of $K_A \times K_B$; the product of the cell of K_A with the cell of K_B.

Using the notation of McRae et al. (2016), we can clearly distinguish between different types of entities, especially those with the same geometric dimension. For example, if K_A is a triangle and K_B an interval, then the product cell $K_A \times K_B$ is a triangular prism in $\mathbb{R}^3$ with triangular facets labeled as $(2, 0)$ and quadrilateral facets $(1, 1)$. This labeling convention allows software to exploit the geometric structure of product cells, resulting in efficient code-generation techniques for iterating data structures attached to meshes with tensor product cells (Bercea et al., 2016).

Nodes of a tensor product element

Now suppose we have two finite elements $\mathcal{FE}_A$ and $\mathcal{FE}_B$ defined by the Ciarlet triples $(K_A, V_A, \mathcal{N}_A)$ $(K_B, V_B, \mathcal{N}_B)$ respectively. The *tensor product element* $\mathcal{FE}_A \otimes \mathcal{FE}_B$ is defined as the Ciarlet triple $(K, V, \mathcal{N})$, where

$$K = K_A \times K_B, \tag{2.213}$$
$$V = V_A \otimes V_B, \tag{2.214}$$

and the nodes are given by:

$$\mathcal{N} = \{n_{i,j} = n_i^{(A)} \star n_j^{(B)} : n_i^{(A)} \in \mathcal{N}_A, n_j^{(B)} \in \mathcal{N}_B\}. \tag{2.215}$$

The operator $\star$ in (2.215) denotes the "product" of nodes from the dual bases $\mathcal{N}_A$ and $\mathcal{N}_B$. Within our context, both $\mathcal{N}_A$ and $\mathcal{N}_B$ will be *nodal* in the Kronecker delta sense (2.107), thus we can explicitly define $n_i^{(A)} \star n_j^{(B)}$ as the functional

$$\left(n_i^{(A)} \star n_j^{(B)}\right)(v_A \cdot v_B) = n_i^{(A)}(v_A) \cdot n_j^{(B)}(v_B). \tag{2.216}$$

Given a node $n_{i,j} \in \mathcal{N}$ and a basis function $\Psi_{k,l} \in V$, then

$$n_{i,j}(\Psi_{k,l}) = n_i^{(A)}(\phi_k) \cdot n_j^{(B)}(\psi_l) = \delta_{ik}\delta_{jl}. \tag{2.217}$$

Setting $\alpha = (i, j)$ and $\beta = (k, l)$ to be multi-indices associated with the nodes and basis functions defining the product element respectively, we have

$$n_\alpha(\Psi_\beta) = \delta_{\alpha\beta}. \tag{2.218}$$

The set $\mathcal{N}$ is a basis for V' and characterizes the degrees of freedom for $\mathcal{FE}_A \otimes \mathcal{FE}_B$. A similar construction holds for the tensor product of arbitrarily many finite elements. For more information on the construction of tensor product elements, we refer the reader to McRae et al. (2016).

2.4.4 Compatible finite elements in three-dimensions

Since there is only a single de-Rham complex in $\mathbb{R}^3$ based on the construction in Section 2.4.1, we can focus solely on the follow L^2 complex over Ω_h:

$$0 \to H^1 \xrightarrow{\nabla} H(\text{curl}) \xrightarrow{\nabla\times} H(\text{div}) \xrightarrow{\nabla\cdot} L^2 \to 0. \tag{2.219}$$

Let $V_h^0 \subset H^1$, $V_h^1 \subset H(\text{curl})$, $V_h^2 \subset H(\text{div})$ and $V_h^3 \subset L^2$ be finite element spaces. Then we say that the spaces V_h^k are *compatible* if:

1. $\nabla\psi \in V_h^1$ for all functions $\psi \in V_h^0$;
2. $\nabla \times \boldsymbol{w} \in V_h^2$ for all vectors $\boldsymbol{w} \in V_h^1$;
3. $\nabla \cdot \boldsymbol{v} \in V_h^3$ for all vectors $\boldsymbol{v} \in V_h^2$; and lastly,
4. There exists bounded projections π^0, π^1, π^2, and π^3 such that the diagram commutes:

$$\begin{array}{ccccccc} H^1 & \xrightarrow{\nabla} & H(\text{curl}) & \xrightarrow{\nabla\times} & H(\text{div}) & \xrightarrow{\nabla\times} & L^2 \\ \downarrow{\scriptstyle \pi^0} & & \downarrow{\scriptstyle \pi^1} & & \downarrow{\scriptstyle \pi^2} & & \downarrow{\scriptstyle \pi^3} \\ V_h^0 & \xrightarrow{\nabla} & V_h^1 & \xrightarrow{\nabla\times} & V_h^2 & \xrightarrow{\nabla\cdot} & V_h^3 \end{array} \tag{2.220}$$

This is a natural extension of the two-dimensional case in Section 2.4.2. The three-dimensional complex is summarized more abstractly (for simplicial, cubical, and curvilinear meshes) by Arnold, Falk, and Winther (2010), Arnold and Awanou (2014), and Arnold, Boffi, and Bonizzoni (2015).

As before, a three-dimensional compatible sequence can be constructed from the elements previously defined. The following finite elements form a compatible set of spaces for the L^2 complex in (2.220) for $q \geq 1$:

$$\mathrm{P}_q \xrightarrow{\nabla} \mathrm{N1}_q^e \xrightarrow{\nabla\times} \mathrm{N1}_q^f \xrightarrow{\nabla\cdot} \mathrm{dP}_{q-1}, \tag{2.221}$$

$$\mathrm{Q}_q \xrightarrow{\nabla} \mathrm{NC}_q^e \xrightarrow{\nabla\times} \mathrm{NC}_q^f \xrightarrow{\nabla\cdot} \mathrm{dQ}_{q-1}. \tag{2.222}$$

The first complex (2.221) is a standard sequence of elements defined on tetrahedral cells. The second complex, however, can be constructed in a different manner. In fact, (2.222) is the first *tensor product complex* we have introduced.

Tensor product complex. In one-dimension, we have the complex defined on an interval I:

$$0 \to H^1(I) \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}x}} L^2(I) \to 0. \tag{2.223}$$

The obvious choice of finite elements to form the discrete complex are the continuous and discontinuous Lagrange elements respectively:

$$\mathrm{P}_q(I) \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}x}} \mathrm{dP}_{q-1}(I). \tag{2.224}$$

We can form a compatible finite element complex in three-dimensions by taking the tensor product of elements in (2.224) with elements of a two-dimensional complex. In general, if we have the one- and two-dimensional finite element complexes:

$$V_h^0(I) \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}x}} V_h^1(I), \tag{2.225}$$

$$U_h^0(K) \xrightarrow{\nabla^\perp} U_h^1(K) \xrightarrow{\nabla\cdot} U_h^2(K), \tag{2.226}$$

a three-dimensional compatible tensor product complex is formed on $K \times I$:

$$W_h^0 \xrightarrow{\nabla} W_h^1 \xrightarrow{\nabla\times} W_h^2 \xrightarrow{\nabla\cdot} W_h^3, \tag{2.227}$$

where

$$W_h^0 = U_h^0 \otimes V_h^0, \tag{2.228}$$

$$W_h^1 = \underbrace{\texttt{HCurl}(U_h^1 \otimes V_h^0)}_{W_h^{1,\mathrm{horiz.}}} \oplus \underbrace{\texttt{HCurl}(U_h^0 \otimes V_h^1)}_{W_h^{1,\mathrm{vert.}}}, \tag{2.229}$$

$$W_h^2 = \underbrace{\texttt{HDiv}(U_h^1 \otimes V_h^1)}_{W_h^{2,\mathrm{horiz.}}} \oplus \underbrace{\texttt{HDiv}(U_h^2 \otimes V_h^0)}_{W_h^{2,\mathrm{vert.}}}, \tag{2.230}$$

$$W_h^3 = U_h^2 \otimes V_h^1. \tag{2.231}$$

The operators `HCurl/HDiv` are modifiers applying appropriate transformations to product elements. They are a consequence of implementation, as summarized in McRae et al. (2016, §2.4.5). We elaborate further on the construction of each W_h^i. For the purpose of presentation, we take the two-dimensional cell to be a triangle, $K = \triangle$. However, the discussion here is equally valid for quadrilateral cells.

W_h^0: To form the H^1 element, the component elements must have nodes associated with the vertices of the product cell. As we previously discussed, these vertices are formed by taking the product of vertices of K and I. The constituent finite elements must therefore be H^1 elements, U_h^0 and V_h^0 respectively. For $u \in U_h^0$, $v \in V_h^0$, functions in $W_h^0 = U_h^0 \otimes V_h^0$ take the form of the scalar product: uv. An illustration for a lowest-order element is illustrated in Figure 2.16.

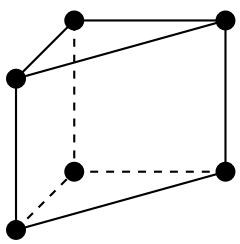
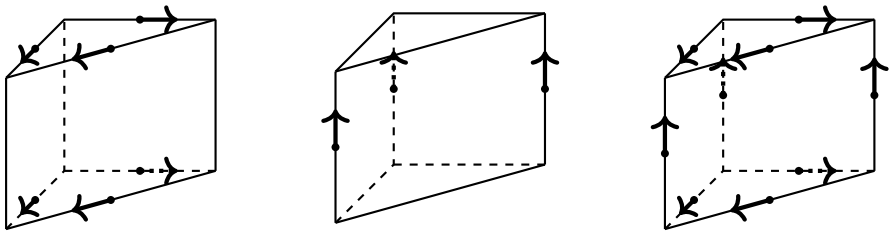


FIGURE 2.16: The result of taking the tensor product of a $\mathrm{P}_1(\triangle)$ triangular Lagrange element with a $\mathrm{P}_1(I)$ interval Lagrange element.



(A) $W_h^{1,\text{horiz.}}$ (B) $W_h^{1,\text{vert.}}$ (C) $W_h^{1,\text{horiz.}} \oplus W_h^{1,\text{vert.}}$

FIGURE 2.17: A lowest-order $H(\text{curl})$ element on a triangular prism (Fig. 2.17C). Here, the horizontal element (Fig. 2.17A) is constructed from $U_h^1 = \text{RT}_1(\triangle)$, the lowest-order Raviart-Thomas-Nédélec $H(\text{div})$ element (rotated 90 degrees), and $V_h^0 = \text{P}_1(I)$. The vertical element (Fig. 2.17B) is formed from the tensor product of $U_h^0 = \text{P}_1(\triangle)$ with $V_h^1 = \text{dP}_0(I)$.

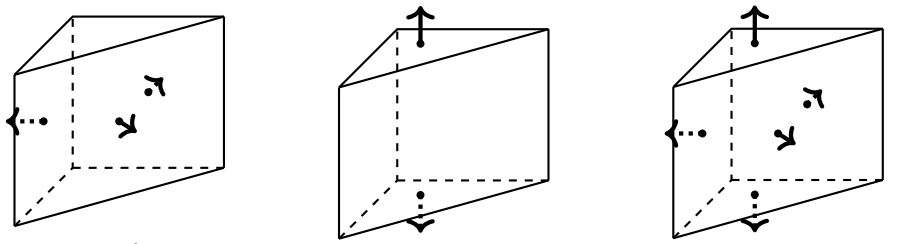
W_h^1: For the $H(\text{curl})$ element in (2.229), we must have nodes associated with cell edges to enforce tangential continuity. The construction of W_h^1, in our case, is described as the direct sum of a "horizontal" and "vertical" element, $W_h^{1,\text{horiz.}}$ and $W_h^{1,\text{vert.}}$ respectively. One way to construct the horizontal component is by taking the tensor product of a two-dimensional $H(\text{div})$ finite element, U_h^1, with a one-dimensional H^1 element, V_h^0. This produces a vector-valued element with nodes on the $(1,0)$-edges. For $\boldsymbol{u} \in U_h^1$, $v \in V_h^0$, elements of $U_h^1 \otimes V_h^0$ are vectors of the form: $\boldsymbol{u} \times v = \begin{Bmatrix} vu_x & vu_y \end{Bmatrix}$.

The product naturally takes values in $\mathbb{R}^2$, however, the $H(\text{curl})$ element must take values in $\mathbb{R}^3$. Using the modifier `HCurl`, we translate the two-dimensional vector into a three-dimensional vector with x and y components *rotated* by 90 degrees (identical to the construction of the two-dimensional edge elements presented in Section 2.2.2). The resulting three-dimensional vector becomes $\begin{Bmatrix} -vu_y & vu_x & 0 \end{Bmatrix}$. Moreover, the covariant Piola transform must be employed when mapping from reference cell to physical cell. We write the resulting horizontal $H(\text{curl})$ element as $W_h^{1,\text{horiz.}} = \texttt{HCurl}(U_h^1 \otimes V_h^0)$ (see Figure 2.17A).

Remark 5. *Alternatively, one may construct $W_h^{1,\text{horiz.}}$ by taking the tensor product of a two-dimensional $H(curl)$ element with an H^1 one-dimensional element. In this case, it is enough to interpret $\begin{Bmatrix} vu_x & vu_y \end{Bmatrix}$ to be the first two components of the vector in $\mathbb{R}^3$.*

Finally, we construct the vertical component, $W_h^{1,\text{vert.}}$, by taking the tensor product of an H^1 two-dimensional element with an L^2 one-dimensional element. This produces a scalar-valued element with nodes on the $(0,1)$-edges. We transform the product into a vector-valued quantity in $\mathbb{R}^3$ by simply scaling by the unit vector $e_z = \begin{Bmatrix} 0,0,1 \end{Bmatrix}$, which produces a vector whose tangential components are continuous on all vertical edges/faces. Since we need to preserve tangential components, the covariant Piola transform must be used to map functions from reference to physical space. We write the resulting vertical element as $W_h^{1,\text{vert.}} = \texttt{HCurl}(U_h^0 \otimes V_h^1)$. See Figure 2.17 for a complete illustration of a lowest-order $H(\text{curl})$ element.

W_h^2: The $H(\text{div})$ element follows a similar construction to that of the $H(\text{curl})$ element. We require that W_h^2 must have nodes associated with cell facets to ensure continuity of normal components. The element W_h^2 can be constructed as the direct sum of a "horizontal" element, $W_h^{2,\text{horiz.}}$, and a "vertical" element, $W_h^{2,\text{vert.}}$. To construct $W_h^{2,\text{horiz.}}$, we take the tensor product of a two-dimensional $H(\text{div})$ element U_h^1



(A) $W_h^{2,\text{horiz.}}$ (B) $W_h^{2,\text{vert.}}$ (C) $W_h^{2,\text{horiz.}} \oplus W_h^{2,\text{vert.}}$

FIGURE 2.18: A lowest-order $H(\text{div})$ element on a triangular prism (Fig. 2.18C). The horizontal element (Fig. 2.18A) is constructed from $U_h^1 = \text{RT}_1(\triangle)$, and $V_h^1 = \text{dP}_0(I)$. The vertical element (Fig. 2.18B) is formed from the product of $U_h^2 = \text{dP}_0(\triangle)$ with $V_h^1 = \text{P}_1(I)$.

with an L^2 one-dimensional element V_h^1. Again, the product naturally takes values in $\mathbb{R}^2$. Since U_h^1 is in $H(\text{div})$, it is sufficient to interpret the product as the first two components of a vector in $\mathbb{R}^3$ whose third component vanishes. Lastly, we require the contravariant Piola transform as our method of pullback. The resulting element is $W_h^{2,\text{horiz.}} = \texttt{HDiv}(U_h^1 \otimes V_h^1)$, where the modifier `HDiv` applies the necessary transformations to the $U_h^1 \otimes V_h^1$ vector-valued element (Figure 2.18A).

Remark 6. *Similarly with the construction of the horizontal $H(curl)$ element, if U_h^1 is a two-dimensional $H(curl)$, then the product $U_h^1 \otimes V_h^1$ must be rotated by 90 degrees.*

The vertical element $W_h^{2,\text{vert.}}$ is constructed from taking the product of an L^2 two-dimensional element U_h^2 with a one-dimensional H^1 element V_h^0. This produces a scalar-valued element with nodes on the $(2,0)$-facets of the product cell. Scaling the result by the unit vector e_z produces the desired finite element (Figure 2.18B).

W_h^3: Finally, the L^2 element must only have nodes associated with cell interiors. Therefore, both constituent finite elements must be in L^2. The result is illustrated for the lowest-order elements in Figure 2.19. An overview of the tensor product elements mentioned here is presented in Table 2.2.

Having the notion of product cells and tensor product spaces is particularly advantageous in atmospheric modeling. Typically, atmospheric meshes consist of an unstructured horizontal "base" mesh (of a sphere, for example) which is uniformly *extruded* in the vertical. The structured vertical grid is used to facilitate staggering of thermodynamic variables, which are common for standard finite difference or finite volume methods used in atmospheric dynamical cores (Wood et al., 2014; Staniforth and Wood, 2008; Staniforth, Melvin, and Cotter, 2013).

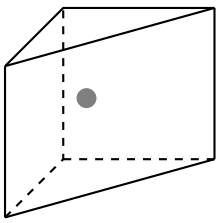


FIGURE 2.19: The result of taking the tensor product of a $\text{dP}_0(\triangle)$ discontinuous Lagrange element with a $\text{dP}_0(I)$ interval (discontinuous) element.

TABLE 2.2: Summary of the construction of tensor product elements in three-dimensions, using the finite element complexes (2.225) and (2.226). This is a modification of Table 2 by McRae et al. (2016). The first two columns denote the pairs of function spaces from which the tensor product element is constructed. The modifiers `HCurl` and `HDiv` ensure the resulting finite element has the appropriate shape (vector-valued) and reference cell mapping (covariant/contravariant Piola transform). Tensor product elements with no modifier and an "identity" mapping are scalar-valued elements with pullbacks consisting of *only* a change-of-coordinates transformation.

Product (2D × 1D)	Element $\left(U_h^i \otimes V_h^j\right)$	Components	Modifier	Shape	Result	Mapping
$H^1 \times H^1$	$U_h^0 \otimes V_h^0$	$u \times v$	–	scalar	uv	identity
$H^1 \times L^2$	$U_h^0 \otimes V_h^1$	$u \times v$	–	scalar	uv	identity
$H^1 \times L^2$	$U_h^0 \otimes V_h^1$	$u \times v$	`HCurl`	vector ($\mathbb{R}^3$)	$\{0 \quad 0 \quad uv\}$	covariant Piola
$H(\mathrm{div}) \times H^1$	$U_h^1 \otimes V_h^0$	$\{u_x \quad u_y\} \times v$	–	vector ($\mathbb{R}^2$)	$\{vu_x \quad vu_y\}^\dagger$	–
$H(\mathrm{div}) \times H^1$	$U_h^1 \otimes V_h^0$	$\{u_x \quad u_y\} \times v$	`HCurl`	vector ($\mathbb{R}^3$)	$\{-vu_y \quad vu_x \quad 0\}$	covariant Piola
$H(\mathrm{div}) \times L^2$	$U_h^1 \otimes V_h^1$	$\{u_x \quad u_y\} \times v$	–	vector ($\mathbb{R}^2$)	$\{vu_x \quad vu_y\}^\dagger$	–
$H(\mathrm{div}) \times L^2$	$U_h^1 \otimes V_h^1$	$\{u_x \quad u_y\} \times v$	`HDiv`	vector ($\mathbb{R}^3$)	$\{vu_x \quad vu_y \quad 0\}$	contravariant Piola
$L^2 \times H^1$	$U_h^2 \otimes V_h^0$	$u \times v$	–	scalar	uv	identity
$L^2 \times H^1$	$U_h^2 \otimes V_h^0$	$u \times v$	`HDiv`	vector ($\mathbb{R}^3$)	$\{0 \quad 0 \quad uv\}$	contravariant Piola
$L^2 \times L^2$	$U_h^2 \otimes V_h^1$	$u \times v$	–	scalar	uv	identity

Entries marked with † have little practical use for our application; they are two-dimensional vectors defined over three-dimensional cells. No mappings have been given for these product elements, as the Piola transform on three-dimensional cells requires all three spatial components to be explicitly defined.

2.4.5 Finite element space for the potential temperature

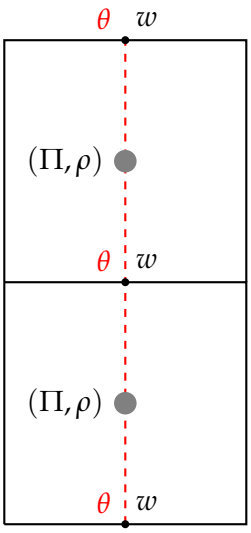
In Section 2.1.5, we summarized desirable numerical properties for atmospheric models. Most of those properties depend on the choice of horizontal grid and variable staggering. Within the context of finite difference methods, the Arakawa C-grid (recall Figure 2.3) is primarily used due to its favorable numerical properties. In three-dimensions, vertical grids are equally important for maintaining discrete balance relations in the vertical direction, most notably the hydrostatic balance relation (in Exner pressure formulation):

$$g\hat{\boldsymbol{k}} = -c_p\theta\frac{\partial\Pi}{\partial z}, \tag{2.232}$$

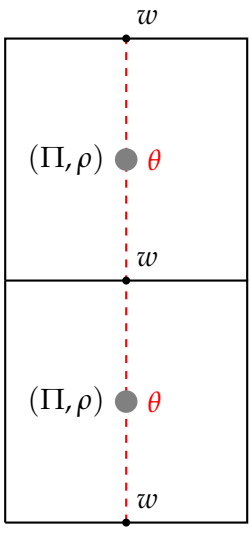
where $\hat{\boldsymbol{k}}$ is the upward-pointing normal, c_p is the specific heat at constant pressure, θ is the potential temperature (previously defined in (2.6)), and Π is the Exner pressure (defined in (2.9)). It is critically important that flows initially in hydrostatic balance do not deviate far from this state over time. A spontaneous loss of balance will render numerical models useless for weather prediction purposes (Staniforth and Wood, 2008).

For the vertical discretization, a structured grid is typically employed which allows for column-wise staggering of the thermodynamic variables, like θ. The two most popular choices in atmospheric modeling are the Lorenz grid (Lorenz, 1960), where the temperature is collocated with the pressure variable in a C-grid staggering, and the Charney-Phillips grid (Charney and Phillips, 1953), where temperature is positioned with the vertical component of the velocity. Figure 2.20 provides a visualization of the two grids in a vertical-slice (x, z) geometry.

The Lorenz grid is ideal for point-wise physics parameterization in atmospheric models, since all thermodynamic variables (pressure, density, and temperature) are



(A) The Charney-Phillips staggered grid.



(B) The Lorenz staggered grid.

FIGURE 2.20: Two commonly used vertical grids ((x, z) vertical slice) in atmospheric modeling. The Lorenz grid in (2.20B) collocates the temperature, θ, with the pressure, Π, and density, ρ, while the Charney-Phillips grid (2.20A) collocates θ with the vertical component of the velocity, w.

easily available and local to cell centers. However, it is well-known that the Lorenz grid supports the formation of an unphysical (zero-frequency) computational mode, which degrades numerical solutions over time near hydrostatic balance (Schneider, 1987; Arakawa and Konor, 1996). The Charney-Phillips grid avoids this problem entirely. However, extrapolation of temperature to cell centers is required (often through vertical averaging) in physics parameterization schemes. For a more detailed comparison of the two vertical grids, we refer the reader to the detailed studies performed by Holdaway, Thuburn, and Wood (2012a) and Holdaway, Thuburn, and Wood (2012b).

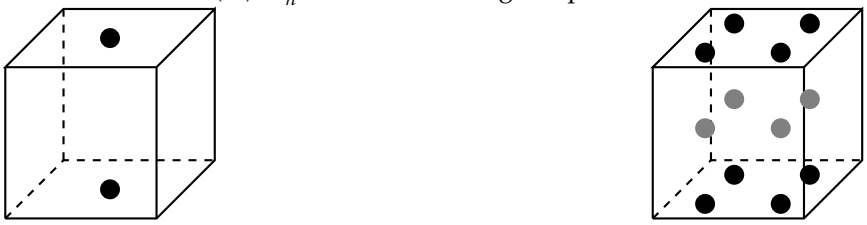
In a compatible finite element discretization, the horizontal C-grid staggering of variables is analogous to choosing the velocity to be in an $H(\mathrm{div})$-finite element space, and pressure/density to be in an L^2-finite element space. A three-dimensional discretization, with a structured vertical grid, is achieved by constructing a tensor product complex; that is, taking the product of a horizontal two-dimensional complex with a structured one-dimension complex in the vertical (see the constructions presented in (2.228)–(2.231)). Due to the vertical structure of a tensor product complex, a number of options for vertical staggering become available.

To emulate the Lorenz grid in a compatible finite element discretization, we can simply construct θ, the potential temperature, in the L^2 space, W_h^3 (the same finite element space for pressure/density). However, as we have previously stated, this particular grid can produce unphysical phenomena which rapidly degrades solutions near hydrostatic balance. Indeed the same numerical phenomena of the Lorenz grid can be observed when using this approach, as demonstrated by Melvin et al. (2018). Instead, we can construct a new finite element which mirrors the Charney-Phillips grid. To accomplish this, we create a tensor product element with nodes corresponding to the locations of nodes in the vertical component of the $H(\mathrm{div})$ velocity element, W_h^2. We call this space W_h^θ.

W_h^θ: We introduce a new finite element space W_h^θ which is constructed from a horizontally discontinuous element, but continuous in the vertical: $W_h^\theta = U_h^2 \otimes V_h^1$. This is, in fact, a scalar-version of the vertical velocity element $W_h^{2,\mathrm{vert.}}$ (without the `HDiv` modifier summarized in Table 2.2). Meaning that the finite element defining W_h^θ has the same number of nodes as $W_h^{2,\mathrm{vert.}}$, but differ in how they are pulled back to the reference element. The pullback in this case is simply the standard one for scalar functions. See Figure 2.21 for an illustration of the degrees of freedom for this particular element on triangular prisms and cubes.

This constructs the compatible finite element extension of the Charney-Phillips staggered grid. It was shown by Natale, Shipton, and Cotter (2016) that this choice of finite element space leads to an injective mapping between the pressure and θ in (2.232). Therefore, spurious hydrostatic pressure modes are avoided. A comparison of various tensor product finite element spaces for the temperature, including the W_h^θ element shown here, was presented by Melvin et al. (2018). They showed, for a simplified compressible (gravity wave) model, that using the space W_h^θ for the temperature variable produced the most consistent results and agreed with analytic dispersion relations for gravity waves. As a result, we shall exclusively use the space W_h^θ for the temperature in our compressible models throughout this dissertation.

(A) W_h^θ element on triangular prisms.



(B) W_h^θ element on cubes.

FIGURE 2.21: The lowest order and next-to-lowest order spaces for the W_h^θ finite element in three-dimensions.

2.4.6 Approximations of de-Rham complexes on hypersurfaces

It is worth taking a moment to discuss de-Rham complexes within a more general setting. Consider the following complex:

$$\cdots \longrightarrow W^i \xrightarrow{\mathrm{d}^i} W^{i+1} \longrightarrow \cdots \tag{2.233}$$

where W^i are Hilbert spaces and $\mathrm{d}^i : W^i \to W^{i+1}$ are closed linear maps, possibly unbounded, satisfying $\mathrm{d}^i \circ \mathrm{d}^{i-1} = 0$ for each i. When we say d^i are closed, we mean specifically that image of $\mathrm{d}^i W^i$ is closed in W^{i+1} for each i. Now suppose we have another de-Rham complex consisting of Hilbert spaces W_h^i, along with close linear maps d_h^i:

$$\cdots \longrightarrow W_h^i \xrightarrow{\mathrm{d}_h^i} W_h^{i+1} \longrightarrow \cdots \tag{2.234}$$

with $\mathrm{d}_h^i \circ \mathrm{d}_h^{i-1} = 0$ and W_h^i are finite-dimensional. The observant reader will notice that, in our definitions of the compatible finite element complexes (2.200) and (2.220), we restricted ourselves to the case when $W_h^i \subset W^i$. In this situation, we say (2.234) is a *subcomplex* of (2.233). This is also the main setting of the theory presented by Arnold, Falk, and Winther (2010). A natural question arises when we consider the commutative diagrams in Sections 2.4.2 and 2.4.4: what happens if $W_h^i \not\subset W^i$?

As discussed in Section 2.3.2, finite element discretizations on hypersurfaces (embedded manifolds) puts us in the more complicated situation where our discrete spaces are no longer subspaces of the continuous solution spaces. This is the aforementioned *variational crime* we alluded to in Remark 4. Fortunately, all is not lost. In particular, the compatible finite element de-Rham complex *can* be extended to handle this exact situation, as shown by Holst and Stern (2012) and Natale, Shipton, and Cotter (2016). We shall quickly summarize here to put the reader at ease.

Consider the following extended diagram:

$$\begin{array}{ccccccc} \cdots & \longrightarrow & W^i & \xrightarrow{\mathrm{d}^i} & W^{i+1} & \longrightarrow & \cdots \\ & & \iota^i \uparrow \downarrow \pi^i & & \iota^{i+1} \uparrow \downarrow \pi^{i+1} & & \\ \cdots & \longrightarrow & W_h^i & \xrightarrow{\mathrm{d}_h^i} & W_h^{i+1} & \longrightarrow & \cdots \end{array} \tag{2.235}$$

where $\pi^i : W^i \to W_h^i$ is a projection and $\iota^i : W_h^i \hookrightarrow W^i$ is an inclusion mapping such that $\pi^i \circ \iota^i$ is the identity map on W_h^i. In the analysis of Arnold, Falk, and Winther (2010) where $W_h^i \subset W^i$, ι^i is an isometry. Therefore, W_h^i can be identified with the subcomplex $\iota^i W_h^i \subset W^i$ since:

$$\langle \iota^i u, \iota^i v \rangle_{W^i} = \langle u, v \rangle_{W^i} = \langle u, v \rangle_{W_h^i}, \quad \forall u, v \in W_h^i, \tag{2.236}$$

where $\langle \cdot, \cdot \rangle_X$ denotes an inner product on a Hilbert space X. If a weak formulation is posed on the complex W^i, then the well-posedness of a Galerkin method on W_h^i relies on one critical assumption: π^k exists. Indeed, for the class of finite element spaces we consider, such a projection does exist (Arnold, Falk, and Winther, 2006; Arnold, Falk, and Winther, 2010; Arnold, 2013; Arnold and Awanou, 2014) This means a Galerkin problem may be posed on the subcomplex W_h^i, and standard error estimates follow per usual finite element theory.

Holst and Stern (2012) extended the diagram (2.235) to the case where $W_h^i \not\subset W^i$; that is, where ι^i do *not* satisfy (2.236). The only requirement is that ι^i must at least be an injective morphism (structure-preserving map) of Hilbert complexes with the property that $\pi^i \circ \iota^i$ is the identity. The existence of such projections π^i follows from Theorem 3.7 of Holst and Stern (2012).

As a result, a complex on W_h^i is an *approximation* to a true subcomplex of W^i, and each W_h^i can still be identified with the subspaces $\iota^i W_h^i \subset W^i$ using the modified inner product:

$$\langle \iota^i u, \iota^i v \rangle_{W^i} = \langle \iota^{i*} \iota^i u, v \rangle_{W_h^i}, \quad \forall u, v \in W_h^i, \tag{2.237}$$

where ι^{i*} is the adjoint of ι^i (Holst and Stern, 2012, Theorem 3.8). Notice that when $W_h^i \subset W^i$, then ι^i is an isometry and (2.237) reduces to the original identification in (2.236) since $\iota^{i*}\iota^i$ becomes the identity operator on W_h^i.

It was also shown by Holst and Stern (2012, §4) that if the computational mesh is a piece-wise polynomial approximation of a smooth embedded manifold (using isoparametric finite elements), then functions can still be approximated at the optimal rate. These results were extended by Natale, Shipton, and Cotter (2016) to piece-wise polynomial meshes of a spherical annulus within the context of tensor product compatible finite elements. Therefore, everything discussed thus far can be addressed without relying on subcomplex inclusion.

2.5 Chapter summary

In this chapter, we established the equations sets of interest for simulating atmospheric flows. This includes all relevant definitions of PDE systems ranging in complexity through the hierarchy of models. In doing so, we present core ideas that are particularly relevant for modeling the atmosphere. This includes: the effects due

to planetary rotation (Coriolis), Boussinesq and hydrostatic approximations, geophysical balance relations (hydrostatic and geostrophic balanced flows), and wave solutions of shallow water systems. Moreover, we highlight desirable properties of numerical models for operational dynamical cores, which motivates our choice of numerics throughout the rest of the dissertation.

Next, we give a rigorous definition of the finite element, followed by constructions of various finite element families which are relevant for our application. The elements presented are commonly used throughout many different application areas, but have only relatively recently been considered before for use in simulating geophysical flows. We also present some less well-known concepts concerning finite element discretization on immersed manifolds.

The rest of the chapter is dedicated to summarizing the notion of a compatible finite element discretization. The framework of compatible finite elements is intimately connected with the discipline of finite element exterior calculus (FEEC) (Arnold, Falk, and Winther, 2006; Arnold, Falk, and Winther, 2010; Arnold and Awanou, 2014), and is the core reason compatible finite elements possesses the key numerical properties which makes them appealing for atmospheric application. Specifically, we discuss the construction of elements with tensor product structure, allowing the resulting compatible finite element discretization to facilitate variable staggering of thermodynamic quantities. Having formally introduced all relevant topics, the rest of this dissertation will focus on deriving compatible finite element discretizations of the various equation sets and their hybridizable formulations.

3 Hybridizable compatible finite element methods

3.1 The hybridizable mixed method

The first hybridization of a finite element method was proposed by De Veubeke (1965) as an implementation technique for solving linear elasticity problems. The idea was to relax the continuity between elements and enforce the lost information through a Lagrange multiplier. The global matrix equation is then algebraically reduced to a symmetric system for the multiplier. Static condensation, a widely known algebraic technique at the time for reducing the size of an already assembled system of equations (Irons, 1965; Guyan, 1965), was largely indistinguishable from hybridization methods. However, Arnold and Brezzi in 1985 proved that hybridization was more than just an implementation trick. They showed that the new unknown λ, the Lagrange multiplier enforcing a continuity condition on the approximate flux variable, contains extra information about the exact solution (Arnold and Brezzi, 1985). This information is then used to enhance the accuracy of the finite element solution through local post-processing.

The hybridizable mixed method is well-established for the mixed formulations of second-order elliptic equations (Cockburn and Gopalakrishnan, 2004; Cockburn, Gopalakrishnan, and Lazarov, 2009; Brezzi and Fortin, 1991). In this section, we outline the construction of a hybridizable method for equation sets relevant for geophysical flows. We start with deriving the method for a linear shallow water model, as this equation set mirrors many aspects of standard mixed formulations of the positive-definite Helmholtz equation. We provide some new analysis which builds on previous characterization results for simplified mixed systems.

3.1.1 Compatible discretization of a linear shallow water model

Consider the following linearized shallow water model defined on a periodic domain $\Omega \subset \mathbb{R}^2$:

$$\frac{\partial \boldsymbol{u}}{\partial t} + f\boldsymbol{u}^{\perp} + g\nabla D = 0, \tag{3.1}$$

$$\frac{\partial D}{\partial t} + H\nabla \cdot \boldsymbol{u} = 0, \tag{3.2}$$

where $\boldsymbol{u}$ is the fluid velocity, D the fluid depth, f is the Coriolis parameter, g is the acceleration due to gravity, and H is a mean depth coefficient. Equations (3.1)–(3.2) is a linearization of the nonlinear shallow water model presented in (2.59)–(2.60), and primarily serves as a toy-model of barotropic waves in the atmosphere. We note that $\boldsymbol{u}$ and D here are actually linear perturbations around a steady-state with mean

depth H (previously written as $\delta \boldsymbol{u}$ and δD in equations (2.83)–(2.84)). In a slight abuse of notation, we neglect δ in our terms for simplicity.

Now consider the two-dimensional discrete de-Rham complex defined on a mesh $\mathcal{T}_h$ of Ω[1], consisting of polygonal cells K:

$$U_h^0 \xrightarrow{\nabla^\perp} U_h^1 \xrightarrow{\nabla\cdot} U_h^2. \tag{3.3}$$

The finite element spaces U_h^i, $i = 0, 1, 2$, are defined as:

$$U_h^0 = \{\psi \in H^1 : \psi|_K \in U_0(K), \forall K \in \mathcal{T}_h\}, \tag{3.4}$$
$$U_h^1 = \{\boldsymbol{w} \in H(\mathrm{div}) : \boldsymbol{w}|_K \in U_1(K), \forall K \in \mathcal{T}_h\}, \tag{3.5}$$
$$U_h^2 = \{\phi \in L^2 : \phi|_K \in U_2(K), \forall K \in \mathcal{T}_h\}, \tag{3.6}$$

where $U_0(K)$, $U_1(K)$, and $U_2(K)$ are appropriate polynomial spaces for their respective finite element families. For example, if $U_h^0 = \mathrm{P}_q$, $U_h^1 = \mathrm{RT}_q$, and $U_h^2 = \mathrm{dP}_{q-1}$, then $U_0(K) = \mathcal{P}_q(K)$, $U_1(K) = [\mathcal{P}_q(K)]^d + \boldsymbol{x}\mathcal{P}_{q-1}(K)$, and $U_2(K) = \mathcal{P}_{q-1}(K)$.

Proceeding using the method of Rothe (1930), we first discretize in time using the implicit-midpoint rule[2] to obtain the following semi-discrete PDE system for the fields at time-step n:

$$\boldsymbol{u}^n + \frac{\Delta t}{2} f\,(\boldsymbol{u}^n)^\perp + \frac{g\Delta t}{2}\nabla D^n = \boldsymbol{r}_1(\boldsymbol{u}^{n-1}, D^{n-1}), \tag{3.7}$$
$$D^n + \frac{H\Delta t}{2}\nabla\cdot\boldsymbol{u}^n = r_2(\boldsymbol{u}^{n-1}, D^{n-1}), \tag{3.8}$$

where

$$\boldsymbol{r}_1(\boldsymbol{u}^{n-1}, D^{n-1}) = \boldsymbol{u}^{n-1} - \frac{\Delta t}{2}\left(\boldsymbol{u}^{n-1}\right)^\perp - \frac{g\Delta t}{2}\nabla D^{n-1}, \tag{3.9}$$
$$r_2(\boldsymbol{u}^{n-1}, D^{n-1}) = D^{n-1} - \frac{H\Delta t}{2}\nabla\cdot\boldsymbol{u}^{n-1}. \tag{3.10}$$

Multiplying equation (3.7) by a test function $\boldsymbol{w} \in U_h^1$ and (3.8) by $\phi \in U_h^2$, we integrate by parts to obtain the fully discrete weak formulation of the linear model: find $\boldsymbol{u}_h \in U_h^1$ and $D_h \in U_h^2$ such that

$$\int_{\mathcal{T}_h} \boldsymbol{w}\cdot\boldsymbol{u}_h^n\,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2}\int_{\mathcal{T}_h}\boldsymbol{w}\cdot f\,(\boldsymbol{u}_h^n)^\perp\,\mathrm{d}\boldsymbol{x} - \frac{g\Delta t}{2}\int_{\mathcal{T}_h} D_h^n\nabla_h\cdot\boldsymbol{w}\,\mathrm{d}\boldsymbol{x} = R_{\boldsymbol{u}}^{n-1}[\boldsymbol{w}], \tag{3.11}$$
$$\int_{\mathcal{T}_h}\phi D_h^n\,\mathrm{d}\boldsymbol{x} + \frac{H\Delta t}{2}\int_{\mathcal{T}_h}\phi\nabla_h\cdot\boldsymbol{u}_h^n\,\mathrm{d}\boldsymbol{x} = R_D^{n-1}[\phi], \tag{3.12}$$

for all $\boldsymbol{w} \in U_h^1$ and $\phi \in U_h^2$, where $\nabla_h\cdot$ is defined cell-wise via:

$$(\nabla_h\cdot\boldsymbol{w})\,|_K = \nabla|_K\cdot\boldsymbol{w}, \quad \forall K \in \mathcal{T}_h. \tag{3.13}$$

Note that we have made the implicit assumption that g is not spatially varying, but the discussion in this section still follows similarly. Here, the residuals of the linear

[1] If Ω is the surface of a sphere, then $\mathcal{T}_h$ is assumed to be a mesh constructed from a piece-wise polynomial approximation of Ω, as described in the framework of Holst and Stern (2012) and Natale, Shipton, and Cotter (2016).

[2] Also equivalent to the Crank-Nicolson method when the PDE is linear.

system $R_{\boldsymbol{u}}^{n-1}[\boldsymbol{w}]$ and $R_D^{n-1}[\phi]$ are the functionals (covectors) given by:

$$R_{\boldsymbol{u}}^{n-1}[\boldsymbol{w}] = \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{u}_h^{n-1} \,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2} \int_{\mathcal{T}_h} \boldsymbol{w} \cdot f \left(\boldsymbol{u}_h^{n-1}\right)^{\perp} \,\mathrm{d}\boldsymbol{x} + \frac{g\Delta t}{2} \int_{\mathcal{T}_h} D_h^{n-1} \nabla_h \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x}, \tag{3.14}$$

$$R_D^{n-1}[\phi] = \int_{\mathcal{T}_h} \phi D_h^{n-1} \,\mathrm{d}\boldsymbol{x} - \frac{H\Delta t}{2} \int_{\mathcal{T}_h} \phi \nabla_h \cdot \boldsymbol{u}_h^{n-1} \,\mathrm{d}\boldsymbol{x}. \tag{3.15}$$

The residual covectors can be further rewritten as

$$R_{\boldsymbol{u}}^{n-1}[\boldsymbol{w}] = \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{r}_1^h(\boldsymbol{u}_h^{n-1}, D_h^{n-1}) \,\mathrm{d}\boldsymbol{x}, \tag{3.16}$$

$$R_D^{n-1}[\phi] = \int_{\mathcal{T}_h} \phi r_2^h(\boldsymbol{u}_h^{n-1}, D_h^{n-1}) \,\mathrm{d}\boldsymbol{x}, \tag{3.17}$$

where

$$\boldsymbol{r}_1^h(\boldsymbol{u}_h^{n-1}, D_h^{n-1}) = \boldsymbol{u}_h^{n-1} - \frac{\Delta t}{2} \left(\boldsymbol{u}_h^{n-1}\right)^{\perp} - \frac{g\Delta t}{2} \widetilde{\nabla}_h D_h^{n-1}, \tag{3.18}$$

$$r_2^h(\boldsymbol{u}_h^{n-1}, D_h^{n-1}) = D_h^{n-1} - \frac{H\Delta t}{2} \nabla_h \cdot \boldsymbol{u}_h^{n-1}. \tag{3.19}$$

In (3.18), $\widetilde{\nabla}_h$ should be interpreted as the operator which is dual to $\nabla_h \cdot$ through integration by parts; this allows us to approximate gradients of functions in L^2. More concretely, we define $\widetilde{\nabla}_h p$, for some $p \in U_h^2$, via

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \widetilde{\nabla}_h p \,\mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}_h} p \nabla_h \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x}, \tag{3.20}$$

for all $\boldsymbol{w} \in U_h^1$.

Remark 7. *Since we must solve the* same *linear system* (3.11)–(3.12) *with different right-hand sides (determined from initial conditions or fields from previous time-steps), we drop the superscript "n" in our notation for brevity. Hence, the residual functions* $\boldsymbol{r}_1^h$ *and* r_2^h *should always be interpreted as expressions of fields which are already known at any given time-step.*

The semi-discrete compatible finite element (spatial) discretization of (3.1)–(3.2) was shown by Cotter and Shipton (2012) to be both energy- and mass-conserving. This will also be preserved in the full discrete system, assuming a structure-preserving (symplectic) time-integrator is used, such as the implicit midpoint method (see Wimmer, Cotter, and Bauer (2019)).

Additionally, it was shown that the spatial discretization is free of spurious pressure modes, a critical requirement for atmospheric dynamical cores (see the discussion in Section 2.1.5). Within our context, this means there exists a constant $c > 0$, independent of the mesh resolution h, such that for all $\phi_h \in U_h^2$, there exists a nonzero $\boldsymbol{u}_h \in U_h^1$ satisfying

$$\int_{\mathcal{T}_h} \phi_h \nabla_h \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} \geq c \|\phi_h\|_{L^2} \|\boldsymbol{u}_h\|_{H(\mathrm{div})}. \tag{3.21}$$

Moreover, the fact that the compatible finite element discretization supports steady geostrophic modes, and avoids spurious computational modes in the discrete wave

equations (see (2.88) and (2.104)) for f- and β-plane approximations, is discussed in detail by Cotter and Shipton (2012, §2.5–2.8).

Let $\{\boldsymbol{\Psi}\}_{i=1}^{p}$ and $\{\Phi\}_{i=1}^{q}$ denote the global finite element bases for U_h^1 and U_h^2 respectively. Then in typical finite element fashion, the linear system of equations for the velocity and depth unknowns is obtained by first expanding $\boldsymbol{u}_h$ and D_h as the linear combinations:

$$\boldsymbol{u}_h = \sum_{i=1}^{p} u_i \boldsymbol{\Psi}_i, \tag{3.22}$$

$$D_h = \sum_{i=1}^{q} D_i \Phi_i, \tag{3.23}$$

where the coefficients u_i and D_i must be determined at each time-step. Following the standard Galerkin approach, we take $\boldsymbol{w} = \boldsymbol{\Psi}_j$ and $\phi = \Phi_j$ in (3.11)–(3.12), which produces the $(p+q) \times (p+q)$ matrix system:

$$\boldsymbol{\mathcal{A}}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{M}_1 + \frac{\Delta t}{2}\boldsymbol{C} & -\frac{g\Delta t}{2}\boldsymbol{B}^T \\ \frac{H\Delta t}{2}\boldsymbol{B} & \boldsymbol{M}_2 \end{bmatrix} \begin{Bmatrix} \boldsymbol{U} \\ \boldsymbol{D} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F} \\ \boldsymbol{G} \end{Bmatrix} = \boldsymbol{b}, \tag{3.24}$$

where $\boldsymbol{U} = \begin{Bmatrix} u_1 & \cdots & u_p \end{Bmatrix}^T$ and $\boldsymbol{D} = \begin{Bmatrix} D_1 & \cdots & D_q \end{Bmatrix}^T$ are the coefficient vectors, the matrix operators are

$$(\boldsymbol{M}_1)_{ij} = \int_{\mathcal{T}_h} \boldsymbol{\Psi}_i \cdot \boldsymbol{\Psi}_j \, \mathrm{d}\boldsymbol{x} = \sum_{K \in \mathcal{T}_h} \int_K \boldsymbol{\Psi}_i \cdot \boldsymbol{\Psi}_j \, \mathrm{d}\boldsymbol{x}, \tag{3.25}$$

$$(\boldsymbol{C})_{ij} = \int_{\mathcal{T}_h} \boldsymbol{\Psi}_i \cdot f\boldsymbol{\Psi}_j^{\perp} \, \mathrm{d}\boldsymbol{x} = \sum_{K \in \mathcal{T}_h} \int_K \boldsymbol{\Psi}_i \cdot f\boldsymbol{\Psi}_j^{\perp} \, \mathrm{d}\boldsymbol{x}, \tag{3.26}$$

$$(\boldsymbol{B})_{ij} = \int_{\mathcal{T}_h} \Phi_j \nabla_h \cdot \boldsymbol{\Psi}_i \, \mathrm{d}\boldsymbol{x} = \sum_{K \in \mathcal{T}_h} \int_K \Phi_j \nabla \cdot \boldsymbol{\Psi}_i \, \mathrm{d}\boldsymbol{x}, \tag{3.27}$$

$$(\boldsymbol{M}_2)_{ij} = \int_{\mathcal{T}_h} \Phi_i \Phi_j \, \mathrm{d}\boldsymbol{x} = \sum_{K \in \mathcal{T}_h} \int_K \Phi_i \Phi_j \, \mathrm{d}\boldsymbol{x}, \tag{3.28}$$

and the right-hand sides are given by: $(\boldsymbol{F})_j = R_{\boldsymbol{u}}^{n-1}[\boldsymbol{\Psi}_j]$, $(\boldsymbol{G})_j = R_h^{n-1}[\Phi_j]$.

The system in (3.24) must be inverted at each time-step to determine the coefficients $\boldsymbol{U}$ and $\boldsymbol{D}$. While one could invert $\boldsymbol{\mathcal{A}}$ directly (for example, via an LU factorization), this quickly becomes impractical for big problems due to large memory requirements. Instead, iterative methods like Krylov subspace methods are often the only practical option.

Krylov subspace methods are a class of iterative methods for solving linear systems of equations. They are widely used in practice for problems in engineering and the physical sciences. Commonly used Krylov methods include (but are not limited to):

- The conjugate gradient method (CG) for symmetric positive-definite systems;
- The conjugate residual method (CR) for symmetric systems;
- The minimal residual (MINRES) method for symmetric indefinite systems; and
- The generalized minimal residual (GMRES) and conjugate residual (GCR) methods for invertible matrices which are not necessarily symmetric or positive-definite.

The convergence of Krylov methods are dependent on the *condition number* of the matrix. In many scientific applications, matrices arising from discretized PDEs often have very large condition numbers, meaning Krylov methods will take longer to converge. For this reason, *preconditioners* are used to accelerate the convergence of a Krylov method. We will discuss this in more detail in Chapter 4. For a comprehensive overview of Krylov methods, we refer the interested reader to Saad (2003).

The matrix $\boldsymbol{\mathcal{A}}$ is a *saddle-point* operator, where standard iterative solvers for elliptic equations (such as the CG method) are ineffective. This is due to the fact that $\boldsymbol{\mathcal{A}}$ is non-symmetric and *indefinite*.[3] Instead, solvers designed around the inverse of the Schur-complement factorization of $\boldsymbol{\mathcal{A}}$ are typically used (Benzi, Golub, and Liesen, 2005). Alternatively, one can use $H(\mathrm{div})$-multigrid methods (Arnold, Falk, and Winther, 2000), requiring complex overlapping smoothers, or auxiliary space multigrid (Hiptmair and Xu, 2007).

In exact arithmetic, the inverse of the Schur-complement factorization of $\boldsymbol{\mathcal{A}}$ is

$$\boldsymbol{\mathcal{A}}^{-1} = \begin{bmatrix} \boldsymbol{I} & \frac{g\Delta t}{2}\widetilde{\boldsymbol{M}}_1^{-1}\boldsymbol{B}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{M}}_1^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\frac{H\Delta t}{2}\boldsymbol{B}\widetilde{\boldsymbol{M}}_1^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{3.29}$$

where $\widetilde{\boldsymbol{M}}_1 = \boldsymbol{M}_1 + \frac{\Delta t}{2}\boldsymbol{C}$, and $\boldsymbol{S}$ is the Schur-complement operator

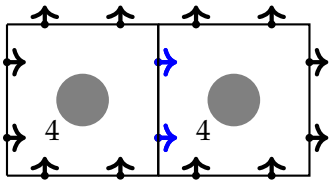
$$\boldsymbol{S} = \boldsymbol{M}_2 + \frac{\Delta t^2}{4} g H \boldsymbol{B}\widetilde{\boldsymbol{M}}_1^{-1}\boldsymbol{B}^T. \tag{3.30}$$

Inverting $\boldsymbol{S}$ directly is often impractical due to the fact that the matrix $\widetilde{\boldsymbol{M}}_1$, while sparse, has a globally *dense* inverse. This is due to the fact that elements are coupled through the continuity of normal components of fields in U_h^1. The main challenge in constructing an efficient preconditioner for numerically inverting $\boldsymbol{S}$ is finding good sparse approximations of $\widetilde{\boldsymbol{M}}_1^{-1}$. Since we require an approximation of $\boldsymbol{S}$, the preconditioner is inexact and therefore requires an outer nonsymmetric Krylov method, like GMRES or GCR, to reduce the problem residual.

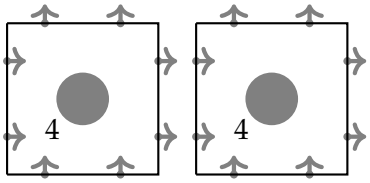
We remark here that treatment of the Coriolis term can be handled in a couple of different ways. One approach is to treat the term explicitly, meaning the contribution only contributes to the right-hand side residuals. This means that $\widetilde{\boldsymbol{M}}_1^{-1} = \boldsymbol{M}_1^{-1}$ and the operator becomes symmetric positive-definite. Moreover, diagonal approximations of $\boldsymbol{M}_1$ (either through mass-lumping or just extracting the diagonal) have been shown to give fairly good sparse approximations to $\boldsymbol{S}$ (Mitchell and Müller, 2016). Another option is for a purely implicit formulation of the Coriolis term (as written in this section). This results in a non-symmetric operator due to the asymmetry of the Coriolis matrix: $\boldsymbol{C}^T = -\boldsymbol{C}$. However, diagonal approximations of $\widetilde{\boldsymbol{M}}_1$ become less effective, especially when increasing the time-step size Δt. We shall elaborate further on this in later experiments featured in Chapter 4.

The hybridization technique can overcome this by producing discrete systems that can be algebraically transformed efficiently (and exactly) into a reduced elliptic equation for unknowns lying on element boundaries. Using the model problem (3.11)–(3.12), we derive the hybridizable equations and present some important results for the resulting condensed system.

[3] In other words, $\boldsymbol{\mathcal{A}}$ does not induce a discrete inner-product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{\mathcal{A}}} = \boldsymbol{x}^T \boldsymbol{\mathcal{A}} \boldsymbol{y}$ and therefore the conjugate gradient method is not guaranteed to converge.



(A) Two-cell diagram of a global RTC_2^f $H(\mathrm{div})$ space on quadrilaterals.



(B) Two-cell diagram of a global *broken* RTC_2^f space.

FIGURE 3.1: Two-cell diagrams of $H(\mathrm{div})$ finite element space and its discontinuous counterpart. Blue degrees of freedom are shared topologically between adjacent cells. Gray degrees of freedom are associated with the cell only.

3.1.2 A hybridizable discretization of a linear shallow water model

Broken $H(\mathrm{div})$ spaces

The main complication in forming (3.30) is directly related to the inter-elemental coupling induced by functions in U_h^1. To remove this coupling, we introduce a *broken* variant of U_h^1 from which we shall construct a new approximation of the velocity field. With $\Omega \subset \mathbb{R}^d$ as our domain and $\mathcal{T}_h$ denoting a mesh of Ω, let $U_h^1 \subset H(\mathrm{div})$ be as defined in (3.5). We define the broken space as the set:

$$\widehat{U}_h^1 = \{\boldsymbol{w} \in [L^2(\Omega)]^d : \boldsymbol{w}|_K \in U_1(K), \forall K \in \mathcal{T}_h\}. \tag{3.31}$$

The space $\widehat{U}_h^1$ is a subspace of $[L^2(\Omega)]^d$ with local shape functions from the $H(\mathrm{div})$ finite element space U_h^1. The key distinction between the two space are that the topological association with the finite element nodes are *always* restricted to the cell; there is no continuity of normal components between cell edges/faces. Notice that U_h^1 is actually a subspace of $\widehat{U}_h^1$ with the additional requirement that all functions are in $H(\mathrm{div})$:

$$U_h^1 = \widehat{U}_h^1 \cap \{\boldsymbol{w} \in [L^2(\Omega)]^d : [\![\boldsymbol{w}]\!]_e = 0, \forall e \in \mathcal{E}_h\} \tag{3.32}$$

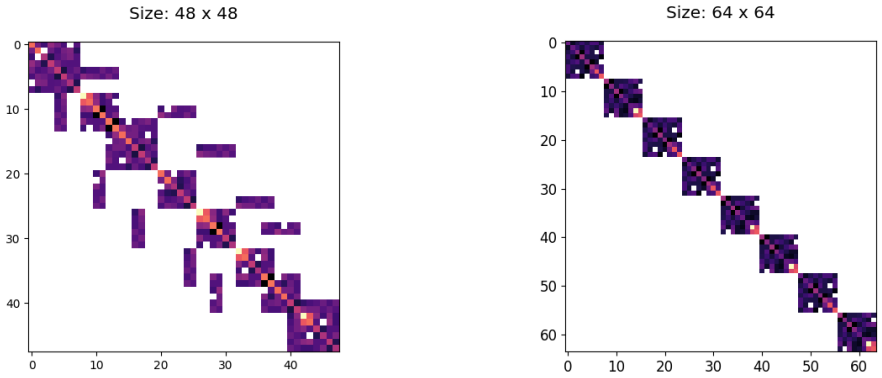
$$\equiv \widehat{U}_h^1 \cap H(\mathrm{div}; \Omega), \tag{3.33}$$

where $\mathcal{E}_h$ is the *skeleton* of the mesh:

$$\mathcal{E}_h = \{e \subset \partial K, \forall K \in \mathcal{T}_h\}, \tag{3.34}$$

and $[\![\cdot]\!]_e$ is the jump of the normal components of $\boldsymbol{w}$ as defined in (2.187).

For each of the $H(\mathrm{div})$ finite elements presented in Section 2.2.2, we can form a discontinuous variant to form the global space $\widehat{U}_h^1$. This is done by topologically disassociating degrees of freedom on shared edges/faces and restricting them to the cell only. See Figure 3.1 for an illustration of a Raviart-Thomas-Nédélec space on quadrilaterals and the corresponding broken variant.



(A) Mass matrix in U_h^1

(B) Mass matrix in $\widehat{U}_h^1$

FIGURE 3.2: Mass matrix global sparsity patterns for the U_h^1 and $\widehat{U}_h^1$ operators. Note that the $\widehat{U}_h^1$ mass matrix is block-diagonal, similar to the mass operator in U_h^2.

A direct consequence by testing in $\widehat{U}_h^1$, rather than U_h^1, is that the mass operator coupling velocity degrees of freedom is rendered block-diagonal; there is no coupling between cell interfaces. This makes global inversion of the new operator a purely local operation since the inverse can be computed by inverting each block separately. More importantly, the resulting global inverse is now *sparse*. See Figure 3.2 for a comparison of two mass operators corresponding to RT_2 elements on a triangular mesh consisting of eight cells. Since degrees of freedom are no longer shared between facets, the global size of the mass operator is larger than the original matrix; degrees of freedom on interior faces are counted *twice*.

Trace spaces

Now, we introduce an auxiliary finite element space defined only on the $d-1$ topological entities of $\mathcal{T}_h$. The finite element space on the mesh skeleton, $\mathcal{E}_h$, is defined as the set:

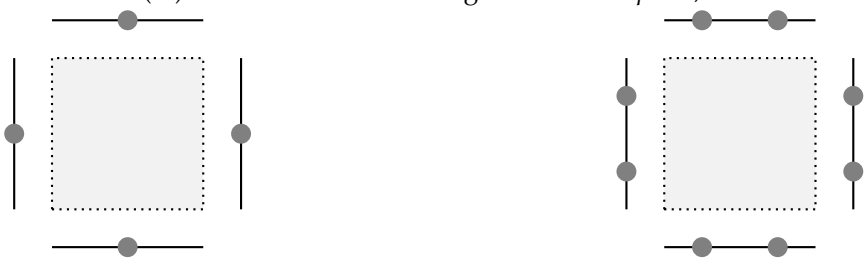
$$U_h^{\mathrm{tr}} = \{\gamma \in L^2(\mathcal{E}_h) : \gamma|_e \in M(e), \forall e \in \mathcal{E}_h\}, \tag{3.35}$$

where $M(e)$ is a function space on the facet e, typically taken to be a polynomial space $M(e) = \mathcal{P}_q(e)$. We refer to the space U_h^{tr} as the *trace space*. The trace space consists of scalar-valued functions which are discontinuous across vertices in two-dimensions, and edges/vertices in three-dimensions. By construction, all functions in U_h^{tr} are not defined in the interior of cells. Trace spaces on quadrilateral and triangular cells are illustrated in Figure 3.3. Three-dimensional generalizations follows an identical construction. See Figure 3.4 for a diagram illustrating the corresponding *global* trace space using constant polynomials on cell edges.

Just as with previous finite element constructions, evaluating a surface integral with arguments in U_h^{tr} is performed via a change-of-coordinates. This is accomplished by mapping from the facet in physical space to the corresponding facet of the reference cell $\hat{K}$, $\hat{e} \subset \partial\hat{K}$ (edges in two-dimensions and faces in three-dimensions). The local shape functions of the trace element on $\hat{e}$ correspond to the basis functions of the polynomial space $\mathcal{P}_q(\hat{e})$.

Now let U_h^1 be as previous defined in (3.5), where the local shape functions in $U_1(K)$ are vector-polynomials of degree $\leq q$ (specific constructions of $U_1(K)$ are shown for

(A) Trace elements on triangular cells for $q = 0, 1$.



(B) Trace elements on quadrilateral cells for $q = 0, 1$.

FIGURE 3.3: Trace elements on triangular and quadrilateral cells of degree $q = 0$ and $q = 1$. All degrees of freedom are topologically associated with each edge separately; there is no continuity across vertices.

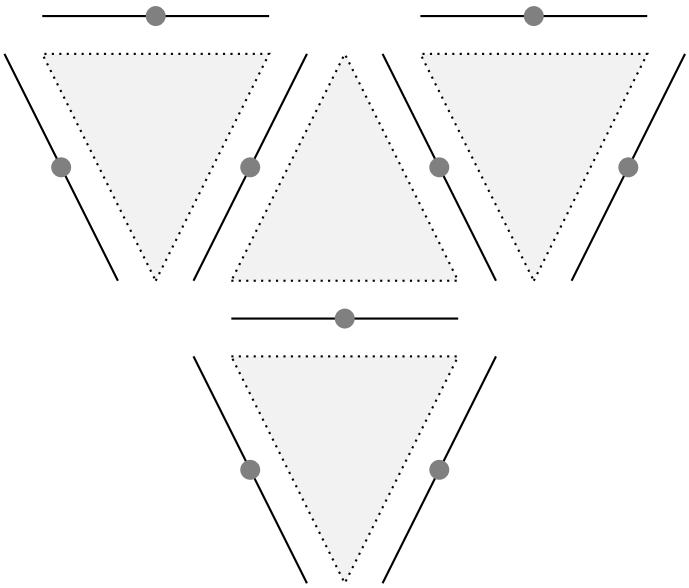


FIGURE 3.4: A global discontinuous trace space of lowest-order ($q = 0$) on a triangular mesh.

$H(\mathrm{div})$ elements in Section 2.2.2). Then for $\boldsymbol{w} \in U_h^1$, we have that $\boldsymbol{w} \cdot \boldsymbol{n}|_e \in \mathcal{P}_q(e)$, for all $e \in \mathcal{E}_h$. With U_h^{tr} as defined in (3.35) and taking $M(e) = \mathcal{P}_q(e)$, the following lemma is a direct consequence of the definition (3.33) (see Arnold and Brezzi (1985, Lemma 1.2)):

Lemma 2. *Suppose that $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$. Then $\widehat{\boldsymbol{w}} \in U_h^1$ if and only if*

$$\sum_{e \in \mathcal{E}_h} \int_e \gamma [\![\widehat{\boldsymbol{w}}]\!] \, \mathrm{d}S = 0, \tag{3.36}$$

for all $\gamma \in U_h^{\mathrm{tr}}$.

Proof. If $\widehat{\boldsymbol{w}} \in U_h^1$, then (3.36) holds trivially since, by definition, $[\![\widehat{\boldsymbol{w}}]\!] = 0$ over the entire mesh skeleton. To show (3.36) implies $\widehat{\boldsymbol{w}} \in U_h^1$, we remark that $\widehat{\boldsymbol{w}} \cdot \boldsymbol{n}|_e \in \mathcal{P}_q(e)$ for each facet e. Therefore, $[\![\widehat{\boldsymbol{w}}]\!]_e = p_e$ is some polynomial on e of degree $\leq q$. Since (3.36) is true for all γ, choosing a γ with $\gamma|_e = p_e$ implies that $p_e \equiv 0$. Hence, we have $[\![\widehat{\boldsymbol{w}}]\!]_e = 0$ for all e, i.e., $\widehat{\boldsymbol{w}} \in U_h^1$. □

Additionally, we also have the following result from Boffi, Brezzi, and Fortin (2013, Lemma 7.2.2).

Lemma 3. *If $\lambda \in U_h^{\mathrm{tr}}$ satisfies*

$$\sum_{e \in \mathcal{E}_h} \int_e \lambda [\![\widehat{\boldsymbol{w}}]\!] \, \mathrm{d}S = 0 \tag{3.37}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$, then $\lambda = 0$.

Proof. Let $e^* \in \mathcal{E}_h$ denote an edge on the boundary of an element K^*, i.e., $e^* \subset \partial K^*$, where $K^* \in \mathcal{T}_h$ is some cell of the mesh. Now we construct a vector $\widehat{\boldsymbol{w}}^* \in \widehat{U}_h^1$ that vanishes on all cells except for K^*:

$$\widehat{\boldsymbol{w}}^*|_K = 0, \quad \forall K \neq K^*, \tag{3.38}$$

and define the normal components of $\widehat{\boldsymbol{w}}^*$ on K^* by

$$\widehat{\boldsymbol{w}}^*|_{K^*} \cdot \boldsymbol{n}_e = \begin{cases} 1 \text{ if } e = e^* \\ 0 \text{ otherwise.} \end{cases} \tag{3.39}$$

Then we have

$$\sum_{e \in \mathcal{E}_h} \int_e \lambda [\![\widehat{\boldsymbol{w}}^*]\!] \, \mathrm{d}S = \int_{e^*} \lambda \, \mathrm{d}S. \tag{3.40}$$

Thus, (3.37) implies $\lambda = 0$, since the choice of e^* was arbitrary. □

Now let us piece everything together and derive a hybridizable formulation of the compatible finite element discretization (3.11)–(3.12).

Discrete hybridizable system

Proceeding from (3.7), We spatially discretize the linear momentum equation within a single cell K by testing with functions $\boldsymbol{w} \in U_1(K)$, and integrating by parts. We obtain the discrete equation with $\widehat{\boldsymbol{u}}_h \in U_1(K)$ and $D_h \in U_2(K)$:

$$\begin{aligned} &\int_K \widehat{\boldsymbol{w}} \cdot \widehat{\boldsymbol{u}}_h \, \mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_K \widehat{\boldsymbol{w}} \cdot f \widehat{\boldsymbol{u}}_h^{\perp} \, \mathrm{d}\boldsymbol{x} \\ -\frac{g\Delta t}{2} &\left(\int_K D_h \nabla_h \cdot \widehat{\boldsymbol{w}} \, \mathrm{d}\boldsymbol{x} + \int_{\partial K} \lambda_h \widehat{\boldsymbol{w}} \cdot \boldsymbol{n} \, \mathrm{d}S \right) = \int_K \widehat{\boldsymbol{w}} \cdot \boldsymbol{r}_1^h \, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.41}$$

for all $\widehat{\boldsymbol{w}} \in U_1(K)$. Here, the boundary integral should be interpreted as the sum of integrals over each facet of ∂K: $\int_{\partial K} \equiv \sum_{e \subset \partial K} \int_e$. The new unknown $\lambda_h|_e \in M(e)$ is the Lagrange multiplier appearing in the boundary term after integrating by parts.

Equation (3.41) reveals a physical interpretation of the multiplier; λ_h is an approximation to the fluid depth on cell facets. The discrete equation for linear mass conservation (3.12) remains unchanged.

After summing over all $K \in \mathcal{T}_h$, consider the extended discrete finite element problem: find $\widehat{\boldsymbol{u}}_h \in \widehat{U}_h^1$, $D_h \in U_h^2$, and $\lambda_h \in U_h^{\mathrm{tr}}$ such that

$$\int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot \widehat{\boldsymbol{u}}_h \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot f \widehat{\boldsymbol{u}}_h^{\perp} \,\mathrm{d}\boldsymbol{x}$$
$$- \frac{g\Delta t}{2} \int_{\mathcal{T}_h} D_h \nabla_h \cdot \widehat{\boldsymbol{w}} \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \lambda_h [\![\widehat{\boldsymbol{w}}]\!] \,\mathrm{d}S = \int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot \boldsymbol{r}_1^h \,\mathrm{d}\boldsymbol{x}, \tag{3.42}$$
$$\int_{\mathcal{T}_h} \phi D_h \,\mathrm{d}\boldsymbol{x} + \frac{H\Delta t}{2} \int_{\mathcal{T}_h} \phi \nabla_h \cdot \widehat{\boldsymbol{u}}_h \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \phi r_2^h \,\mathrm{d}\boldsymbol{x}, \tag{3.43}$$
$$\int_{\mathcal{E}_h} \gamma [\![\widehat{\boldsymbol{u}}_h]\!] \,\mathrm{d}S = 0, \tag{3.44}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$, $\phi \in U_h^2$, and $\gamma \in U_h^{\mathrm{tr}}$. We call (3.42)–(3.44) the *hybridizable formulation* of the compatible finite element discretization (3.11)–(3.12). We denote the hybridization of standard mixed methods, such as the Raviart-Thomas method (RT) (Raviart and Thomas, 1977) or the Brezzi-Douglas-Marini (BDM) method (Brezzi, Douglas, and Marini, 1985), as H-RT and H-BDM respectively. Table 3.1 summarizes our nomenclature and provides explicit constructions of the relevant finite element spaces.

The Lagrange multiplier λ_h approximates the (continuous) variable $\frac{g\Delta t}{2} D$ on the mesh skeleton $\mathcal{E}_h$ (the approximation properties of λ_h was shown within the context of mixed formulations of second-order elliptic equations by Arnold and Brezzi (1985)). The third equation (3.44), which we call the *jump condition*, is an application of Lemma 2; using functions $\gamma \in U_h^{\mathrm{tr}}$, we can weakly enforce the continuity of $\widehat{\boldsymbol{u}}_h \cdot \boldsymbol{n}|_e$ across all facets. This ensures that, despite being constructed in a discontinuous finite element space, $\widehat{\boldsymbol{u}}_h \in U_h^1$. In fact, we can show that the solutions $\widehat{\boldsymbol{u}}_h$ and D_h *coincide* with the solutions of the original compatible finite element formulation (3.11)–(3.12).[4] We shall explicitly show this in Section 3.2.

[4] We mean specifically that the nodal values of both $\widehat{\boldsymbol{u}}_h$ and $\boldsymbol{u}_h$ are in exact agreement. Clearly the corresponding coefficient vectors in a computer implementation will not be equal (they are of different dimensions).

TABLE 3.1: Summary of the hybridizable variants of well-known mixed finite element methods: the Raviart-Thomas method on simplices (RT), the Brezzi-Douglas-Marini method on simplices (BDM), and the RT method on quadrilaterals (RTCF).

Method	$U_1(K)$	$U_2(K)$	$M(e)$
H-RT	$[\mathcal{P}_{q-1}(K)]^d + \boldsymbol{x}\mathcal{P}_{q-1}(K)$	$\mathcal{P}_{q-1}(K)$	$\mathcal{P}_{q-1}(e)$
H-BDM	$[\mathcal{P}_q(K)]^d$	$\mathcal{P}_{q-1}(K)$	$\mathcal{P}_q(e)$
H-RTCF†	$\mathcal{P}_{q,q-1}(K) \times \mathcal{P}_{q-1,q}(K)$ $\mathcal{P}_{q,q-1,q-1}(K) \times \mathcal{P}_{q-1,q,q-1}(K) \times \mathcal{P}_{q-1,q-1,q}(K)$	$\mathcal{Q}_{q-1}(K)$	$\mathcal{P}_{q-1}(e)$

†: Constructions for $V(K)$ of the H-RTCF method are shown for two- and three-dimensions.

At a first glance, the system (3.42)–(3.44) appears to not be any improvement over the original system. Indeed, the total number of unknowns greatly exceeds that of the original mixed system (3.11)–(3.12). However, the discrete matrix equations are far more amenable to work with.

Let $\widehat{\boldsymbol{U}}$, $\boldsymbol{D}$, and $\boldsymbol{\Lambda}$ be the coefficient vectors for the finite element expansions of $\widehat{u}_h \in \widehat{U}_h^1$, $D_h \in U_h^2$ and $\lambda_h \in U_h^{\mathrm{tr}}$ respectively. Then the discrete matrix equations have the form:

$$\begin{bmatrix} \widehat{\widehat{\boldsymbol{M}}}_1 & -\frac{g\Delta t}{2}\widehat{\boldsymbol{B}}^T & \boldsymbol{K}^T \\ \frac{H\Delta t}{2}\widehat{\boldsymbol{B}} & \boldsymbol{M}_2 & \boldsymbol{0} \\ \boldsymbol{K} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \widehat{\boldsymbol{U}} \\ \boldsymbol{D} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \widehat{\boldsymbol{F}} \\ \boldsymbol{G} \\ \boldsymbol{0} \end{Bmatrix}, \tag{3.45}$$

where $\widehat{\cdot}$ denotes matrices/vectors tested against discontinuous functions in $\widehat{U}_h^1$. The operator $\boldsymbol{K}$ is the matrix associated with the trace variables. If $\{\xi_j\}$ is a basis for U_1^{tr}, then

$$(\boldsymbol{K})_{ij} = \sum_{e \in \mathcal{E}_h} \int_e \xi_j [\![\widehat{\boldsymbol{\Psi}}_i]\!] \, \mathrm{d}S, \tag{3.46}$$

where $\{\widehat{\boldsymbol{\Psi}}_i\}$ is a basis for $\widehat{U}_h^1$.

Treating the velocity-depth component as a separate 2×2 block, with mixed operator $\widehat{\boldsymbol{\mathcal{A}}}$, we rewrite (3.45) as

$$\begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}} & \boldsymbol{\mathcal{K}}^T \\ \boldsymbol{\mathcal{K}} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{X} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{Bmatrix}, \tag{3.47}$$

where $\boldsymbol{\mathcal{K}} = \begin{bmatrix} K & \boldsymbol{0} \end{bmatrix}$, $\boldsymbol{X} = \begin{Bmatrix} \widehat{\boldsymbol{U}} & \boldsymbol{D} \end{Bmatrix}^T$, and $\boldsymbol{R} = \begin{Bmatrix} \widehat{\boldsymbol{F}} & \boldsymbol{G} \end{Bmatrix}^T$. Then the inverse of the Schur-complement factorization of the left-hand side matrix in (3.47) is:

$$\begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}} & \boldsymbol{\mathcal{K}}^T \\ \boldsymbol{\mathcal{K}} & \boldsymbol{0} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{I} & -\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{K}}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\mathcal{S}}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\mathcal{K}}\widehat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{3.48}$$

where $\boldsymbol{\mathcal{S}} = -\boldsymbol{\mathcal{K}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{K}}^T$ is the Schur-complement operator of the hybridizable system. It can be shown that $\boldsymbol{\mathcal{S}}$ is a positive-definite operator, and symmetric positive-definite when the Coriolis parameter is zero (see Section 3.2).

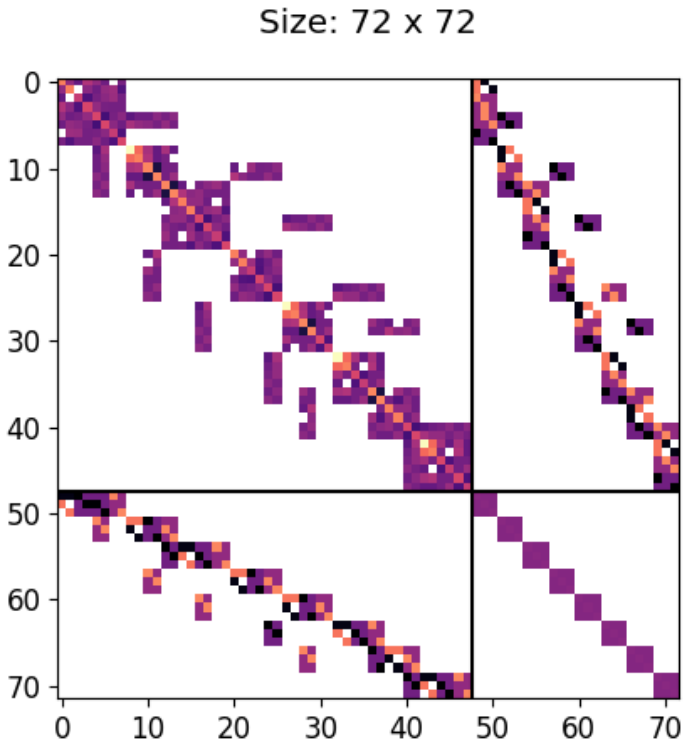
The term *hybridizable* in our context is a specific characterization of the discrete equations. By our choice of function spaces, the coupling of velocity and depth are purely *cell-local*. That is, $\widehat{\boldsymbol{\mathcal{A}}}$ is *block-diagonal*. The introduction of the Lagrange multiplier $\boldsymbol{\Lambda}$ permits the local elimination of $\boldsymbol{X}$ to construct the global problem for $\boldsymbol{\Lambda}$. In other words, the system

$$\boldsymbol{\mathcal{S}}\boldsymbol{\Lambda} = -\boldsymbol{\mathcal{K}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{R}, \tag{3.49}$$

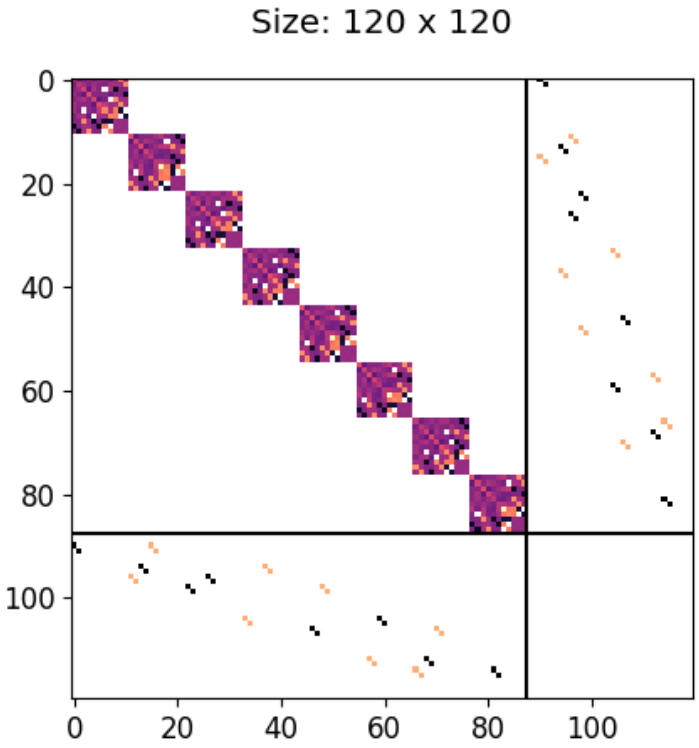
which we refer to as the equation associated with the *hybridized* problem, is the only globally coupled system requiring iterative inversion. The velocity and depth unknowns can be recovered locally by solving the system in each cell:

$$\boldsymbol{X} = \widehat{\boldsymbol{\mathcal{A}}}^{-1}\left(\boldsymbol{R} + \boldsymbol{\mathcal{K}}^T\boldsymbol{\Lambda}\right). \tag{3.50}$$

The fact that we can perform these local procedures is more clearly realized by inspecting the resulting sparsity patterns of the original mixed and hybridizable systems, shown in Figure 3.5.



(A) Original matrix system



(B) Hybridizable matrix system

FIGURE 3.5: Global sparsity patterns for the original discretization and the corresponding hybridizable system on mesh consisting of eight triangular cells. The block-structure in 3.5B is a direct consequence of our choice in finite element spaces (the global matrix has been reordered in such a way that the degrees of freedom $\boldsymbol{X}$ are organized by cell).

Analysis was done for the hybridization of first-order mixed systems of elliptic equations by Cockburn, Gopalakrishnan, and Lazarov (2009) and Cockburn and Gopalakrishnan (2004), which showed that the operator $\boldsymbol{\mathcal{S}}$ defines a positive-definite equation for the hybridized system, expressed as the variational problem: find $\lambda_h \in U_h^{\text{tr}}$ such that

$$s_h(\lambda_h, \gamma) = r_h(\gamma), \quad \forall \gamma \in U_h^{\text{tr}}. \tag{3.51}$$

At the time of this work, no such analysis exists for the systems we consider. The simplest relevant system to examine is precisely the shallow water model (3.1)–(3.2). Therefore, we shall extend the analysis of Cockburn, Gopalakrishnan, and Lazarov (2009) to the model equation considered in this section.

3.2 Analysis of the hybridizable method

In this section, we analyze the discrete hybridizable system for the linear shallow water model on a boundary-free domain Ω (such as on the surface of a sphere or a periodic box), following similar arguments from Cockburn, Gopalakrishnan, and Lazarov (2009), Cockburn and Gopalakrishnan (2004), and Arnold and Brezzi (1985), but tailored for the system we are considering. The main motivation here is to reveal important properties about the system in (3.42)–(3.44), and give a characterization result for the Lagrange multiplier λ_h. More specifically, we shall show the following:

- The so-called *local solvers*, which are the cell-wise local problems for the hybridizable variables $\widehat{\boldsymbol{u}}_h$ and D_h, are well-posed.
- The discrete system (3.42)–(3.44) admits a unique solution triplet $(\widehat{\boldsymbol{u}}_h, D_h, \lambda_h)$.
  - Moreover, if $(\bar{\boldsymbol{u}}_h, \bar{D}_h)$ are solution to (3.11)–(3.12), then $(\bar{\boldsymbol{u}}_h, \bar{D}_h, \lambda_h)$ is the unique solution to (3.42)–(3.44).
- The Lagrange multiplier λ_h is the unique solution to a variational problem of the form (3.51).
- The matrix operator $\boldsymbol{\mathcal{S}}$ in (3.49) is positive-definite and symmetric (without Coriolis).

The last point is particularly important when considering solver strategies for inverting $\boldsymbol{\mathcal{S}}$. Hence why we take the time to perform this analysis.

3.2.1 Local solvers

Just like before, we drop superscripts denoting time-steps n. To simplify our notation further, we also treat time-stepping coefficients as the constant $\beta = \frac{1}{2}\Delta t$. Coefficients like g, H are treated as just arbitrary positive constants. The Coriolis term will be denoted as previously: f, which, may vary in space.

Before we begin, we establish some additional notation to aid us in our discussion. With Ω denoting our computational domain and $\mathcal{T}_h = \{K\}$, a mesh of Ω with skeleton $\mathcal{E}_h$, we define the bilinear forms:

$$a_h(\boldsymbol{u}, \boldsymbol{w}) = \sum_{K \in \mathcal{T}_h} a_K(\boldsymbol{u}, \boldsymbol{w}), \quad \boldsymbol{u}, \boldsymbol{w} \in \widehat{U}_h^1 \tag{3.52}$$

$$b_h(\boldsymbol{w}, D) = \sum_{K \in \mathcal{T}_h} b_K(\boldsymbol{w}, D), \quad \boldsymbol{w} \in \widehat{U}_h^1, D \in U_h^2, \tag{3.53}$$

$$c_h(D, \phi) = \sum_{K \in \mathcal{T}_h} c_K(D, \phi), \quad D, \phi \in U_h^2, \tag{3.54}$$

$$\tag{3.55}$$

where

$$a_K(\boldsymbol{u}, \boldsymbol{w}) = \int_K \boldsymbol{w} \cdot \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} + \beta \int_K \boldsymbol{w} \cdot f \boldsymbol{u}^\perp \, \mathrm{d}\boldsymbol{x}, \tag{3.56}$$

$$b_K(\boldsymbol{w}, D) = \int_K D \nabla \cdot \boldsymbol{w} \, \mathrm{d}\boldsymbol{x}, \tag{3.57}$$

$$c_K(D, \phi) = \int_K \phi D \, \mathrm{d}\boldsymbol{x}. \tag{3.58}$$

To characterize the solution of the hybridizable problem, we start by defining a lifting operator associated with functions on the skeleton $\mathcal{E}_h$ and maps to functions on the interior of cells $K \in \mathcal{T}_h$. By way of a slight abuse of notation, we shall denote the set of all square-integrable functions on $\mathcal{E}_h$ as $L^2(\mathcal{E}_h)$. We define operators $\boldsymbol{\mathcal{Q}}^\partial$ and $\mathcal{U}^\partial$, which associates a function $\mathtt{m} \in L^2(\mathcal{E}_h)$ to functions in $\widehat{U}_h^1$ and U_h^2 respectively. The operators are defined by requiring that:

$$a_h(\boldsymbol{\mathcal{Q}}^\partial \mathtt{m}, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, \mathcal{U}^\partial \mathtt{m}) = -\sum_{K \in \mathcal{T}_h} \int_{\partial K} \mathtt{m} \widehat{\boldsymbol{w}} \cdot \boldsymbol{n} \, \mathrm{d}S, \tag{3.59}$$

$$C_H b_h(\boldsymbol{\mathcal{Q}}^\partial \mathtt{m}, \phi) + c_h(\mathcal{U}^\partial \mathtt{m}, \phi) = 0, \tag{3.60}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$, $\phi \in U_h^2$, where $C_g = \beta g$ and $C_H = \beta H$.

In a similar fashion, we define another set of local operators $\boldsymbol{\mathcal{Q}}_1^\circ$ and $\mathcal{U}_1^\circ$, this time associating a vector function $\boldsymbol{\alpha}_1 \in [L^2(\Omega)]^d$ with functions in $\widehat{U}_h^1$ and U_h^2. These operators satisfy the local equations

$$a_h(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{\alpha}_1, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, \mathcal{U}_1^\circ \boldsymbol{\alpha}_1) = \sum_{K \in \mathcal{T}_h} \int_K \boldsymbol{\alpha}_1 \cdot \widehat{\boldsymbol{w}} \, \mathrm{d}\boldsymbol{x}, \tag{3.61}$$

$$C_H b_h(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{\alpha}_1, \phi) + c_h(\mathcal{U}_1^\circ \boldsymbol{\alpha}_1, \phi) = 0, \tag{3.62}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$ and $\phi \in U_h^2$. Additionally, we have another pair of operators $\boldsymbol{\mathcal{Q}}_2^\circ$ and $\mathcal{U}_2^\circ$ which associates a function $\alpha_2 \in L^2(\Omega)$ with the functions $\boldsymbol{\mathcal{Q}}_2^\circ \alpha_2 \in \widehat{U}_h^1$ and $\mathcal{U}_2^\circ \alpha_2 \in U_h^2$ satisfying

$$a_h(\boldsymbol{\mathcal{Q}}_2^\circ \alpha_2, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, \mathcal{U}_2^\circ \alpha_2) = 0, \tag{3.63}$$

$$C_H b_h(\boldsymbol{\mathcal{Q}}_2^\circ \alpha_2, \phi) + c_h(\mathcal{U}_2^\circ \alpha_2, \phi) = \sum_{K \in \mathcal{T}_h} \int_K \alpha_2 \phi \, \mathrm{d}\boldsymbol{x} \tag{3.64}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$ and $\phi \in U_h^2$.

The linear systems (3.59)–(3.60), (3.61)–(3.62), and (3.63)–(3.64) can be evaluated in an element-by-element fashion, which is a direct consequence of the choice of function spaces and using the surjectivity of the (broken) divergence operator $\nabla_h \cdot : \widehat{U}_h^1 \to U_h^2$ when restricted to a single cell K. We call these the *local solvers* for the hybridizable compatible finite element discretization (3.42)–(3.44).

The solutions $(\boldsymbol{\mathcal{Q}}^\partial \mathtt{m}, \mathcal{U}^\partial \mathtt{m})$ are best interpreted as the functions whose restriction to a cell are the finite element approximations of the solutions $(\mathscr{Q}^\partial \mathtt{m}, \mathscr{U}^\partial \mathtt{m})$ to the local boundary value problem:

$$\mathscr{Q}^\partial \mathtt{m} + \beta \mathscr{Q}^{\partial\perp} \mathtt{m} + C_g \nabla \mathscr{U}^\partial \mathtt{m} = \mathbf{0}, \quad \text{in } K, \tag{3.65}$$

$$\mathscr{U}^\partial \mathtt{m} + C_H \nabla \cdot \mathscr{Q}^\partial \mathtt{m} = 0, \quad \text{in } K, \tag{3.66}$$

$$C_g \mathscr{U}^\partial \mathtt{m} = \mathtt{m}, \quad \text{on } \partial K, \tag{3.67}$$

for $\mathtt{m} \in L^2(\partial K)$. Similarly for $(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{\alpha}_1, \mathcal{U}_1^\circ \boldsymbol{\alpha}_1)$ and $(\boldsymbol{\mathcal{Q}}_2^\circ \alpha_2, \mathcal{U}_2^\circ \alpha_2)$, the two solution pairs are finite element approximates of $(\mathscr{Q}_1^\circ \boldsymbol{\alpha}_1, \mathscr{U}_1^\circ \boldsymbol{\alpha}_1)$ and $(\mathscr{Q}_2^\circ \alpha_2, \mathscr{U}_2^\circ \alpha_2)$, the solutions to the local Dirichlet problems:

$$\mathscr{Q}_1^\circ \boldsymbol{\alpha}_1 + \beta \mathscr{Q}_1^{\circ\perp} \boldsymbol{\alpha}_1^\perp + C_g \nabla \mathscr{U}_1^\circ \boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_1, \quad \text{in } K, \tag{3.68}$$

$$\mathscr{U}_1^\circ \boldsymbol{\alpha}_1 + C_H \nabla \cdot \mathscr{Q}_1^\circ \boldsymbol{\alpha}_1 = 0, \quad \text{in } K, \tag{3.69}$$

$$\mathscr{U}_1^\circ \boldsymbol{\alpha}_1 = 0, \quad \text{on } \partial K. \tag{3.70}$$

and

$$\mathscr{Q}_2^\circ \alpha_2 + \beta \mathscr{Q}_2^{\circ\perp} \alpha_2^\perp + C_g \nabla \mathscr{U}_2^\circ \alpha_2 = \mathbf{0}, \quad \text{in } K, \tag{3.71}$$

$$\mathscr{U}_2^\circ \alpha_2 + C_H \nabla \cdot \mathscr{Q}_2^\circ \alpha_2 = \alpha_2, \quad \text{in } K, \tag{3.72}$$

$$\mathscr{U}_2^\circ \alpha_2 = 0, \quad \text{on } \partial K. \tag{3.73}$$

The *discrete* local solvers are arrived at by discretizing (3.65)–(3.73) in each cell using the local spaces of shape functions: $U_1(K)$, $U_2(K)$. Summing over all elements produces the discrete systems (3.59)–(3.64).[5] The fact that the linear systems (3.59)–(3.60), (3.61)–(3.62), and (3.63)–(3.64) are uniquely solvable is a direct consequence of the solvability of the underlying mixed finite element discretization in each cell. We summarize this in following proposition.

Proposition 1. *(Existence and uniqueness of the local solvers). For all cells $K \in \mathcal{T}_h$ and every trace function $m \in L^2(\mathcal{E}_h)$, there is a unique set of local functions $(\boldsymbol{\mathcal{Q}}^\partial m, \mathcal{U}^\partial m)$ solving (3.59)–(3.60). Furthermore, for a pair of functions $(\boldsymbol{\alpha}_1, \alpha_2) \in [L^2(\Omega)]^d \times L^2(\Omega)$, there exists corresponding unique pairs $(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{\alpha}_1, \mathcal{U}_1^\circ \boldsymbol{\alpha}_1)$ solving (3.61)–(3.62), and $(\boldsymbol{\mathcal{Q}}_2^\circ \alpha_2, \mathcal{U}_2^\circ \alpha_2)$ solving (3.63)–(3.64).*

Proof. The fact that the local systems are well defined can be established by realizing that they are *exactly* the compatible finite element discretization of (3.65)–(3.73) in each cell. For example, in our context, the compatible method corresponds to the well-known Raviart-Thomas (RT) or Brezzi-Douglas-Marini (BDM) mixed methods (Raviart and Thomas, 1977; Brezzi, Douglas, and Marini, 1985; Brezzi et al., 1987a). Since the local solvers involve inverting the same element-wise system with different right-hand sides, it is sufficient to consider the mixed method summarized here.

[5] Note that the "broken" space $\widehat{U}_h^1$ of the hybridizable method and U_h^1 from the original compatible mixed method consist of the *same* local spaces in each cell by construction.

We shall proceed by showing both the continuous and discrete formulations on K are well-posed. For simplicity, we assume each K is a "flat" (linear) cell. The case where K is a curved approximation to a patch of a hypersurface will be discussed shortly afterwards.

As a concrete example, suppose K is an arbitrary simplex and we discretize by way of the RT method. Then resulting mixed method seeks solutions in the spaces:

$$U_1(K) = \{\boldsymbol{\tau}_h \in H(\mathrm{div}; K) : \boldsymbol{\tau}_h \in [\mathcal{P}_{q-1}(K)]^d + \boldsymbol{x}\mathcal{P}_{q-1}(K)\}, \tag{3.74}$$

$$U_2(K) = \{q_h \in L^2(K) : q_h \in \mathcal{P}_{q-1}(K)\}. \tag{3.75}$$

The local solvers within a single cell K is the following finite element problem: find $(\boldsymbol{v}_h, w_h) \in U_1 \times U_2$ satisfying

$$a_K(\boldsymbol{v}_h, \boldsymbol{\tau}_h) - C_g b_K(\boldsymbol{\tau}_h, w_h) = F(\boldsymbol{\tau}_h), \tag{3.76}$$

$$C_H b_K(\boldsymbol{v}_h, q_h) + c_K(w_h, q_h) = G(q_h), \tag{3.77}$$

for all $(\boldsymbol{\tau}_h, q_h) \in U_1 \times U_2$, where $F(\boldsymbol{\tau}_h)$ and $G(q_h)$ are linear forms containing the right-hand side data, and a_K, b_K, and c_K are as defined in (3.56)–(3.58). The system (3.76)–(3.77) defines the local solvers for the hybridizable RT (H-RT) method.

The discrete problem in (3.76)–(3.77), closely resembles a standard perturbed saddle-point system. And so, determining the solvability of the local solvers fits within the standard theory of mixed methods (Boffi, Brezzi, and Fortin, 2013, §4.3.1 and 5.5.1). There is only one primary difference, which is the inclusion of the skew-symmetric Coriolis term in a_K. Fortunately, we can show that (3.76)–(3.77) is uniquely solvable by just verifying the following conditions:

1. The bilinear form on $U_1 \times U_1$:

$$a_K(\boldsymbol{v}_h, \boldsymbol{\tau}_h) = \int_K \boldsymbol{\tau}_h \cdot \boldsymbol{v}_h \,\mathrm{d}\boldsymbol{x} + \beta \int_K \boldsymbol{\tau}_h \cdot f\boldsymbol{v}_h^{\perp} \,\mathrm{d}\boldsymbol{x}, \quad \beta > 0, \tag{3.78}$$

   is continuous on $U_1 \times U_1$, i.e., there exists a $C_1 > 0$ such that

$$|a_K(\boldsymbol{v}_h, \boldsymbol{\tau}_h)| \leq C_1 \|\boldsymbol{v}_h\|_{H(\mathrm{div})} \|\boldsymbol{\tau}_h\|_{H(\mathrm{div})}, \tag{3.79}$$

   for all $\boldsymbol{v}_h, \boldsymbol{\tau}_h \in U_1$. Additionally, we require a_K to satisfy the coercivity condition:

$$a_K(\boldsymbol{\tau}_h, \boldsymbol{\tau}_h) \geq C_2 \|\boldsymbol{\tau}_h\|^2_{H(\mathrm{div})}, \quad C_2 > 0, \tag{3.80}$$

   for all $\boldsymbol{\tau}_h \in Q_h$, where

$$Q_h := \left\{ \boldsymbol{\tau}_h \in U_1 : b_K(\boldsymbol{\tau}_h, q_h) = \int_K q_h \nabla \cdot \boldsymbol{\tau}_h \,\mathrm{d}\boldsymbol{x} = 0, \forall q_h \in U_2 \right\} \tag{3.81}$$

   (that is: $\boldsymbol{\tau}_h$ is in the kernel of the divergence operator).

2. The bilinear form b_K is continuous on $U_1 \times U_2$ and satisfies the discrete analogue of the well-known Ladyzhenskaya-Babuška-Brezzi (LBB) condition:

$$\sup_{\boldsymbol{\tau}_h \in U_1, \boldsymbol{\tau}_h \neq \mathbf{0}} \frac{b_K(\boldsymbol{\tau}_h, q_h)}{\|\boldsymbol{\tau}_h\|_{H(\mathrm{div})}} \geq C_3 \|q_h\|_{L^2}, \quad C_3 > 0, \quad \forall q_h \in U_2. \tag{3.82}$$

To show a_K satisfies item (1), we first demonstrate that a_K is continuous on $H(\mathrm{div}; K) \times H(\mathrm{div}; K)$. First note that we have the following relation for $\boldsymbol{\tau} \in H(\mathrm{div}; K)$:

$$\|f\boldsymbol{\tau}^\perp\|_{L^2} \leq \|f\|_\infty \|\boldsymbol{\tau}^\perp\|_{L^2} = \|f\|_\infty \|\boldsymbol{\tau}\|_{L^2}, \tag{3.83}$$

where $\|f\|_\infty = \inf\{\sup_{x\in K} |g(x)| : g(x) = f(x) \text{ a. e. on } K\}$[6] is the essential supremum of f. Showing that the "perp" operator $(\cdot)^\perp$ preserves $\|\cdot\|_{L^2}$ is immediate when realizing that $\boldsymbol{\tau}^\perp \cdot \boldsymbol{\tau}^\perp = |\boldsymbol{\tau}|^2$. Thus, for any $\boldsymbol{v}, \boldsymbol{\tau} \in H(\mathrm{div}; K)$, we have

$$\begin{aligned} |a_K(\boldsymbol{v}, \boldsymbol{\tau})| &= \left| \int_K \boldsymbol{\tau} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \beta \int_K \boldsymbol{\tau} \cdot f\boldsymbol{v}^\perp \, \mathrm{d}\boldsymbol{x} \right| \\ &\leq \left| \int_K \boldsymbol{\tau} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \right| + \beta \left| \int_K \boldsymbol{\tau} \cdot f\boldsymbol{v}^\perp \, \mathrm{d}\boldsymbol{x} \right| \\ &\leq \|\boldsymbol{\tau}\|_{L^2} \|\boldsymbol{v}\|_{L^2} + \beta \|\boldsymbol{\tau}\|_{L^2} \|f\|_\infty \|\boldsymbol{v}\|_{L^2} \\ &\leq (1 + \beta \|f\|_\infty) \|\boldsymbol{\tau}\|_{H(\mathrm{div})} \|\boldsymbol{v}\|_{H(\mathrm{div})}, \end{aligned} \tag{3.84}$$

where the last inequality comes from two applications of the Cauchy-Schwarz inequality, (3.83), and using the fact that $\|\cdot\|_{H(\mathrm{div})}$ bounds the L^2-norm. Equation (3.84) implies a continuity constant $C_1 = (1 + \beta\|f\|_\infty)$, which therefore implies a_K is continuous on $H(\mathrm{div}; K)$ for bounded $\|f\|_\infty$. Indeed, the Coriolis parameter in geophysical settings is *always* bounded (see (2.12)). Since $U_1 \subset H(\mathrm{div}; K)$, we can conclude that the bilinear form a_K satisfies (3.79).

To show (3.80), let us start by defining the operator B mapping $H(\mathrm{div}; K)$ into the dual space $L^2(K)'$ (space of linear functionals on $L^2(K)$). The mapping B is defined in the usual way: $\langle B\boldsymbol{v}, w\rangle_{L^2(K)' \times L^2(K)} := b_K(\boldsymbol{v}, w)$ for all $(\boldsymbol{v}, w) \in H(\mathrm{div}; K) \times L^2(K)$. Then the divergence-free subspace of $H(\mathrm{div}; K)$ can be written compactly as:

$$Q = \ker B. \tag{3.85}$$

Observe that, for all $\boldsymbol{w} \in H(\mathrm{div}; K)$, we have

$$a_K(\boldsymbol{w}, \boldsymbol{w}) = \int_K \boldsymbol{w} \cdot \boldsymbol{w} \, \mathrm{d}\boldsymbol{x} + \beta \underbrace{\int_K \boldsymbol{w} \cdot f\boldsymbol{w}^\perp \, \mathrm{d}\boldsymbol{x}}_{=0} = \int_K |\boldsymbol{w}|^2 \, \mathrm{d}\boldsymbol{x} = \|\boldsymbol{w}\|_{L^2}^2. \tag{3.86}$$

And so, for all $\boldsymbol{w} \in Q$:

$$\|\boldsymbol{w}\|_{H(\mathrm{div})}^2 = \|\boldsymbol{w}\|_{L^2}^2 + \underbrace{\|\nabla \cdot \boldsymbol{w}\|_{L^2}^2}_{=0} = \|\boldsymbol{w}\|_{L^2}^2. \tag{3.87}$$

Hence, (3.87) and (3.86) implies a_K is coercive on Q, with coercivity constant $C_2 = 1$.

Now, let us restrict our attention to the discrete setting. We denote the discrete version of B as $B_h : U_1 \to U_2'$, defined via $\langle B_h \boldsymbol{v}_h, w_h\rangle_{U_2' \times U_2} := b_K(\boldsymbol{v}_h, w_h)$ for all $(\boldsymbol{v}_h, w_h) \in U_1 \times U_2$. Then, as before, we can write (3.81) as:

$$Q_h = \ker B_h. \tag{3.88}$$

[6] The concept of a property holding "almost everywhere" (written as "a.e." or "p.p." for the French phrase: *presque partout*) is ubiquitous in measure theory. In a technical sense, a property is said to hold a.e. if the set where the property does *not* hold has *zero measure*. In our context, the measure in question is the *Lebesgue measure*. The curious reader may consult any standard textbook on measure theory or real analysis for more details.

Note that, in general, $Q_h \subset Q$ is not necessarily true. However, by virtue of the compatible finite element de-Rham complex (2.198), we have the following property for the RT method:

$$\nabla \cdot V_h = W_h \implies B_h = B|_{V_h}. \tag{3.89}$$

That is: B_h is the restriction of the divergence operator to the finite-dimensional space V_h (Raviart and Thomas, 1977). Hence, we have the particularly interesting case where

$$Q_h = \ker B_h \subseteq \ker B = Q. \tag{3.90}$$

While (3.90) is not *strictly* necessary to ensure solvability of the mixed method, it is an incredibly desirable property. In particular, (3.90) implies immediately that a_K is coercive on Q_h.

For item (2): the bilinear form $b_K(\boldsymbol{v}, w)$, for $\boldsymbol{v} \in H(\mathrm{div}; K)$ and $w \in L^2(K)$ is well-understood. It is clearly continuous on $H(\mathrm{div}; K) \times L^2(K)$:

$$|b_K(\boldsymbol{v}, w)| = \left| \int_K w \nabla \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \right| \leq \|\nabla \cdot \boldsymbol{v}\|_{L^2} \|w\|_{L^2} \leq \|\boldsymbol{v}\|_{H(\mathrm{div})} \|w\|_{L^2}. \tag{3.91}$$

The result follows for the discrete setting. We also have that $b_K(\boldsymbol{v}, w)$ satisfies the continuous analogue of the LBB condtion. Namely, there exists a constant $\xi > 0$ satisfying for all $w \in L^2(K)$:

$$\sup_{\boldsymbol{v} \in H(\mathrm{div}; K), \boldsymbol{v} \neq \boldsymbol{0}} \frac{b_K(\boldsymbol{v}, w)}{\|\boldsymbol{v}\|_{H(\mathrm{div})}} \geq \xi \|w\|_{L^2}. \tag{3.92}$$

This is a direct application of Corollary 4.1.1 of Boffi, Brezzi, and Fortin (2013). Given that the continuous analogue of b_K satisfies the LBB condition, we quickly show this is true in the discrete setting *precisely* due to the compatible finite element framework.

Recall from (2.198) that we have the existence of a bounded projection operator $\pi_{U_1} : H(\mathrm{div}; K) \to U_1$, with $\|\pi_{U_1} \boldsymbol{v}\|_{H(\mathrm{div})} \leq C_\pi \|\boldsymbol{v}\|_{H(\mathrm{div})}$, $\boldsymbol{v} \in H(\mathrm{div}; K)$. Or equivalently, for any $\boldsymbol{v} \in H(\mathrm{div}; K)$:

$$\|\boldsymbol{v}\|_{H(\mathrm{div})} \geq \frac{1}{C_\pi} \|\pi_{U_1} \boldsymbol{v}\|_{H(\mathrm{div})} = \frac{1}{C_\pi} \|\boldsymbol{v}_h\|_{H(\mathrm{div})}. \tag{3.93}$$

For the RT method, the operator π_{U_1} is defined through the equations on K:

$$\int_K (\pi_{U_1} \boldsymbol{v} - \boldsymbol{v}) \cdot \boldsymbol{\tau_h} \, \mathrm{d}\boldsymbol{x} = 0, \quad \forall \boldsymbol{\tau}_h \in [\mathcal{P}_{q-1}(K)]^d, \tag{3.94}$$

$$\int_e (\pi_{U_1} \boldsymbol{v} - \boldsymbol{v}) \cdot \boldsymbol{n} \gamma_h \, \mathrm{d}S = 0, \quad \forall \gamma_h \in \mathcal{P}_q(e), \tag{3.95}$$

for all facets e of K. Together with the commuting diagram property (2.198), π_{U_1} satisfies the following property (Boffi, Brezzi, and Fortin, 2013, Proposition 5.1.2):

$$b_K(\boldsymbol{v} - \pi_{U_1} \boldsymbol{v}, w_h) = 0, \quad \forall w_h \in U_2, \tag{3.96}$$

which can be easily verified by direct calculation:

$$\begin{aligned}\int_K w_h \nabla \cdot \pi_{U_1} \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} &= \int_{\partial K} w_h \pi_{U_1} \boldsymbol{v} \cdot \boldsymbol{n} \,\mathrm{d}S - \int_K \nabla w_h \pi_{U_1} \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} \\ &= \int_{\partial K} w_h \boldsymbol{v} \cdot \boldsymbol{n} \,\mathrm{d}S - \int_K \nabla w_h \boldsymbol{v} \,\mathrm{d}\boldsymbol{x} = \int_K w_h \nabla \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.97}$$

for all $w_h \in \mathcal{P}_q(K)$.

Therefore, for $w_h \in U_2$ arbitrary but fixed, (3.92), (3.93), and (3.96) implies:

$$\begin{aligned}\xi \|w_h\|_{L^2} &\leq \sup_{\boldsymbol{v} \in H(\mathrm{div};K), \boldsymbol{v} \neq \boldsymbol{0}} \frac{b_K(\boldsymbol{v}, w_h)}{\|\boldsymbol{v}\|_{H(\mathrm{div})}} = \sup_{\boldsymbol{v} \in H(\mathrm{div};K), \boldsymbol{v} \neq \boldsymbol{0}} \frac{b_K(\pi_{U_1}\boldsymbol{v}, w_h)}{\|\boldsymbol{v}\|_{H(\mathrm{div})}} \\ &\leq C_\pi \sup_{\boldsymbol{v} \in H(\mathrm{div};K), \boldsymbol{v} \neq \boldsymbol{0}} \frac{b_K(\pi_{U_1}\boldsymbol{v}, w_h)}{\|\pi_{U_1}\boldsymbol{v}\|_{H(\mathrm{div})}} \leq C_\pi \sup_{\boldsymbol{v}_h \in U_1, \boldsymbol{v}_h \neq \boldsymbol{0}} \frac{b_K(\boldsymbol{v}_h, w_h)}{\|\boldsymbol{v}_h\|_{H(\mathrm{div})}}, \end{aligned} \tag{3.98}$$

hence,

$$\sup_{\boldsymbol{v}_h \in U_1, \boldsymbol{v}_h \neq \boldsymbol{0}} \frac{b_K(\boldsymbol{v}_h, w_h)}{\|\boldsymbol{v}_h\|_{H(\mathrm{div})}} \geq \frac{\xi}{C_\pi} \|w_h\|_{L^2}. \tag{3.99}$$

We can now conclude that the mixed system (3.76)–(3.77) is well-posed.

The discussion here for the RT method can be applied to *any* compatible finite element formulation. In particular, the proof here relies on the commuting diagram property (2.198). Methods that fall within this category include the BDM method (Brezzi, Douglas, and Marini, 1985; Brezzi et al., 1987a) and the RT method on quadrilateral cells (RTCF). Thus, we can conclude that the local solvers of the hybridizable mixed methods (H-RT, H-BDM, and H-RTCF) for (3.42)–(3.44) are well-defined. □

The proof of Proposition 1 only considered the case where the cells K defining the local finite element complex $H^1(K) \xrightarrow{\nabla^\perp} H(\mathrm{div};K) \xrightarrow{\nabla\cdot} L^2(K)$ is flat. In Section 2.4.6, we summarized the framework of Holst and Stern (2012) which allows us to extend our results to mixed formulations on isoparametric cells (see also Natale, Shipton, and Cotter (2016)). In particular, the well-posedness of the mixed problem (3.76)–(3.77), where K is a curved approximation to a patch of a smooth embedded manifold (via a piece-wise polynomial representation) is a direct consequence of Theorems 3.5 and 3.9 of Holst and Stern (2012). Therefore, everything discussed thus far can be applied in this more general context. For the rest of this section, we require no explicit assumptions on K or the mesh $\mathcal{T}_h$ beyond shape-regularity (that is, all facets of the cells K have an area bounded away from zero).

Having established the solvability of the local solvers, we now turn our attention to the full discrete system arising from the hybridization of a compatible finite element discretization.

3.2.2 Solvability of the discrete hybridizable system

We now demonstrate that the linear system (3.45) is uniquely solvable.

Proposition 2. *There exists unique solutions* $(\widehat{\boldsymbol{u}}_h, D_h, \lambda_h) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$ *satisfying the discrete formulation* (3.42)–(3.44).

Proof. Since we are in a finite-dimensional setting, it is sufficient to prove unique solvability by a standard Fredholm argument. Namely, when the right-hand side data is zero, this implies the solutions are the trivial ones: $\widehat{\boldsymbol{u}}_h = \mathbf{0}$, $D_h = 0$, $\lambda_h = 0$. The homogenous system is summarized by the equations: find $(\widehat{\boldsymbol{u}}_h, D_h, \lambda_h) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$ satisfying

$$a_h(\widehat{\boldsymbol{u}}_h, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, D_h) + \sum_{K\in\mathcal{T}_h} \int_{\partial K} \lambda_h \widehat{\boldsymbol{w}} \cdot \boldsymbol{n}\, \mathrm{d}S = 0, \tag{3.100}$$

$$C_H b_h(\widehat{\boldsymbol{u}}_h, \phi) + c_h(D_h, \phi) = 0, \tag{3.101}$$

$$\sum_{K\in\mathcal{T}_h} \int_{\partial K} \gamma \widehat{\boldsymbol{u}}_h \cdot \boldsymbol{n}\, \mathrm{d}S = 0, \tag{3.102}$$

for all $(\widehat{\boldsymbol{w}}, \phi, \gamma) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$.

Since (3.100)–(3.102) is true for all test functions, choosing $\widehat{\boldsymbol{w}} = H\widehat{\boldsymbol{u}}_h$, $\phi = gD_h$, and adding the first two equations gives:

$$H a_h(\widehat{\boldsymbol{u}}_h, \widehat{\boldsymbol{u}}_h) + g c_h(D_h, D_h) + H \sum_{K\in\mathcal{T}_h} \int_{\partial K} \lambda_h \widehat{\boldsymbol{u}}_h \cdot \boldsymbol{n}\, \mathrm{d}S = 0. \tag{3.103}$$

Since the jump condition (3.102) is true for all $\gamma \in U_h^{\mathrm{tr}}$ (in particular, $\gamma = \lambda_h$), we can rewrite (3.103) as:

$$a_h(\widehat{\boldsymbol{u}}_h, \widehat{\boldsymbol{u}}_h) + \frac{g}{H} c_h(D_h, D_h) = 0. \tag{3.104}$$

Using the definition of a_h and noting that $\widehat{\boldsymbol{u}}_h \cdot \widehat{\boldsymbol{u}}_h^{\perp} = 0$, we have

$$a_h(\widehat{\boldsymbol{u}}_h, \widehat{\boldsymbol{u}}_h) + \frac{g}{H} c_h(D_h, D_h) = \sum_{K\in\mathcal{T}_h} \int_K |\widehat{\boldsymbol{u}}_h|^2\, \mathrm{d}\boldsymbol{x} + \frac{g}{H} \sum_{K\in\mathcal{T}_h} \int_K |D_h|^2\, \mathrm{d}\boldsymbol{x} = 0, \tag{3.105}$$

or equivalently:

$$\|\widehat{\boldsymbol{u}}_h\|_{\widehat{U}_h^1}^2 + \frac{g}{H} \|D_h\|_{U_h^2}^2 = 0 \implies \|\widehat{\boldsymbol{u}}_h\|_{\widehat{U}_h^1}^2 = -\frac{g}{H} \|D_h\|_{U_h^2}^2, \tag{3.106}$$

where $\|\cdot\|_{\widehat{U}_h^1}$ and $\|\cdot\|_{U_h^2}$ denote the usual "broken" L^2-norms on $\widehat{U}_h^1$ and U_h^2 respectively:

$$\|\widehat{\boldsymbol{u}}_h\|_{\widehat{U}_h^1} := \left(\sum_{K\in\mathcal{T}_h} \|\widehat{\boldsymbol{u}}_h\|_{L^2(K)^d}^2 \right)^{1/2}, \quad \|D_h\|_{U_h^2} := \left(\sum_{K\in\mathcal{T}_h} \|D_h\|_{L^2(K)}^2 \right)^{1/2}. \tag{3.107}$$

Thus, we can conclude that $\widehat{\boldsymbol{u}}_h = \mathbf{0}$ and $D_h = 0$. Finally, after substituting the trivial solutions $\widehat{\boldsymbol{u}}_h$ and D_h into (3.100), we get

$$\sum_{K\in\mathcal{T}_h} \int_{\partial K} \lambda_h \widehat{\boldsymbol{w}} \cdot \boldsymbol{n}\, \mathrm{d}S = 0, \tag{3.108}$$

for all $\widehat{\boldsymbol{w}} \in \widehat{U}_h^1$. Lemma 3 allows us to conclude that λ_h must also be zero. □

We now state a key result which establishes the connection between the solutions of the hybridizable system (3.42)–(3.44) with the solutions of the original compatible finite element discretization (3.11)–(3.12).

Proposition 3. *(Equivalence of solutions): Let $(\boldsymbol{u}_h, D_h) \in U_h^1 \times U_h^2$ be the unique solution pair of the compatible finite element discretization:*

$$a_h(\boldsymbol{u}_h, \boldsymbol{w}) - C_g b_h(\boldsymbol{w}, D_h) = R_{\boldsymbol{u}}[\boldsymbol{w}], \tag{3.109}$$
$$C_H b_h(\boldsymbol{u}_h, \phi) + c_h(D_h, \phi) = R_D[\phi], \tag{3.110}$$

for all $(\boldsymbol{w}, \phi) \in U_h^1 \times U_h^2$, where $R_{\boldsymbol{u}}$ and R_D are as defined in (3.16)–(3.17).

Then the solution triple $(\boldsymbol{u}_h, D_h, \lambda_h)$ is the unique solution to the hybridizable problem: find $(\boldsymbol{u}_h, D_h, \lambda_h) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$

$$a_h(\boldsymbol{u}_h, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, D_h) + \sum_{K \in \mathcal{T}_h} \int_{\partial K} \lambda_h \widehat{\boldsymbol{w}} \cdot \boldsymbol{n} \, \mathrm{d}S = R_{\boldsymbol{u}}[\widehat{\boldsymbol{w}}], \tag{3.111}$$
$$C_H b_h(\boldsymbol{u}_h, \phi) + c_h(D_h, \phi) = R_D[\phi], \tag{3.112}$$
$$\sum_{K \in \mathcal{T}_h} \int_{\partial K} \gamma \boldsymbol{u}_h \cdot \boldsymbol{n} \, \mathrm{d}S = 0, \tag{3.113}$$

for all $(\widehat{\boldsymbol{w}}, \phi, \gamma) \in \widehat{U}_1 \times U_2 \times U_1^{\mathrm{tr}}$.

Proof. From Proposition 2, we know the hybridizable problem has a unique solution triple. Let $(\widehat{\boldsymbol{u}}_h, \widehat{D}_h, \lambda_h) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$ denote the solution triple for (3.111)–(3.113). Since $U_h^1 \subset \widehat{U}_h^1$, set $\widehat{\boldsymbol{w}} = \boldsymbol{w} \in U_h^1$. Then clearly (3.111) still holds for all such test functions. Moreover, the normal components of $\boldsymbol{w}$ are continuous over the entire mesh skeleton $\mathcal{E}_h$, i.e., for all $\lambda_h \in U_h^{\mathrm{tr}}$ and $\boldsymbol{w} \in U_h^1$:

$$\sum_{K \in \mathcal{T}_h} \int_{\partial K} \lambda_h \boldsymbol{w} \cdot \boldsymbol{n} \, \mathrm{d}S = \sum_{e \in \mathcal{E}_h} \int_e \lambda_h [\![\boldsymbol{w}]\!] \, \mathrm{d}S = 0. \tag{3.114}$$

As a result, (3.111) simplifies to

$$a_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{w}) - C_g b_h(\boldsymbol{w}, \widehat{D}_h) = R_{\boldsymbol{u}}[\boldsymbol{w}], \quad \forall \boldsymbol{w} \in U_h^1. \tag{3.115}$$

Now, using the jump condition (3.113), we can conclude from Lemma 2 that $\widehat{\boldsymbol{u}}_h \in U_h^1$. Thus, we have that $(\widehat{\boldsymbol{u}}_h, \widehat{D}_h) \in U_h^1 \times U_h^2$ solves the compatible finite element problem:

$$a_h(\widehat{\boldsymbol{u}}_h, \boldsymbol{w}) - C_g b_h(\boldsymbol{w}, \widehat{D}_h) = R_{\boldsymbol{u}}[\boldsymbol{w}], \tag{3.116}$$
$$C_H b_h(\widehat{\boldsymbol{u}}_h, \phi) + c_h(\phi, \widehat{D}_h) = R_D[\phi], \tag{3.117}$$

for all $(\boldsymbol{w}, \phi) \in U_h^1 \times U_h^2$. By uniqueness of the original mixed method, we can conclude that $\widehat{\boldsymbol{u}}_h = \boldsymbol{u}_h$ and $\widehat{D}_h = D_h$. □

By virtue of Proposition 3, the hybridizable formulation (3.111)–(3.113) can be interpreted as a "reformulation" of (3.109)–(3.110). We henceforth drop the $\widehat{\cdot}$ in our notation for the velocity and depth fields. As we have previously stated in Section 3.1.2, using a hybridizable method has significant computational advantages.

In a hybridizable discretization, we can employ local static condensation procedures to eliminate $\boldsymbol{u}_h$ and D_h from the global equations and assemble a sparse equation for the Lagrange multiplier. Local elimination is not possible in the original compatible finite element formulation, which requires extensive (often expensive) preconditioning to iteratively invert the resulting global matrix system. The trade-off, when using hybridization, is avoiding the inversion of globally dense operators at the expense

of doing local inversions of dense matrix systems. Fortunately, these local linear systems are quite small. The dimension reduction of the global problem to a condensed system for the Lagrange multiplier is significant. We shall elaborate further on the implementation aspects in Chapter 4.

The new computational bottleneck in the solution procedure for solving (3.111)–(3.113) is now the global inversion of the Lagrange multiplier system. Therefore, it will be useful to develop a characterization of the unknown λ_h which reveals some innate structure of the new discrete system. Specifically, we shall show that λ_h is actually the solution to a discrete variational problem.

3.2.3 Characterization of the hybridizable solutions

Before we state the main result, we shall proceed by establishing key identities and relations in a similar manner to Cockburn, Gopalakrishnan, and Lazarov (2009). Using the local solvers: (3.59)–(3.60), (3.61)–(3.62), and (3.63)–(3.64), we start with the following lemma.

Lemma 4. *For any $m, \gamma \in L^2(\mathcal{E}_h)$, $\boldsymbol{\alpha}_1 \in [L^2(\Omega)]^d$, and $\alpha_2 \in L^2(\Omega)$, the following identities hold:*

$$(i) \quad -\int_{\mathcal{E}_h} \gamma[\![\boldsymbol{\mathcal{Q}}^\partial m]\!]\,\mathrm{d}S = \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial\gamma\cdot\boldsymbol{\mathcal{Q}}^\partial m\,\mathrm{d}\boldsymbol{x} - \beta\int_{\mathcal{T}_h}\boldsymbol{\mathcal{Q}}^\partial\gamma\cdot f\boldsymbol{\mathcal{Q}}^\partial m^\perp\,\mathrm{d}\boldsymbol{x} + \frac{g}{H}\int_{\mathcal{T}_h}\mathcal{U}^\partial\gamma\mathcal{U}^\partial m\,\mathrm{d}\boldsymbol{x},$$

$$(ii) \quad -\int_{\mathcal{E}_h} \gamma[\![\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1]\!]\,\mathrm{d}S = \int_{\mathcal{T}_h}\boldsymbol{\mathcal{Q}}^\partial\gamma\cdot\boldsymbol{\alpha}_1\,\mathrm{d}\boldsymbol{x} - 2\beta\int_{\mathcal{T}_h}\boldsymbol{\mathcal{Q}}^\partial\gamma\cdot f\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1^\perp\,\mathrm{d}\boldsymbol{x},$$

$$(iii) \quad -\int_{\mathcal{E}_h} \gamma[\![\boldsymbol{\mathcal{Q}}_2^\circ\alpha_2]\!]\,\mathrm{d}S = -\frac{g}{H}\int_{\mathcal{T}_h}\mathcal{U}^\partial\gamma\alpha_2\,\mathrm{d}\boldsymbol{x} - 2\beta\int_{\mathcal{T}_h}\boldsymbol{\mathcal{Q}}^\partial\gamma\cdot f\boldsymbol{\mathcal{Q}}_2^\circ\alpha_2^\perp\,\mathrm{d}\boldsymbol{x}.$$

Proof. We prove each identity sequentially. To show (i), the calculation is straightforward. First, we take $\mathtt{m} = \gamma$ and $\widehat{\boldsymbol{w}} = H\boldsymbol{\mathcal{Q}}^\partial\mathtt{m}$ in (3.59). This gives

$$Ha_h(\boldsymbol{\mathcal{Q}}^\partial\gamma, \boldsymbol{\mathcal{Q}}^\partial\mathtt{m}) - HC_g b_h(\boldsymbol{\mathcal{Q}}^\partial\mathtt{m}, \mathcal{U}^\partial\gamma) = -H\int_{\mathcal{E}_h}\gamma[\![\boldsymbol{\mathcal{Q}}^\partial\mathtt{m}]\!]\,\mathrm{d}S. \tag{3.118}$$

Now take $\phi = g\mathcal{U}^\partial\gamma$ in (3.60) to get the second relation:

$$gC_H b_h(\boldsymbol{\mathcal{Q}}^\partial\mathtt{m}, \mathcal{U}^\partial\gamma) + gc_h(\mathcal{U}^\partial\mathtt{m}, \mathcal{U}^\partial\gamma) = 0. \tag{3.119}$$

Adding the two equations gives identity (i).

To show (ii), take $\mathtt{m} = \gamma$ and $\phi = g\mathcal{U}_1^\circ\boldsymbol{\alpha}_1$ in (3.60), $\widehat{\boldsymbol{w}} = H\boldsymbol{\mathcal{Q}}^\partial\gamma$ in (3.61), and add the two results together to get:

$$Ha_h(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1, \boldsymbol{\mathcal{Q}}^\partial\gamma) + gc_h(\mathcal{U}^\partial\gamma, \mathcal{U}_1^\circ\boldsymbol{\alpha}_1) = H\int_{\mathcal{T}_h}\boldsymbol{\mathcal{Q}}^\partial\gamma\cdot\boldsymbol{\alpha}_2\,\mathrm{d}\boldsymbol{x}. \tag{3.120}$$

Next, set $\mathtt{m} = \gamma$ and $\widehat{\boldsymbol{w}} = H\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1$ in (3.59), $\phi = g\mathcal{U}^\partial\gamma$ in (3.62), and add the two expressions. This produces:

$$Ha_h(\boldsymbol{\mathcal{Q}}^\partial\gamma, \boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1) + gc_h(\mathcal{U}_1^\circ\boldsymbol{\alpha}_1, \mathcal{U}^\partial\gamma) = -H\int_{\mathcal{E}_h}\gamma[\![\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1]\!]\,\mathrm{d}S. \tag{3.121}$$

Then, to get (ii), subtract (3.120) from (3.121) and use the fact that the Coriolis term is skew-symmetric:

$$\begin{aligned} a_h(\boldsymbol{\mathcal{Q}}^\partial\gamma, \boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1) - a_h(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1, \boldsymbol{\mathcal{Q}}^\partial\gamma) &= \beta\int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1 \cdot f\boldsymbol{\mathcal{Q}}^\partial\gamma^\perp \,\mathrm{d}\boldsymbol{x} \\ &\quad - \beta\int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial\gamma \cdot f\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1^\perp \,\mathrm{d}\boldsymbol{x} \\ &= -2\beta\int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial\gamma \cdot f\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{\alpha}_1^\perp \,\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{3.122}$$

Finally, we prove the last identity (iii). Performing similar algebraic manipulations, we take $\mathtt{m} = \gamma$ and $\phi = g\mathcal{U}_2^\circ\alpha_2$ in (3.60), $\widehat{\boldsymbol{w}} = H\boldsymbol{\mathcal{Q}}^\partial\gamma$ in (3.63), then sum the expressions:

$$Ha_h(\boldsymbol{\mathcal{Q}}_2^\circ\alpha_2, \boldsymbol{\mathcal{Q}}^\partial\gamma) + gc_h(\mathcal{U}^\partial\gamma, \mathcal{U}_2^\circ\alpha_2) = 0. \tag{3.123}$$

With $\mathtt{m} = \gamma$ and $\widehat{\boldsymbol{w}} = \boldsymbol{\mathcal{Q}}_2^\circ\alpha_2$ in (3.59), $\phi = g\mathcal{U}^\partial\gamma$ in (3.64), add the equations together to obtain:

$$\begin{aligned} Ha_h(\boldsymbol{\mathcal{Q}}^\partial\gamma, \boldsymbol{\mathcal{Q}}_2^\circ\alpha_2) + gc_h(\mathcal{U}_2^\circ\alpha_2, \mathcal{U}^\partial\gamma) &= -H\int_{\mathcal{E}_h} \gamma[\![\boldsymbol{\mathcal{Q}}_2^\circ\alpha_2]\!] \,\mathrm{d}S \\ &\quad + g\int_{\mathcal{T}_h} \mathcal{U}^\partial\gamma\alpha_2 \,\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{3.124}$$

Thus, (iii) is obtained by simply subtracting (3.123) from (3.124) and using the skew-symmetry property of the Coriolis term in the definition of a_h. □

Having established the identities in Lemma 4, we now state the main result of this section.

Theorem 1. *(Characterization of $\boldsymbol{u}_h$, D_h, and λ_h): Let $(\boldsymbol{u}_h, D_h, \lambda_h) \in \widehat{U}_h^1 \times U_h^2 \times U_h^{\mathrm{tr}}$ be the unique solution triple for the hybridizable compatible finite element method (3.111)–(3.113). Let $\boldsymbol{r}_1^h$ and r_2^h denote the residual functions in (3.18)–(3.19). Then,*

$$\boldsymbol{u}_h = \boldsymbol{\mathcal{Q}}^\partial\lambda_h + \boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h, \tag{3.125}$$

$$D_h = \mathcal{U}^\partial\lambda_h + \mathcal{U}_1^\circ\boldsymbol{r}_1^h + \mathcal{U}_2^\circ r_2^h, \tag{3.126}$$

with the Lagrange multiplier λ_h being the unique solution to the discrete variational problem: find $\lambda_h \in U_h^{\mathrm{tr}}$ satisfying

$$s_h(\lambda_h, \gamma) = r_h(\gamma), \quad \forall\gamma \in U_h^{\mathrm{tr}}, \tag{3.127}$$

where

$$s_h(\lambda_h, \gamma) := \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial\gamma \cdot \left(\boldsymbol{\mathcal{Q}}^\partial\lambda_h - \beta f\boldsymbol{\mathcal{Q}}^\partial\lambda_h^\perp\right) \mathrm{d}\boldsymbol{x} + \frac{g}{H}\int_{\mathcal{T}_h} \mathcal{U}^\partial\gamma\mathcal{U}^\partial\lambda_h \,\mathrm{d}\boldsymbol{x}, \tag{3.128}$$

$$\begin{aligned} r_h(\gamma) := &\int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial\gamma \cdot \left(2\beta f\left(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h\right)^\perp - \boldsymbol{r}_1^h\right) \mathrm{d}\boldsymbol{x} \\ &+ \frac{g}{H}\int_{\mathcal{T}_h} \mathcal{U}^\partial\gamma r_2^h \,\mathrm{d}\boldsymbol{x}. \end{aligned} \tag{3.129}$$

Proof. To show (3.125)–(3.126), we simply use linearity of the problem and the definition of the local solvers: (3.59)–(3.60), (3.61)–(3.62), and (3.63)–(3.64). Combining

all three systems and setting $\mathtt{m} = \lambda_h$ in (3.59)–(3.60), we obtain the following system:

$$a_h(\widetilde{\boldsymbol{u}}_h, \widehat{\boldsymbol{w}}) - C_g b_h(\widehat{\boldsymbol{w}}, \widetilde{D}_h) = R_{\boldsymbol{u}}[\widehat{\boldsymbol{w}}] - \sum_{e\in\mathcal{E}_h} \int_e \lambda_h [\![\widehat{\boldsymbol{w}}]\!]\, \mathrm{d}S, \tag{3.130}$$

$$C_H b_h(\widetilde{\boldsymbol{u}}_h, \phi) + c_h(\widetilde{D}_h, \phi) = R_D[\phi], \tag{3.131}$$

for all $(\widehat{\boldsymbol{w}}, \phi) \in \widehat{U}_h^1 \times U_h^2$, where

$$(\widetilde{\boldsymbol{u}}_h, \widetilde{D}_h) = (\boldsymbol{\mathcal{Q}}^\partial \lambda_h, \mathcal{U}^\partial \lambda_h) + (\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{r}_1^h, \mathcal{U}_1^\circ \boldsymbol{r}_1^h) + (\boldsymbol{\mathcal{Q}}_2^\circ r_2^h, \mathcal{U}_2^\circ r_2^h). \tag{3.132}$$

Since (3.130)–(3.130) holds for all $(\widehat{\boldsymbol{w}}, \phi) \in \widehat{U}_h^1 \times U_h^2$, it also holds for all $(\boldsymbol{w}, \phi) \in U_h^1 \times U_h^2$, where $U_h^1 \subset \widehat{U}_h^1$ is the $H(\mathrm{div})$ subspace of $\widehat{U}_h^1$. Then $(\widetilde{\boldsymbol{u}}_h, \widetilde{D}_h)$ are solutions to the compatible finite element discretization. By Proposition 3, $(\widetilde{\boldsymbol{u}}_h, \widetilde{D}_h)$ are also solutions to the extended problem (3.111)–(3.113), with

$$\int_{\mathcal{E}_h} \gamma [\![\widetilde{\boldsymbol{u}}_h]\!]\, \mathrm{d}S = \int_{\mathcal{E}_h} \gamma [\![\boldsymbol{\mathcal{Q}}^\partial \lambda_h + \boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h]\!]\, \mathrm{d}S = 0, \quad \forall \gamma \in U_h^{\mathrm{tr}}. \tag{3.133}$$

Then $(\widetilde{\boldsymbol{u}}_h, \widetilde{D}_h, \lambda_h)$ satisfy all equations of the hybridizable method. By virtue of Proposition 2, the solution triple of the equations (3.111)–(3.113) is unique and we may write: $\widetilde{\boldsymbol{u}}_h = \boldsymbol{u}_h$ and $\widetilde{D}_h = D_h$.

To show (3.127), we use Lemma 4. Take $\boldsymbol{\alpha}_1 = \boldsymbol{r}_1^h$, $\alpha_2 = r_2^h$, and sum the identities (i)–(iii) to get an equation for any $\mathtt{m} \in U_h^{\mathrm{tr}}$:

$$\begin{aligned} -\int_{\mathcal{E}_h} \gamma [\![\boldsymbol{\mathcal{Q}}^\partial \mathtt{m} + \boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h]\!]\, \mathrm{d}S = &\int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot \boldsymbol{\mathcal{Q}}^\partial \mathtt{m}\, \mathrm{d}\boldsymbol{x} - \beta \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot f \boldsymbol{\mathcal{Q}}^\partial \mathtt{m}^\perp\, \mathrm{d}\boldsymbol{x} \\ &+ \frac{g}{H} \int_{\mathcal{T}_h} \mathcal{U}^\partial \gamma \mathcal{U}^\partial \mathtt{m}\, \mathrm{d}\boldsymbol{x} - \frac{g}{H} \int_{\mathcal{T}_h} \mathcal{U}^\partial \gamma r_2^h\, \mathrm{d}\boldsymbol{x} \\ &+ \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot \boldsymbol{r}_1^h\, \mathrm{d}\boldsymbol{x} \\ &- 2\beta \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot f \left(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h \right)^\perp \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.134}$$

for all $\gamma \in U_h^{\mathrm{tr}}$. Taking $\mathtt{m} = \lambda_h$ asserts that equation (3.134) is equal to zero via the jump condition (3.133). Therefore, we have λ_h satisfying

$$\begin{aligned} \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot \boldsymbol{\mathcal{Q}}^\partial \lambda_h\, \mathrm{d}\boldsymbol{x} - \beta \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot f \boldsymbol{\mathcal{Q}}^\partial \lambda_h^\perp\, \mathrm{d}\boldsymbol{x} & \\ + \frac{g}{H} \int_{\mathcal{T}_h} \mathcal{U}^\partial \gamma \mathcal{U}^\partial \lambda_h\, \mathrm{d}\boldsymbol{x} = \frac{g}{H} \int_{\mathcal{T}_h} \mathcal{U}^\partial \gamma r_2^h\, \mathrm{d}\boldsymbol{x} & \\ + 2\beta \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot f \left(\boldsymbol{\mathcal{Q}}_1^\circ \boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h \right)^\perp \mathrm{d}\boldsymbol{x} & \\ - \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot \boldsymbol{r}_1^h\, \mathrm{d}\boldsymbol{x}, & \end{aligned} \tag{3.135}$$

for all $\gamma \in U_h^{\mathrm{tr}}$. The fact that λ_h is the unique trace function satisfying (3.135) is a direct consequence of Proposition 2. From the definition of $s_h(\lambda_h, \gamma)$ and $r_h(\gamma)$ in (3.128) and (3.129), the result follows. □

Theorem 1 is a variation of a similar result, presented by Cockburn, Gopalakrishnan, and Lazarov (2009, Theorem 2.1), which establishes a characterization of λ_h for the

hybridization of mixed methods for second-order elliptic equations. Indeed, when no Coriolis is present ($f = 0$), the linear shallow water system (3.1)–(3.2) closely resembles a mixed formulation of a typical scalar elliptic equation. In this case, s_h and r_h in (3.127) reduces to nearly identical constructions presented by Cockburn and Gopalakrishnan (2004) and Cockburn, Gopalakrishnan, and Lazarov (2009). Having established the main result, let us now inspect the discrete problem (3.127) more closely.

3.2.4 The discrete variational problem for λ_h

Now we provide an important result, which helps reveal that the equation for the Lagrange multiplier defines a positive-definite system. This motivates our choice of iterative solvers and preconditioning strategies for (4.47), which will be discussed in Chapter 4. It also motivates directions of future work for studying the properties of linear systems arising from other hybridizable discretizations, such as the ones shown later on in Section 3.4.

Proposition 4. *The bilinear form s_h in (3.128) is positive-definite on $U_h^{\mathrm{tr}} \times U_h^{\mathrm{tr}}$ in the sense that:*

$$s_h(\gamma, \gamma) > 0, \quad \forall \gamma \in U_h^{\mathrm{tr}} \text{ with } \gamma \neq 0. \tag{3.136}$$

When $f = 0$, s_h is symmetric and positive-definite.

Proof. We can immediately see that $s_h(\lambda_h, \gamma) = s_h(\gamma, \lambda_h)$ defines a symmetric bilinear form whenever $f = 0$. Therefore, we only need to verify positive-definiteness for general f.

Observe that, for all $\gamma \in U_h^{\mathrm{tr}}$:

$$\begin{aligned} s_h(\gamma, \gamma) &= \int_{\mathcal{T}_h} \boldsymbol{\mathcal{Q}}^\partial \gamma \cdot \left(\boldsymbol{\mathcal{Q}}^\partial \gamma - \beta f \boldsymbol{\mathcal{Q}}^\partial \gamma^\perp \right) \mathrm{d}\boldsymbol{x} + \frac{g}{H} \int_{\mathcal{T}_h} \mathcal{U}^\partial \gamma \mathcal{U}^\partial \gamma \, \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathcal{T}_h} |\boldsymbol{\mathcal{Q}}^\partial \gamma|^2 \, \mathrm{d}\boldsymbol{x} + \frac{g}{H} \int_{\mathcal{T}_h} |\mathcal{U}^\partial \gamma|^2 \, \mathrm{d}\boldsymbol{x} \\ &= \| \boldsymbol{\mathcal{Q}}^\partial \gamma \|^2_{\hat{U}_h^1} + \frac{g}{H} \| \mathcal{U}^\partial \gamma \|^2_{U_h^2} \geq 0, \end{aligned} \tag{3.137}$$

Note that $g/H > 0$ is strictly positive. This immediately gives a positive semi-definite result. Observe that whenever (3.137) is equal to zero, then this implies $\boldsymbol{\mathcal{Q}}^\partial \gamma = \mathbf{0}$ and $\mathcal{U}^\partial \gamma = 0$. Using the definition of $(\boldsymbol{\mathcal{Q}}^\partial, \mathcal{U}^\partial)$ in (3.59)–(3.60), we have that $\gamma = 0$ by a similar argument made in the proof of Proposition 2. Hence, we have positive-definiteness on $U_h^{\mathrm{tr}} \times U_h^{\mathrm{tr}}$. □

Now let $\mathcal{B}_{\mathrm{tr}} = \{\xi_i\}$ be any finite element basis for U_h^{tr} and $\boldsymbol{\Lambda} = \begin{Bmatrix} \lambda_i, & \cdots & \lambda_r \end{Bmatrix}^T$ be the vector of coefficients for the finite element expansion of $\lambda_h \in U_h^{\mathrm{tr}}$:

$$\lambda_h = \sum_{i=1}^{r} \lambda_i \xi_i. \tag{3.138}$$

Then the corresponding discrete matrix system has the form

$$\boldsymbol{\mathcal{S}} \boldsymbol{\Lambda} = \boldsymbol{\mathcal{R}}, \tag{3.139}$$

with

$$(\boldsymbol{\mathcal{S}})_{ij} = \sum_{K\in\mathcal{T}_h}\left(\int_K \boldsymbol{\mathcal{Q}}^\partial\xi_j\cdot\left(\boldsymbol{\mathcal{Q}}^\partial\xi_i - \beta f\,\boldsymbol{\mathcal{Q}}^\partial\xi_i^\perp\right)\mathrm{d}\boldsymbol{x} + \frac{g}{H}\int_K \mathcal{U}^\partial\xi_j\mathcal{U}^\partial\xi_i\,\mathrm{d}\boldsymbol{x}\right) \tag{3.140}$$

$$\begin{aligned}(\boldsymbol{\mathcal{R}})_j = &\sum_{K\in\mathcal{T}_h}\int_K \boldsymbol{\mathcal{Q}}^\partial\xi_j\cdot\left(2\beta f\left(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h\right)^\perp - \boldsymbol{r}_1^h\right)\mathrm{d}\boldsymbol{x}\\ &+\sum_{K\in\mathcal{T}_h}\frac{g}{H}\int_K \mathcal{U}^\partial\xi_j r_h^2\,\mathrm{d}\boldsymbol{x}.\end{aligned} \tag{3.141}$$

Using Lemma 4, we can rewrite the system more compactly as:

$$(\boldsymbol{\mathcal{S}})_{ij} = -\sum_{K\in\mathcal{T}_h}\int_{\partial K}\xi_j\boldsymbol{\mathcal{Q}}^\partial\xi_i\cdot\boldsymbol{n}\,\mathrm{d}S \tag{3.142}$$

$$(\boldsymbol{\mathcal{R}})_j = \sum_{K\in\mathcal{T}_h}\int_{\partial K}\xi_j\left(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h\right)\cdot\boldsymbol{n}\,\mathrm{d}S. \tag{3.143}$$

Then we can immediately deduce the following properties of (3.139).

1. $\boldsymbol{\mathcal{S}}$ is a positive-definite matrix in $\mathbb{R}^{r\times r}$.
2. If the support of ξ_j (the smallest subset of $\mathcal{E}_h$ where ξ_j is non-zero) does not intersect on ∂K, then
$$(\boldsymbol{\mathcal{R}})_j\,|_K = \int_{\partial K}\xi_j\left(\boldsymbol{\mathcal{Q}}_1^\circ\boldsymbol{r}_1^h + \boldsymbol{\mathcal{Q}}_2^\circ r_2^h\right)\cdot\boldsymbol{n}\,\mathrm{d}S = 0. \tag{3.144}$$
3. If the support of either ξ_i or ξ_j does not intersect ∂K, we have
$$(\boldsymbol{\mathcal{S}})_{ij}\,|_K = \int_{\partial K}\xi_j\boldsymbol{\mathcal{Q}}^\partial\xi_i\cdot\boldsymbol{n}\,\mathrm{d}S = 0. \tag{3.145}$$

Recall that the entire point of using a hybridizable method is that inversion of a globally dense Schur-complement is avoided. The latter two points are merely a confirmation on the sparsity of the matrix system for $\boldsymbol{\Lambda}$. Clearly if the support of $\xi_j \in \mathcal{B}_{\text{tr}}$ in (3.144) or (3.145) doesn't intersect with ∂K, the resulting expressions are zero. Now take ξ_i in (3.145) and suppose its support does not intersect ∂K. Then we have by Proposition 1 that $\left(\boldsymbol{\mathcal{Q}}^\partial\xi_i\right)|_K\cdot\boldsymbol{n}_K = 0$, where $\boldsymbol{n}_K$ is the normal vector on ∂K. Thus (3) follows. Item (1) is a direct consequence of Proposition 4.

Remark on the operator $\boldsymbol{\mathcal{S}}$

Our analysis has led us to the results of Theorem 1. In doing so, we have established the relationship between the Lagrange multiplier λ_h and the variational problem which characterizes it. As we have previously alluded to, the linear system for λ_h is a positive-definite equation. For the hybridization of mixed methods for scalar-elliptic PDEs, Gopalakrishnan (2003) first showed that the system for the Lagrange multipliers is spectrally equivalent to an elliptic operator. This motivated the extension of standard elliptic solvers and preconditioners to the discrete system for the multipliers (Gopalakrishnan and Tan, 2009; Cockburn et al., 2013). In particular: the application of *multigrid methods* as robust preconditioners for inverting the trace system.

The hybridizable method we presented in this section has not be formally analyzed in the literature. However, we can say quite a lot about the system when $f = 0$. In this case, the variational problem in (3.127) reduces to an identical system shown by Cockburn and Gopalakrishnan (2004, Equation (2.13)). Cockburn and Gopalakrishnan (2005) extended the results of Gopalakrishnan (2003) and showed that the "energy" norm $|||\cdot||| := \sqrt{s_h(\cdot,\cdot)}$ is equivalent to an "H^1-like" norm on U_h^{tr} with *mesh independent* constants (see Theorem 3.6 of Cockburn and Gopalakrishnan (2005)). This motivates the use of preconditioning strategies for H^1 elliptic equations directly on the trace problem, as described by Dobrev et al. (2019).

This inspired the choice of solver algorithms featured in later chapters. We shall elaborate later on when the performance of iterative solvers becomes a central topic in Chapters 4 and 5. While there is no formal proof yet on the spectral properties of $\boldsymbol{\mathcal{S}}$ in (3.139), we show strong numerical evidence that the non-symmetric systems arising from the hybridization methods presented in this dissertation respond well to multigrid preconditioning. Extending the analysis of Gopalakrishnan (2003) to (3.127) is a subject for future investigations.

3.3 Nonlinear method and numerical examples

Having established the hybridization method for the linear shallow water model, we now describe how this fits together within a numerical method for the nonlinear shallow water equations. The nonlinear method we shall present here is inspired by the approach used in ENDGame, the operational dynamical core of the UK Met Office (Thuburn, 2016). We briefly summarize our approach.

Let $\boldsymbol{q}^n$ denote the vector of state variables at time-step n that we want to determine. An approximation to $\boldsymbol{q}^n$ is computed by means of a Picard fixed-point iteration of the form:

$$\mathcal{J}(\bar{\boldsymbol{q}})\delta\boldsymbol{q}^k = \mathcal{R}(\boldsymbol{q}^{n,k-1}), \tag{3.146}$$

where and $\mathcal{R}$ is the nonlinear residual, and the linear increment at the k-th Picard iteration $\delta\boldsymbol{q}^k$ is defined via

$$\boldsymbol{q}^{n,k} = \boldsymbol{q}^{n,k-1} + \delta\boldsymbol{q}^k, \tag{3.147}$$

with $\boldsymbol{q}^{n,k}$ denoting the Picard iterates approximating $\boldsymbol{q}^n$. The iterates are initialized by using some initial guess: $\boldsymbol{q}^{n,0} = \boldsymbol{q}^*$. We call this component of the nonlinear method the *Picard cycle*.

In (3.146), $\mathcal{J}(\bar{\boldsymbol{q}})$ denotes the linearization of the nonlinear operator about some reference profile $\bar{\boldsymbol{q}}$. It is more commonly referred to as the approximate *Jacobian* of the nonlinear PDE.[7] Since $\mathcal{J}$ is not a true Jacobian, where $\mathcal{J}$ is constructed from the linearization about the known state $\boldsymbol{q}^{n-1}$, such an approach is typically called "quasi-Newton." For real-world atmospheric simulations, a background profile is used to compute $\mathcal{J}$ in order to save computational costs (Melvin et al., 2010; Wood et al., 2014; Melvin et al., 2019).

More formally, the *stopping criteria* of the Picard cycle is determined by some tolerance $0 < \epsilon \ll 1$, where the cycle terminates whenever the relative change in the solution is small:

$$\|\boldsymbol{q}^{n,k} - \boldsymbol{q}^{n,k-1}\| = \|\delta\boldsymbol{q}^k\| \leq \epsilon\|\boldsymbol{q}^{n,0}\|. \tag{3.148}$$

[7] Not to be confused with the Jacobian of a coordinate transformation!

After termination, at say N_k iterations, the state variable is moved forward in time: $\boldsymbol{q}^n \leftarrow \boldsymbol{q}^{N_k}$ and the next Picard cycle begins. In operational settings, N_k is typically determined heuristically. For example, the Picard method for the UK Met Office's full compressible model uses between 2–4 Picard iterations, which were experimentally determined to be "good enough" for generating accurate forecasts (Wood et al., 2014; Adams et al., 2019).

Since the Picard method approach is currently being used by the experimental (compatible finite element) dynamical core: LFRic (Adams et al., 2019), we choose to solve our nonlinear equations in a similar manner. In Chapter 5, another Picard method is presented for the full compressible Euler equations, which more closely mirrors the one used by (Melvin et al., 2019). First, we start by presenting a compatible finite element discretization of the nonlinear shallow water system.

3.3.1 Nonlinear shallow water equations

The nonlinear shallow water equations in a rotating frame were presented in (2.59)–(2.60). We use the form of the momentum equation given in (2.61):

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\nabla^\perp \cdot \boldsymbol{u} + f)\boldsymbol{u}^\perp + \nabla\left(g(D+\eta_b) + \frac{1}{2}|\boldsymbol{u}|^2\right) = 0, \tag{3.149}$$

$$\frac{\partial D}{\partial t} + \nabla \cdot (\boldsymbol{u}D) = 0. \tag{3.150}$$

Here, we have included a bathymetry term η_b. Our approach extends the compatible finite element discretization of the linear shallow water equations in Section 3.1.1. It is also closely related to the discretization of the incompressible Euler equations described by Natale and Cotter (2017). Like with the linear model, we discretize (3.149)–(3.150) using an $H(\mathrm{div}) \times L^2$ compatible finite element method. That is, $\boldsymbol{u}$ and D are approximated by $\boldsymbol{u}_h \in U_h^1 \subset H(\mathrm{div})$ and $D_h \in U_h^2 \subset L^2$ respectively. Here, we shall start by discretizing in space.

Since D_h is sought in the discontinuous finite element space U_h^2, we use an upwinded discontinuous Galerkin method for the continuity equation (3.150), as detailed in Gibson et al. (2019a, §2.2). Multiplying (3.150) by a test function $\phi \in U_h^2$ and integrating by parts yields:

$$\int_{\mathcal{T}_h} \phi \frac{\partial D_h}{\partial t}\,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \nabla_h\phi \cdot (\boldsymbol{u}_h D_h)\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} [\![\boldsymbol{u}_h\phi]\!]\widetilde{D}_h\,\mathrm{d}\boldsymbol{x} = 0, \tag{3.151}$$

where $[\![\cdot]\!]$ denotes the jump of $\phi\boldsymbol{u}_h \cdot \boldsymbol{n}$[8] on the skeleton $\mathcal{E}_h$ and ∇_h is the "broken" gradient in L^2:

$$\nabla_h|_K\phi = \nabla|_K\phi, \tag{3.152}$$

and $\widetilde{D}_h$ indicates the value of D_h take from the *upwind* direction, i.e., the side where $\boldsymbol{u}_h \cdot \boldsymbol{n}$ is negative.

In the momentum equation (3.149), the main difficulty is in representing the nonlinear term $(\nabla^\perp \cdot \boldsymbol{u}_h)\boldsymbol{u}_h^\perp$ in an $H(\mathrm{div}) \times L^2$ mixed method. There is insufficient continuity in $\boldsymbol{u}_h$ to meaningfully evaluate the (two-dimensional) curl $\nabla^\perp\cdot$ since tangential components of $\boldsymbol{u}_h$ can have discontinuities. Thus, we take the inner product with a

[8] While indeed $[\![\boldsymbol{u}_h]\!] = 0$ since $\boldsymbol{u}_h \in U_h^1$, this does not imply that $[\![\boldsymbol{u}_h\phi]\!] = 0$; ϕ in general can take different values on either side of two connected elements.

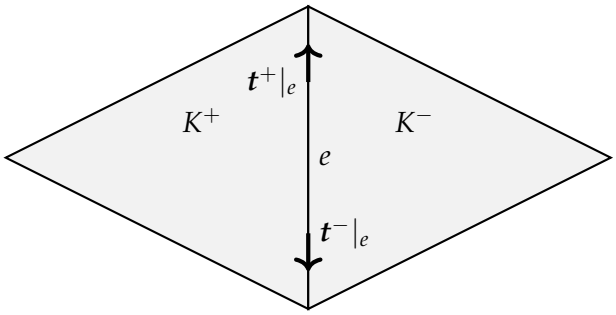


FIGURE 3.6: A shared edge between two adjacent elements K^+ and K^-. The *tangential* normal vectors on the edge e are constructed by rotating (90 degrees) the outward normal vectors on e.

test function $\boldsymbol{w} \in U_h^2$ and integrate by parts, choosing the upwind value of $\boldsymbol{u}_h$ on the cell boundaries:

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot (\nabla^\perp \cdot \boldsymbol{u}_h)\boldsymbol{u}_h^\perp \,\mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}_h} \boldsymbol{u}_h \cdot \nabla_h^\perp(\boldsymbol{u}_h^\perp \cdot \boldsymbol{w}) \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \{\boldsymbol{u}_h^\perp \cdot \boldsymbol{w}\} \cdot \widetilde{\boldsymbol{u}}_h \,\mathrm{d}S. \quad (3.153)$$

In the surface term, $\widetilde{\boldsymbol{u}}_h$ denotes the upwind value of $\boldsymbol{u}_h$ on each facet. The operator $\nabla_h^\perp$ is a (rotated) broken gradient, defined similarly to (3.152). Additionally, we introduce a vector-valued "jump" operator in the tangential directions:

$$\{\Psi\}_e = \Psi|_{K^+}\boldsymbol{t}^+|_e + \Psi|_{K^-}\boldsymbol{t}^-|_e, \quad (3.154)$$

where Ψ is some scalar function and $\boldsymbol{t}^\pm|_e := (\boldsymbol{n}^\pm|_e)^\perp$ is the tangential unit vector on the facet $e \subset \partial K^\pm$ (see Figure 3.6). Note that the surface term in (3.153) appears due to $\boldsymbol{u}_h$ having discontinuous tangential components.

Our semi-discrete finite element formulation reads as follows: find $\boldsymbol{u}_h \in U_h^1$, $D_h \in U_h^2$ such that

$$\begin{aligned}\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \frac{\partial \boldsymbol{u}_h}{\partial t} \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \boldsymbol{u}_h \cdot \nabla_h^\perp(\boldsymbol{u}_h^\perp \cdot \boldsymbol{w}) \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \{\boldsymbol{u}_h^\perp \cdot \boldsymbol{w}\} \cdot \widetilde{\boldsymbol{u}}_h \,\mathrm{d}S \\ + \int_{\mathcal{T}_h} \boldsymbol{w} \cdot (f\boldsymbol{u}_h^\perp) \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w} \left(g(D_h + \eta_b) + \frac{1}{2}|\boldsymbol{u}_h|^2\right) \,\mathrm{d}\boldsymbol{x} = 0, \quad (3.155)\end{aligned}$$

$$\int_{\mathcal{T}_h} \phi \frac{\partial D_h}{\partial t} \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \nabla_h \phi \cdot (\boldsymbol{u}_h D_h) \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} [\![\boldsymbol{u}_h \phi]\!] \widetilde{D}_h \,\mathrm{d}\boldsymbol{x} = 0, \quad (3.156)$$

for all test functions $\boldsymbol{w} \in U_h^1$, $\phi \in U_h^2$. This scheme has no particular conservation properties beyond local and global mass conservation.

For the interested reader, an energy-conserving formulation can be obtained by introducing a mass-flux: $\boldsymbol{F}_h \approx \boldsymbol{u}D$, defined as the L^2-projection of $\boldsymbol{u}D$ onto U_h^1. One then replaces $\boldsymbol{u}_h$ by $\boldsymbol{F}_h/D_h$ in (3.155), and uses the continuity equation $\frac{\partial D_h}{\partial t} + \nabla_h \cdot \boldsymbol{F}_h = 0$. A scheme that conserves mass, energy, and potential enstrophy was given in McRae and Cotter (2014); this introduced extra variables representing mass-flux and potential vorticity. Other sophisticated energy-conserving discretizations can be found in Shipton, Gibson, and Cotter (2018) and Bauer and Cotter (2018).

Quasi-Newton/Picard iteration scheme

The nonlinear method described in this section is as presented by Gibson et al. (2019a) and Gibson et al. (2019b). This method mirrors the solution strategy employed by the UK Met Office's new finite element dynamical core, as presented by Melvin et al. (2019) and Adams et al. (2019). We start by discretizing the nonlinear equations in time and employ a Picard method.

We discretize (3.155) and (3.156) using the implicit midpoint rule. This implies that the nonlinear residuals $R_{\boldsymbol{u}}[\boldsymbol{w}; \boldsymbol{u}_h^n, D_h^n]$ and $R_D[\phi; \boldsymbol{u}_h^n, D_h^n]$ should satisfy

$$R_{\boldsymbol{u}}[\boldsymbol{w}; \boldsymbol{u}_h^n, D_h^n] = 0, \tag{3.157}$$
$$R_D[\phi; \boldsymbol{u}_h^n, D_h^n] = 0, \tag{3.158}$$

for all $\boldsymbol{w} \in U_h^1$ and $\phi \in U_h^2$, where

$$\begin{aligned} R_{\boldsymbol{u}}[\boldsymbol{w}; \boldsymbol{u}_h^n, D_h^n] = & \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \frac{\boldsymbol{u}_h^n - \boldsymbol{u}_h^{n-1}}{\Delta t} \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \boldsymbol{w} \cdot f \left(\boldsymbol{u}_h^{n-\frac{1}{2}}\right)^{\perp} \,\mathrm{d}\boldsymbol{x} \\ & - \int_{\mathcal{T}_h} \boldsymbol{u}_h^{n-\frac{1}{2}} \cdot \nabla_h^{\perp} \left(\left(\boldsymbol{u}_h^{n-\frac{1}{2}}\right)^{\perp} \cdot \boldsymbol{w} \right) \,\mathrm{d}\boldsymbol{x} \\ & + \int_{\mathcal{E}_h} \left\{ \left(\boldsymbol{u}_h^{n-\frac{1}{2}}\right)^{\perp} \cdot \boldsymbol{w} \right\} \cdot \widetilde{\boldsymbol{u}_h^{n-\frac{1}{2}}} \,\mathrm{d}S \\ & - \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w} \left(g \left(D_h^{n-\frac{1}{2}} + \eta_b \right) + \frac{1}{2} \left| \boldsymbol{u}_h^{n-\frac{1}{2}} \right|^2 \right) \,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.159}$$

and

$$\begin{aligned} R_D[\phi; \boldsymbol{u}_h^n, D_h^n] = & \int_{\mathcal{T}_h} \phi \, \frac{D_h^n - D_h^{n-1}}{\Delta t} \,\mathrm{d}\boldsymbol{x} \\ & - \int_{\mathcal{T}_h} \nabla_h \phi \cdot \boldsymbol{u}_h^{n-\frac{1}{2}} D_h^{n-\frac{1}{2}} + \int_{\mathcal{E}_h} \left[\!\left[\phi \boldsymbol{u}_h^{n-\frac{1}{2}} \right]\!\right] \widetilde{D_h^{n-\frac{1}{2}}} \,\mathrm{d}S, \end{aligned} \tag{3.160}$$

with

$$\boldsymbol{u}_h^{n-\frac{1}{2}} = \frac{1}{2}(\boldsymbol{u}_h^n + \boldsymbol{u}_h^{n-1}), \quad D_h^{n-\frac{1}{2}} = \frac{1}{2}(D_h^n + D_h^{n-1}) \tag{3.161}$$

denoting the midpoint fields. This time discretization is unconditionally stable in the sense that solutions do not grow arbitrarily large in time. More specifically, the implicit midpoint scheme is an implicit Runga-Kutta (RK) method, which is well-known for ensuring that solutions are well-controlled in time (Hairer and Wanner, 1996, §4.3).

However, solving (3.157)–(3.158) requires the solution of a nonlinear coupled system of equations for $\boldsymbol{u}_h^n$ and D_h^n. One could use Newton's method directly, but this has a disadvantage: the Jacobian must be reassembled each nonlinear iteration. If a Krylov method is used to solve the linear system, the preconditioner must be rebuilt; if a direct method is used instead, the factorization must be recalculated during each iteration. Therefore, an approximate Jacobian is used in practice, which is obtained by linearizing around a fixed background profile. The resulting Jacobian is then state-independent and allows reuse of a sparse preconditioner.

To control the reduction of the nonlinear residual, we apply a fixed number of Picard iterations, say N_k iterations, towards the solution of the implicit midpoint system. As long as this approximate Jacobian is not too far from the true Jacobian, the nonlinear residuals $R_{\boldsymbol{u}}$ and R_D can still be made arbitrarily small. In practice, this works well since flows on a global scale do not deviate far from the background profile (average wind speeds, pressure, etc.) of the atmosphere. The Picard cycle for (3.157)–(3.158) is summarized as follows.

As we have mentioned at the start of this section, the Picard method generates a sequence of approximations $\boldsymbol{u}_h^{n,0}, \boldsymbol{u}_h^{n,1}, \boldsymbol{u}_h^{n,2}, \ldots, \boldsymbol{u}_h^{n,N_k}$, and $D_h^{n,0}, D_h^{n,1}, D_h^{n,2}, \ldots, D_h^{n,N_k}$ to $\boldsymbol{u}_h^n$ and D_h^n respectively. To start the Picard iteration, a simple approach would be to initialize the Picard iterates using values from the previous time-step: $\boldsymbol{u}_h^{n,0} = \boldsymbol{u}_h^{n-1}$ and $D_h^{n,0} = D_h^{n-1}$. Then, we define the linear increments at the k-th Picard iteration, $\delta\boldsymbol{u}_h^k$ and δD_h^k, via

$$\boldsymbol{u}_h^{n,k} = \boldsymbol{u}_h^{n,k-1} + \delta\boldsymbol{u}_h^k, \quad D_h^{n,k} = D_h^{n,k-1} + \delta D_h^k. \tag{3.162}$$

These increments are chosen to satisfy the approximate Jacobian system of the form:

$$\int_{\mathcal{T}_h} \boldsymbol{w}\cdot\delta\boldsymbol{u}_h^k \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2}\int_{\mathcal{T}_h} \boldsymbol{w}\cdot f\left(\delta\boldsymbol{u}_h^k\right)^\perp \,\mathrm{d}\boldsymbol{x} - \frac{g\Delta t}{2}\int_{\mathcal{T}_h} \delta D_h^k \nabla_h\cdot\boldsymbol{w}\,\mathrm{d}\boldsymbol{x} = -\Delta t R_{\boldsymbol{u}}[\boldsymbol{w}; \boldsymbol{u}_h^{n,k-1}, D_h^{n,k-1}], \tag{3.163}$$

$$\frac{H\Delta t}{2}\int_{\mathcal{T}_h} \phi\nabla_h\cdot\delta\boldsymbol{u}_h^k \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \phi\delta D_h^k \,\mathrm{d}\boldsymbol{x} = -\Delta t R_D[\phi; \boldsymbol{u}_h^{n,k-1}, D_h^{n,k-1}], \tag{3.164}$$

for all $\boldsymbol{w} \in U_h^1$ and $\phi \in U_h^2$, where the residual functionals $R_{\boldsymbol{u}}$ and R_h are the nonlinear residuals. The Jacobian system above is obtained by linearizing the shallow water equations about a state of rest with a mean layer depth H and no bathymetry, as discussed in Section 2.1.4.

The choice to take $\boldsymbol{u}_h^{n,0} = \boldsymbol{u}_h^{n-1}$ and $D_h^{n,0} = D_h^{n-1}$ is a fairly simple approach, but there is a caveat. While the actual time-integrator is unconditionally stable for large Δt, this doesn't necessarily mean the the Picard method will converge. The Picard cycle defined by successive inversion of the system in (3.163)–(3.164) will converge or diverge depending on whether $\delta\boldsymbol{u}_h^k$ and δD_h^k can be made sufficiently small. For large Δt, the Picard method may converge very slowly (requiring several Picard iterations) or not at all.

To accelerate the convergence of the Picard method (hence extend the stable time-step of the nonlinear method), we make one further modification. We now replace $R_{\boldsymbol{u}}$ and R_D by equivalent functionals (i.e. functionals that also vanish when the implicit midpoint rule equations are satisfied). To do this, we set

$$\overline{\boldsymbol{u}_h^{k-1}} = \frac{1}{2}\left(\boldsymbol{u}_h^{n,k-1} + \boldsymbol{u}_h^{n-1}\right), \quad \overline{D_h^{k-1}} = \frac{1}{2}\left(D_h^{n,k-1} + D_h^{n-1}\right) \tag{3.165}$$

at the k-th Picard iteration, with $\boldsymbol{u}_h^{n,0} = \boldsymbol{u}_h^{n-1}$ and $D_h^{n,0} = D_h^{n-1}$ as before. Now we introduce the *candidate* solutions: $\widehat{\boldsymbol{u}}_h^k$, $\widehat{D}_h^k$, and the implicit midpoint quantities:

$$\widehat{\boldsymbol{u}}_h^{k-\frac{1}{2}} = \frac{1}{2}\left(\widehat{\boldsymbol{u}}_h^k + \boldsymbol{u}_h^{n-1}\right), \quad \widehat{D}_h^{k-\frac{1}{2}} = \frac{1}{2}\left(\widehat{D}_h^k + D_h^{n-1}\right). \tag{3.166}$$

Now we solve the implicit equations (3.159)–(3.160) using $\overline{\boldsymbol{u}_h^{k-1}}$ as the advecting velocity. That is, we solve for the candidates $\widehat{\boldsymbol{u}}_h^k \in U_h^1$ and $\widehat{D}_h^k \in U_h^2$ satisfying the *linear* equations:

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \frac{\widehat{\boldsymbol{u}}_h^k - \boldsymbol{u}_h^{n-1}}{\Delta t}\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \boldsymbol{w} \cdot f\left(\widehat{\boldsymbol{u}}_h^{k-\frac{1}{2}}\right)^{\perp}\,\mathrm{d}\boldsymbol{x} \\
&\qquad - \int_{\mathcal{T}_h} \widehat{\boldsymbol{u}}_h^{k-\frac{1}{2}} \cdot \nabla_h^{\perp}\left(\left(\overline{\boldsymbol{u}_h^{k-1}}\right)^{\perp} \cdot \boldsymbol{w}\right)\,\mathrm{d}\boldsymbol{x} \\
&\qquad + \int_{\mathcal{E}_h} \left\{\left(\overline{\boldsymbol{u}_h^{k-1}}\right)^{\perp} \cdot \boldsymbol{w}\right\} \cdot \widetilde{\widehat{\boldsymbol{u}}_h^{k-\frac{1}{2}}}\,\mathrm{d}S \\
&+ \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w}\left(g\left(\overline{D_h^{k-1}} + \eta_b\right) + \frac{1}{2}\left|\overline{\boldsymbol{u}_h^{k-1}}\right|^2\right)\,\mathrm{d}\boldsymbol{x} = 0, \quad \forall \boldsymbol{w} \in U_h^1,
\end{aligned} \tag{3.167}$$

and

$$\begin{aligned}
&\int_{\mathcal{T}_h} \phi \frac{\widehat{D}_h^k - D_h^{n-1}}{\Delta t}\,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \nabla_h \phi \cdot \left(\overline{\boldsymbol{u}_h^{k-1}}\widehat{D}_h^{k-\frac{1}{2}}\right)\,\mathrm{d}\boldsymbol{x} \\
&\qquad + \int_{\mathcal{E}_h} \left[\!\left[\phi \overline{\boldsymbol{u}_h^{k-1}}\right]\!\right] \widetilde{\widehat{D}_h^{k-\frac{1}{2}}}\,\mathrm{d}S = 0, \quad \forall \phi \in U_h^2.
\end{aligned} \tag{3.168}$$

The equations (3.167) and (3.168) are linear in the arguments $\widehat{\boldsymbol{u}}_h^k$, $\widehat{D}_h^k$. Furthermore, they can be solved separately since there is no coupling between them.

This procedure can be thought of as iteratively solving for the average velocity and depth that satisfies the implicit midpoint rule discretization. Once (3.167) and (3.168) are solved, the candidate fields $\widehat{\boldsymbol{u}}_h^k$, $\widehat{D}_h^k$ are then used to define the new residuals for the Picard system:

$$\widehat{R}_{\boldsymbol{u}}[\boldsymbol{w}; \boldsymbol{u}_h^{n,k-1}, D_h^{n,k-1}] = \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \left(\boldsymbol{u}_h^{n,k-1} - \widehat{\boldsymbol{u}}_h^k\right)\,\mathrm{d}\boldsymbol{x}, \tag{3.169}$$

$$\widehat{R}_h[\phi; \boldsymbol{u}_h^{n,k-1}, D_h^{n,k-1}] = \int_{\mathcal{T}_h} \phi\left(D_h^{n,k-1} - \widehat{D}_h^k\right)\,\mathrm{d}\boldsymbol{x}, \tag{3.170}$$

which replace $R_{\boldsymbol{u}}$ and R_h in (3.163) and (3.164). Algorithm 1 summarizes the entire nonlinear method for solving (3.157)–(3.158).

Remark 8. *At the time of this work, no formal analysis has been done for the Picard methods employed throughout this dissertation (including the ones used by Wood et al. (2014), Melvin et al. (2019), and Adams et al. (2019)). However, such methods used in operational settings work quite well and give sufficiently accurate forecasts. It would be interesting future work to analyze these methods in more detail and give concrete convergence results.*

3.3.2 Numerical experiments on the sphere

Having established our nonlinear method, we see that we must invert the linear system (3.163)–(3.164) a total of N_k-times per time-step. The linear system has the exact form of the linear equations presented for the hybridizable method in Sections 3.1 and 3.2. As a proof-of-concept, we will employ Algorithm 1 using a hybridizable formulation of (3.163)–(3.164), exactly as detailed in Section 3.1.2. For all experiments, we use a fixed number of $N_k = 4$ Picard iterations. This was chosen heuristically, as more Picard iterations did not appear to drastically change the solutions. A similar

Algorithm 1 Pseudocode for solving the nonlinear shallow water equations using the implicit midpoint rule and Picard method.

$t_n = 0$
$\boldsymbol{u}_h^n \leftarrow \boldsymbol{u}(\boldsymbol{x}, t = 0)$ ▷ Initial condition for the velocity field
$D_h^n \leftarrow D(\boldsymbol{x}, t = 0)$ ▷ Initial condition for the depth field
while $t < t_{\max}$ **do**
  $\boldsymbol{u}_h^{n,k} = \boldsymbol{u}_h^n$ ▷ Initialize the Picard method
  $D_h^{n,k} = D_h^n$
  for $k \in \{1, \cdots, N_k\}$ **do** ▷ Start Picard cycle
    Solve (3.168) and (3.167) for the candidates $\widehat{D}_h^k$ and $\widehat{\boldsymbol{u}}_h^k$.
    Evaluate new residuals $\widehat{R}_{\boldsymbol{u}}^k$ and $\widehat{R}_D^k$:

$$\widehat{R}_{\boldsymbol{u}}^k = \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \left(\boldsymbol{u}_h^{n,k} - \widehat{\boldsymbol{u}}_h^k \right) \mathrm{d}\boldsymbol{x}$$
$$\widehat{R}_D^k = \int_{\mathcal{T}_h} \phi \left(D_h^{n,k} - \widehat{D}_h^k \right) \mathrm{d}\boldsymbol{x}$$

    Solve the linear system for $\delta \boldsymbol{u}_h^k$ and δD_h^k:

$$\begin{bmatrix} \widetilde{\boldsymbol{M}_1} & -\frac{g\Delta t}{2}\boldsymbol{B}^T \\ \frac{H\Delta t}{2}\boldsymbol{B} & \boldsymbol{M}_2 \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{u}_h^k \\ \delta D_h^k \end{Bmatrix} = -\Delta t \begin{Bmatrix} \widehat{R}_{\boldsymbol{u}}^k \\ \widehat{R}_h^k \end{Bmatrix}$$

    $\boldsymbol{u}_h^{n,k} \leftarrow \boldsymbol{u}_h^{n,k} + \delta \boldsymbol{u}_h^k$ ▷ Linear update for velocity
    $D_h^{n,k} \leftarrow D_h^{n,k} + \delta D_h^k$ ▷ Linear update for depth
  end for
  $t_n \leftarrow t_n + \Delta t$
  $\boldsymbol{u}_h^n \leftarrow \boldsymbol{u}_h^{n,k}$ ▷ Update velocity for next time step
  $D_h^n \leftarrow D_h^{n,k}$ ▷ Update depth for next time step
end while

phenomena was observed by Wimmer, Cotter, and Bauer (2019), where they compared $N_k = 4$ versus $N_k = 8$. No further analysis was performed on how increasing N_k impacts the overall convergence of the nonlinear method.

Note that mixed finite element methods for the nonlinear equations has already been studied in great detail (Cotter and Shipton, 2012; McRae and Cotter, 2014; Shipton, Gibson, and Cotter, 2018). Unlike the previous cases, we employ hybridization as a new solution technique within our nonlinear method. We demonstrate, using standard test cases on the sphere, that using a hybridizable method produces the expected numerical results when compared with existing studies of the same experiments.

We consider two well-known test cases in this section, taken directly from the shallow water test-suite by Williamson et al. (1992). All test cases are implemented using the Firedrake finite element library (Rathgeber et al., 2017). Local static condensation and recovery for the hybridization method uses the framework developed in Chapter 4 of this dissertation. We shall go into more detail on the computational aspects, performance, and iterative solver strategies related to hybridization later on. For the purpose of these examples, we simply use direct methods (LU via the MUMPS package (Amestoy, Duff, and L'Excellent, 2000)) for all linear solves.

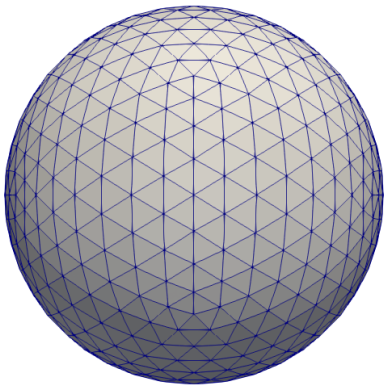


FIGURE 3.7: Grid consisting of three regular refinements of an icosahedra, viewed looking down on the North pole. The tessellation shown here is constructed from flat (linear degree) triangular cells.

The computational domain

For both examples considered here, the governing equations (3.149)–(3.150) will be defined on the two-dimensional surface of a sphere. The sphere has uniform radius $R = 6.3712 \times 10^6$m, with a surface $\mathcal{M}$ embedded in $\mathbb{R}^3$. We previously discussed problems of this type in Section 2.3.2, where we outlined the necessary machinery to evaluate finite element integrals on immersed manifolds. All computations on the sphere performed in Firedrake use the framework developed by Rognes et al. (2013). When convenient, we will use the Firedrake-convention of formulating problems in Cartesian (x, y, z) coordinates with the origin $(0, 0, 0)$ at the center of the sphere.

For our computational domain, we use a mesh $\mathcal{T}_h = \{K\}$ consisting of refined icosahedra (triangles) using a piece-wise cubic polynomial approximation of the surface (see Figure 3.7 for a piece-wise linear example), which places us in the framework of Holst and Stern (2012) for mixed methods on immersed manifolds. In other words, we are using isoparametric cells K to approximate the curvature of the sphere. There are two direct implications of this, as discussed in Section 2.3.2.

1. Our computational domain $\bigcup_i K_i$ is an approximation of $\mathcal{M}$, i.e, $\bigcup_i K_i \approx \mathcal{M}$ and $\bigcup_i K_i \not\subseteq \mathcal{M}$.[9]
2. The coordinate mapping to the reference cell $\hat{K}$ is, in general, no longer affine.[10]

As a consequence of 2, the cell Jacobian of the coordinate mapping $F_K : \mathbb{R}^2 \to \mathbb{R}^3$, $J_K = \frac{\partial F_K}{\partial \hat{\boldsymbol{x}}}$, is non-square and its determinant is polynomial in the reference coordinate $\hat{\boldsymbol{x}} \in \mathbb{R}^2$. This may lead to non-polynomial expressions in the integrands after mapping to $\hat{K}$ (see (2.189) for a concrete example).

Fortunately, by virtue of Lemma 1, all terms which are critical for the conservation properties we desire (Coriolis, divergence and pressure gradient terms) can still be integrated exactly with relative ease (including the surface terms arising from hybridization). This is due to the cancellation of cell Jacobian determinants in the integrals when mapping to the reference cell $\hat{K}$ (Cotter and Thuburn, 2014). For the numerical experiments presented in this section, all critical terms in (3.163)–(3.164) are integrated using exact quadrature.

[9] See Remark 4 in Section 2.3.2 and its mathematical implications in Section 2.4.6.
[10] When using flat triangular elements, the cell Jacobian is still constant.

Solid body rotation

The first example we consider is known as a "solid body" rotation test (Williamson et al., 1992, Test case 2). This test case consists of a steady state solution to the nonlinear equations on the sphere. The zonal (eastward) flow is initialize in a state of geostrophic balance. That is, the flow in its initial state satisfies the balance relation:

$$f\boldsymbol{u}^{\perp} = -g\nabla D, \tag{3.171}$$

where $g = 9.81\text{ms}^{-2}$ is the acceleration due to gravity, $\boldsymbol{u}$ is the flow velocity, and D is the fluid depth, and f is the Coriolis parameter given as a function of the vertical coordinate z:

$$f = 2\Omega_r \frac{z}{R}, \tag{3.172}$$

where $\Omega_r = 7.292 \times 10^{-5}\text{s}^{-1}$ is the angular rotation rate of the Earth, and R is the planetary radius. Note the the definition of f in (3.172) is nothing more than the f-plane approximation in Cartesian coordinates; while f varies with z on the surface, it is *constant* in the zonal directions and achieves its maximal amplitude at the north/-south poles. Along the equator ($z = 0$), f vanishes. There are no topographically-driven ($\eta_b = 0$) or other external forces acting on the fluid. Therefore, as the flow evolves in time, it should remain in geostrophic balance.

For the spatial discretization, we use the following finite element complex in two-dimensions:

$$\text{P}_3 \xrightarrow{\nabla^{\perp}} \text{BDM}_2 \xrightarrow{\nabla\cdot} \text{dP}_1. \tag{3.173}$$

That is, our computed velocity will be constructed in BDM_2 and the depth in dP_1. This implies that the hybridizable method for the linear updates in (3.163)–(3.164) will be an H-BDM method. We use the "broken" version of BDM_2 to construct the velocity updates. Since, for any $\boldsymbol{\psi} \in \text{BDM}_2$

$$\boldsymbol{\psi}\cdot\boldsymbol{n}|_e \in \mathcal{P}_2(e), \quad \forall e \in \mathcal{E}_h, \tag{3.174}$$

a trace space U_h^{tr} consisting of quadratic polynomial functions on each facet is used for the Lagrange multipliers.

The initial conditions for the solid body rotation test are the fields satisfying (3.171), defined as the functions:

$$\boldsymbol{u}(\boldsymbol{x},0) = \frac{u_0}{R}\begin{Bmatrix}-y & x & 0\end{Bmatrix}^T, \quad D(\boldsymbol{x},0) = H_0 - \left(R\Omega_r u_0 + \frac{1}{2}u_0^2\right)\frac{z^2}{gR^2}, \tag{3.175}$$

where $u_0 = \frac{2\pi R}{12\text{days}} \approx 38.6\text{ms}^{-1}$, and $H_0 = 2.94 \times 10^4\text{m}^2\text{s}^{-2}/g \approx 2998\text{m}$. Since the flow is steady, we are able to compare our computed solutions with the analytic result given in (3.175).

The icosahedral grids used in our convergence tests consist of regular refinements of a coarse icosahedral mesh, depicted previously in Figure 3.7. Grid details, including degree of freedom count and time-step information is presented in Table 3.2. The time-step was chosen such that the Courant number $u_0\Delta t/\Delta x$ is approximately 0.2 across all mesh resolutions. The Picard system (3.163)–(3.164) was obtained by linearizing around a state of rest, with a domain-averaged depth given by $H = \int_{\mathcal{T}_h} D(\boldsymbol{x},0)\,\text{d}\boldsymbol{x}/\text{Vol}(\mathcal{T}_h)$, where $\text{Vol}(\mathcal{T}_h)$ is the mesh volume.

TABLE 3.2: Grid properties for the four grids used in the solid body convergence test, including the number of degrees of freedom for the fluid velocity $\boldsymbol{u}_h$ and depth D_h, along with the time-step used.

Grid properties			Degrees of freedom		
cells	$\Delta x_{\max}$ (km)	$\Delta x_{\min}$ (km)	$\boldsymbol{u}_h$	D_h	Δt (s)
1280	1054	720	9600	3840	3000
5120	527	348	38400	15360	1500
20480	263	171	153600	61440	750
81920	132	85	614400	245760	375

The discretization parameters shown in Table 3.2 are the same ones used in the solid body test by Shipton, Gibson, and Cotter (2018). As was done by Williamson et al. (1992), we compute normalized L^2 errors for the prognostic variables at day 5:

$$L^2_{\text{err}}(\Psi) = \frac{\sqrt{\int_{\mathcal{T}_h} |\Psi_h - \Psi_a|^2 \, \mathrm{d}\boldsymbol{x}}}{\sqrt{\int_{\mathcal{T}_h} |\Psi_a|^2 \, \mathrm{d}\boldsymbol{x}}}, \tag{3.176}$$

where Ψ_a is the analytic result, and Ψ_h is the computed solution at day 5. Since the implicit midpoint method is a second-order time-integrator, we expect the solutions to convergence in time proportional to Δx^2. The estimated rates of convergence are computed in the traditional way using the standard L^2-norm:

$$\text{Rate}(\Psi) = \frac{\log\left(\frac{\|\Psi_{h_1} - \Psi_a\|_{L^2}}{\|\Psi_{h_2} - \Psi_a\|_{L^2}}\right)}{\log\left(\frac{\Delta x_{h_1}}{\Delta x_{h_2}}\right)}, \tag{3.177}$$

where h_1 and h_2 are two different (fine) mesh spacings. The computed errors and convergence rates are displayed in Table 3.3 and visualized in Figure 3.8.

We remark here that this test case was also discussed in Shipton, Gibson, and Cotter (2018), but with a different discretization approach for the nonlinear equations. In that study, a Taylor-Galerkin (Donea, 1984) method is used to discretize the momentum equation expressed as a transport equation for the potential vorticity (see (2.67)). A similar upwind discontinuous Galerkin method is used for the mass transport equation (3.151). The main advantage of using the Taylor-Galerkin approach is that conservation of potential vorticity is achieved for both steady and unsteady flow regimes. While the discretization is different from the one presented in this section, the overall nonlinear method follows the same approach as discussed in Section 3.3.1. In particular, the Jacobian for the linear updates is the *same* one used in Algorithm 1. The *same* hybridizable mixed method is applied and produces similar results as here.

TABLE 3.3: Normalized L^2 depth and velocity errors, and the estimated rates of convergence are shown at day 5 for the solid body rotation test.

cells	$L^2_{\text{err}}(\boldsymbol{u})$	$L^2_{\text{err}}(D)$	Rate ($\boldsymbol{u}$)	Rate (D)
1280	3.027×10^{-3}	2.610×10^{-4}	—	—
5120	2.914×10^{-4}	6.188×10^{-5}	3.376	2.077
20480	6.398×10^{-5}	1.623×10^{-5}	2.187	1.930
81920	1.716×10^{-5}	4.050×10^{-6}	1.898	2.003

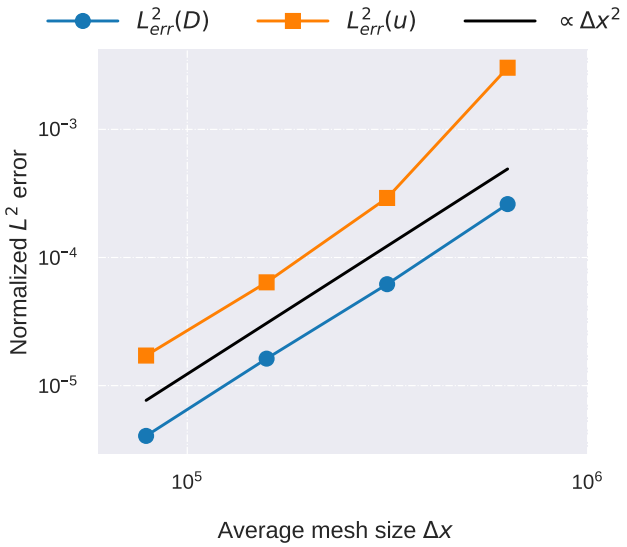


FIGURE 3.8: A log-log plot of normalized L^2 errors versus the average cell resolution Δx. As the resolution is refined, we approach second-order convergence as expected.

Isolated mountain test case

The last example for the shallow water equations we present is based on test case 5 of Williamson et al. (1992). The flow is initialized as in the solid body rotation test (3.175), but with $H_0 = 5960\text{m}$ and $u_0 = 20\text{ms}^{-1}$. However, unlike the solid body test, the flow is now impinging against an isolated conical mountain located in the northern hemisphere. This will cause the flow to become unsteady and develop fine-scale turbulent features as the simulation progresses. There is no analytic solution for this test case, but it is well-documented within the geophysical community. For example, see the more recent studies conducted on this test case by Thuburn, Cotter, and Dubos (2014), Cotter and McRae (2014), and Kang, Giraldo, and Bui-Thanh (2019).

The isolated mountain is added as a topographic feature to the equations via the bathymetry term η_b. Notwithstanding a momentary break in convention, it is more convenient to express the mountain profile in *geographic* coordinates, with (ϕ, λ) denoting latitude and longitude respectively. They are related to the Cartesian coordinates by the relations:

$$\phi = \arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right), \tag{3.178}$$

$$\lambda = \arctan\left(\frac{y}{x}\right), \tag{3.179}$$

where arctan must be suitably defined, taking into account the correct quadrant (x, y) are located in.[11] For the degenerate case when $x = 0$, $\lambda = -\frac{\pi}{2}$ if $y < 0$, $\lambda = \frac{\pi}{2}$ if $y > 0$, and is undefined otherwise. The mountain profile is then defined as the function:

$$\eta_b = D_{\text{peak}}\left(1 - \frac{\mathcal{R}}{R_0}\right), \tag{3.180}$$

[11] See the standard Fortran convention: https://gcc.gnu.org/onlinedocs/gfortran/ATAN2.html

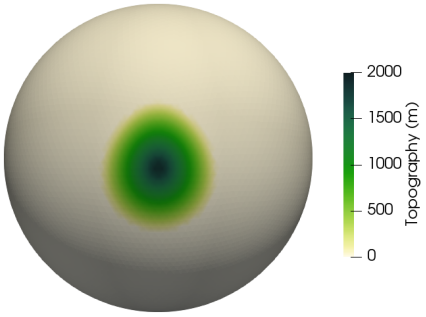


FIGURE 3.9: The topography field in meters, with a peak centered at latitude $\phi_c = \frac{\pi}{6}$ and longitude $\lambda_c = -\frac{\pi}{2}$.

with a peak height $D_{\text{peak}} = 2000\text{m}$, $R_0 = \frac{\pi}{9}$, and $\mathcal{R}^2 = \min\{R_0^2, (\phi - \phi_c)^2 + (\lambda - \lambda_c)^2\}$ with its center in the upper hemisphere at latitude $\phi_c = \frac{\pi}{6}$ and longitude $\lambda_c = -\frac{\pi}{2}$. See Figure 3.9 for a visualization of the topography field.

As before, we discretize using the mixed finite element pair $\text{BDM}_2 \times \text{dP}_1$ for the velocity/depth, and employ an H-BDM discretization of the linear system for the Picard updates. However, this time we run the test at a fixed-resolution of $\Delta x \approx 210$ km (20,480 triangular cells) and a prescribed time-step size of $\Delta t = 450$ seconds. The model was run for a total of 50 simulation days in order to allow the flow to develop strong nonlinearities.

As the flow interacts with the mountain, waves are produced which travel around the globe. Figure 3.10 displays the depth field for days 0, 5, 10, and 15. During this portion of the simulation, the flow is only weakly nonlinear; the deformation of the depth field is apparent, but it remains smooth. The results for days 5, 10, and 15 are comparable to the corresponding results of Nair, Thomas, and Loft (2005), Ullrich, Jablonowski, and Leer (2010), and Kang, Giraldo, and Bui-Thanh (2019).

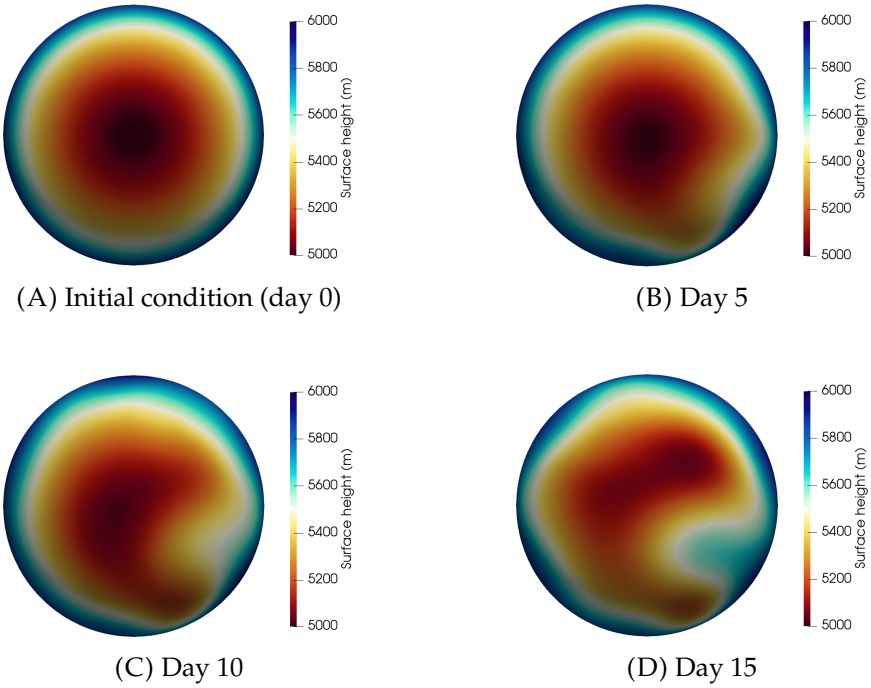
While a convergence study using the approach outlined in Algorithm 1 was not conducted here, a convergence study for this test case was presented by Shipton, Gibson, and Cotter (2018, Figure 4). Errors in the depth field (at day 15) were compute using a high-resolution finite volume code provided by Prof. John Thuburn at the University of Exeter. In that study, we demonstrated second-order convergence using the same time-integrator described in (3.3.1) for the energy-conserving method detailed in Shipton, Gibson, and Cotter (2018, §2.3). The hybridization method presented in Section 3.1.2 was also used in that convergence study.

As was performed by Thuburn, Cotter, and Dubos (2014), we continue the simulation up to day 50 and measure the potential vorticity q. Recall from Section 2.1.4 the definition:

$$q = \frac{\nabla^\perp \cdot \boldsymbol{u} + f}{D}. \tag{3.181}$$

As we can see, q is proportional to the curl of the velocity field. The absolute vorticity $\nabla^\perp \cdot \boldsymbol{u} + f$ denotes the tendency for the flow to locally circulate due to rotational effects, and hence q is just the absolute vorticity normalized by the fluid depth.

In our setup, q is diagnostic; that is, we can determine q using computed solutions for $\boldsymbol{u}$ and D. To accomplish this within the compatible finite element framework, recall the de-Rham complex in (3.173). Since q contains the curl of $\boldsymbol{u}$, we require (in two-dimensions) the H^1 finite element subspace: P_3. To arrive at a diagnostic



(A) Initial condition (day 0)

(B) Day 5

(C) Day 10

(D) Day 15

FIGURE 3.10: Snapshots (view from the northern pole) from the isolated mountain test case. The depth field (m) at days 5, 10, and 15. The snapshots were generated on a mesh with 20,480 simplicial cells, a $\mathrm{BDM}_2 \times \mathrm{dP}_1$ discretization, and $\Delta t = 450$ seconds.

equation for q at time-step n, we rewrite (3.181) and discretize in the usual finite element manner.

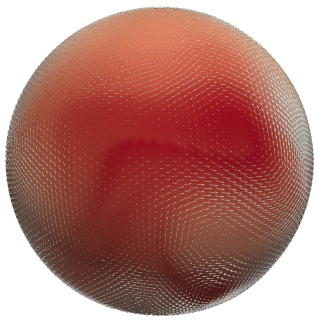
After discretizing (3.181), we obtain the following finite element problem: find $q_h^n \in \mathrm{P}_3$ such that

$$\int_{\mathcal{T}_h} q_h^n D_h^n \psi \, \mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}_h} \nabla_h^\perp \psi \cdot \boldsymbol{u}_h^n \, \mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \psi f \, \mathrm{d}\boldsymbol{x}, \quad \forall \psi \in \mathrm{P}_3, \tag{3.182}$$

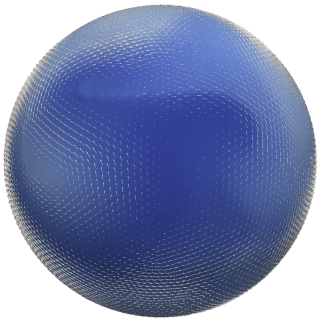
where $\boldsymbol{u}_h^n$ and D_h^n are known fields computed during each nonlinear iteration. No surface terms appear due to the continuity of ψ and the fact the we are discretizing on the surface of the sphere (hence no boundary). Since $\psi \in \mathrm{P}_3$, evaluating the skew-gradient $\nabla^\perp \psi$ in an integral-sense is well-defined.

After solving for $\boldsymbol{u}_h^n$ and D_h^n during each time-step, we diagnose q_h^n by solving the finite element problem (3.182). This is accomplished by a direct factorization (LU) on the resulting matrix system for q_h^n. The resulting potential vorticity field at day 20, 30, 40 and 50 are shown in Figure 3.11.

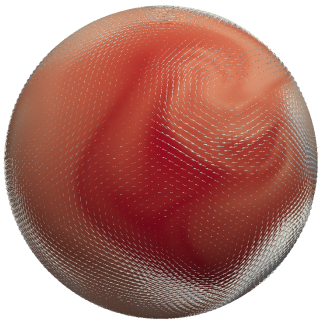
After day 30, the flow becomes more nonlinear due to interactions with the mountain. By day 50, fine scale structures have been generated and are very discernable. The potential vorticity field develops sharp gradients and filaments that stretch out and roll in on themselves. Qualitatively, our potential vorticity results match closely to the experiments performed by Rognes et al. (2013), Cotter and McRae (2014), and Shipton, Gibson, and Cotter (2018). This same test case will be revisited again in Chapter 4, where we demonstrate the computational advantage of using a hybridizable method over a standard compatible finite element discretization for the Picard solver.



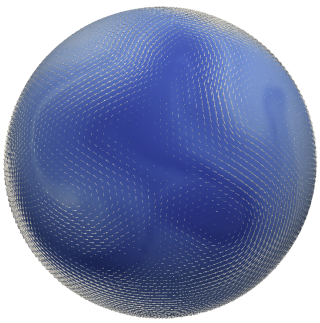
(A) Day 20 (north pole)



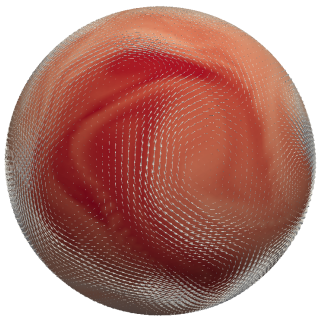
(B) Day 20 (south pole)



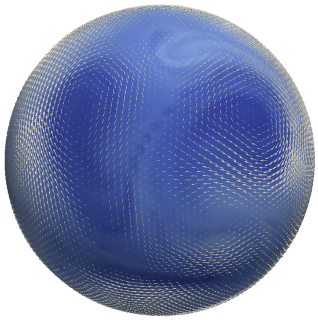
(C) Day 30 (north pole)



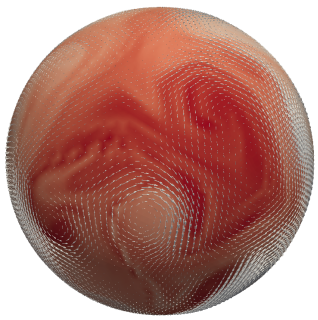
(D) Day 30 (south pole)



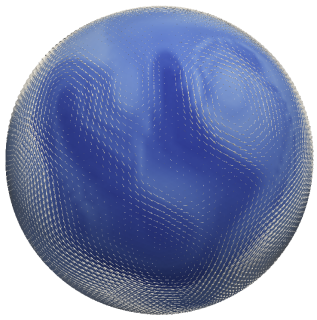
(E) Day 40 (north pole)



(F) Day 40 (south pole)

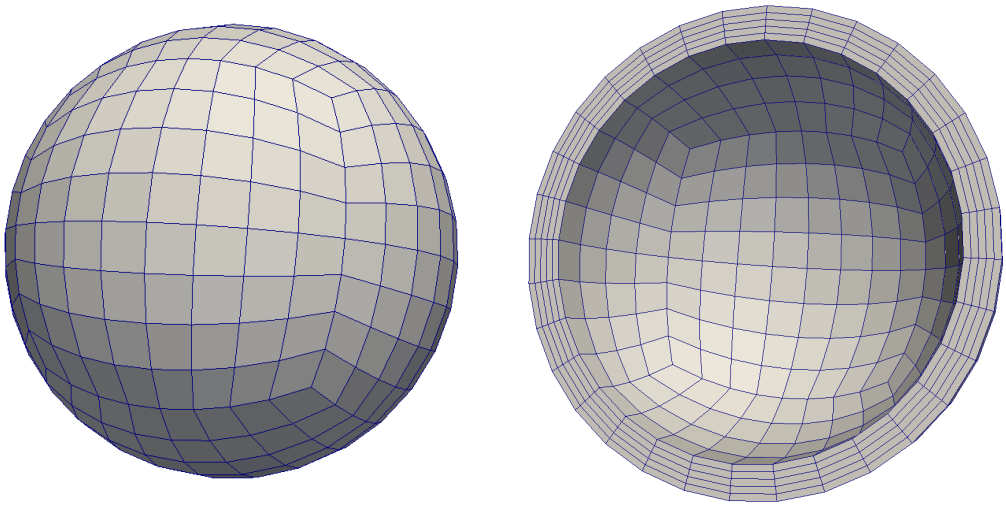


(G) Day 50 (north pole)



(H) Day 50 (south pole)

FIGURE 3.11: Snapshots of the potential vorticity (with velocity fields outlined). The magnitudes of the vorticity range between -3×10^{-8} (blue) and 3×10^{-8} (red).



(A) Horizontal base mesh.

(B) Vertically extruded mesh.

FIGURE 3.12: An $8 \times 8 \times 8$ cubed sphere grid (3.12A) and a uniformly extruded grid with 5 vertical layers (3.12B).

3.4 Other hybridizable discretizations

In preparation for later chapters, we turn our focus to applications which are particularly relevant for fully three-dimensional models. While most of the discussion in this section also applies to two-dimensional applications (such as vertical-slice models), we emphasize problems defined on so-called *atmospheric-shaped* meshes. Such a mesh $\mathcal{T}_h$ consists of polygonal cells K, where each $K = K^{\text{horiz.}} \times K^{\text{vert.}}$ is the product of some horizontal cell $K^{\text{horiz.}}$ with a vertical cell $K^{\text{vert.}}$. By constructing mesh cells in this way, we can obtain a global mesh suitable for vertical staggering. See Section 2.4.3 for a detailed discussion on how these product cells are defined in general.

For the rest of this section and a majority of Chapter 5, the domain will be a spherical annulus with uniform radius R and height H. The domain can therefore be expressed as the product: $\mathcal{S}(R) \times [0, H]$, where $\mathcal{S}(R)$ denotes the surface of a sphere with radius R. Constructing a mesh $\mathcal{T}_h$ will then consist of an unstructured horizontal "base" mesh for $\mathcal{S}(R)$, which is vertically extruded using a structured one-dimensional mesh of $[0, H]$. See Figure 3.12 for an example of such a mesh. Note that it is possible within our framework to have varying resolution in the vertical. However, we shall restrict our discussion to the uniform resolution case.

To construct the compatible finite element spaces, we use the tensor product constructions presented in Section 2.4.4. That is, we build discretizations from the de-Rham complex in three-dimensions:

$$W_h^0 \xrightarrow{\nabla} W_h^1 \xrightarrow{\nabla\times} W_h^2 \xrightarrow{\nabla\cdot} W_h^3. \tag{3.183}$$

Each finite element space is obtained by taking the tensor product of finite elements in the one- and two-dimensional complexes:

$$V_h^0 \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}x}} V_h^1, \quad U_h^0 \xrightarrow{\nabla^\perp} U_h^1 \xrightarrow{\nabla\cdot} U_h^2, \tag{3.184}$$

where the spaces V_h^i consist of finite elements defined on intervals, and U_h^i are defined on two-dimensional polygons making up the horizontal mesh. The resulting tensor product spaces are defined on prismatic cells and have the form:

$$W_h^0 = U_h^0 \otimes V_h^0, \tag{3.185}$$

$$W_h^1 = \underbrace{\texttt{HCurl}(U_h^1 \otimes V_h^0)}_{W_h^{1,\text{horiz.}}} \oplus \underbrace{\texttt{HCurl}(U_h^0 \otimes V_h^1)}_{W_h^{1,\text{vert.}}}, \tag{3.186}$$

$$W_h^2 = \underbrace{\texttt{HDiv}(U_h^1 \otimes V_h^1)}_{W_h^{2,\text{horiz.}}} \oplus \underbrace{\texttt{HDiv}(U_h^2 \otimes V_h^0)}_{W_h^{2,\text{vert.}}}, \tag{3.187}$$

$$W_h^3 = U_h^2 \otimes V_h^1. \tag{3.188}$$

We refer the reader to Section 2.4.4 for a complete discussion on the construction of each W_h^i. For the rest of this section, we describe two new hybridization methods which follow from similar arguments made in the development of the hybridizable method for the shallow water equations.

3.4.1 Hydrostatic pressure equation

A very standard thing to do in geophysical simulations is to solve for a hydrostatic pressure with a vertical component of the pressure gradient that balances the gravitational force, given a pressure boundary condition at the top of the domain and a slip boundary condition for the velocity at the bottom of the domain ($\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on the bottom portion of the domain). The basic pressure structure of the atmosphere is determined by the hydrostatic balance relation:

$$c_p \theta \nabla \Pi = -g\hat{\boldsymbol{k}}, \tag{3.189}$$

where c_p is the specific heat at constant pressure, θ is the potential termperature, Π is the Exner pressure, and $\hat{\boldsymbol{k}}$ is the upward-pointing normal vector. For this reason, it is standard practice to initialize atmospheric simulations using a balanced pressure field. See, for example, Jablonowski and Williamson (2006) and Ullrich et al. (2013).

In a staggered finite difference formulation, this is quite simple, as the pressure values can be solved in each column from the top down to the bottom by computing discrete differences (see Figure 2.20 for the finite difference vertical staggering schemes). In a compatible finite element formulation, this is more complicated, because it requires the solution of a system with different test and trial spaces (hence the resulting discrete system is non-square). In this section, we will describe how this can be dealt with; it will result in a set of decoupled column-wise systems that can also be reduced using a hybridization procedure.

A vertical discretization using compatible finite elements

In the compressible Euler model, the velocity equation takes the form

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -c_p \theta \nabla \Pi - g\hat{\boldsymbol{k}}, \tag{3.190}$$

where $\boldsymbol{\Omega}$ is Coriolis vector (see (2.12)). When in hydrostatic balance, all terms involving $\boldsymbol{u}$ vanish and we are left with (3.189).[12] Now let Ω be the computational domain with boundary $\partial\Omega = \partial\Omega_t \cup \partial\Omega_b$, mesh $\mathcal{T}_h$, and skeleton $\mathcal{E}_h$. We define $\mathring{W}_h^2$ to be the subspace of W_h^2 with vanishing normal components on $\partial\Omega_b$, the bottom of the domain. We now discretize in space by multiplying (3.189) by $\boldsymbol{w} \in \mathring{W}_h^2$ and integrating by parts:

$$\begin{aligned} 0 = &\int_{\mathcal{T}_h} c_p \nabla_h \cdot (\theta \boldsymbol{w}) \Pi_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{E}_h^\circ} c_p [\![\theta \boldsymbol{w}]\!] \{\!\{\Pi_h\}\!\} \,\mathrm{d}S \\ &- \int_{\partial\Omega_t} c_p \theta \boldsymbol{w} \cdot \boldsymbol{n} \Pi_0 \,\mathrm{d}S - \int_{\mathcal{T}_h} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \qquad \forall \boldsymbol{w} \in \mathring{W}_h^2, \end{aligned} \tag{3.191}$$

where Π_0 is the required value of Π_h on the upper boundary $\partial\Omega_t$, $\Pi_h \in W_h^3$ is the pressure variable to be determined, and θ is a known potential temperature field constructed in the space $W_h^\theta = U_h^2 \otimes V_h^0$ (see Section 2.4.5). We also introduce the averaging operator for scalar functions as the single-valued function on the interior skeleton $\mathcal{E}_h^\circ = \mathcal{E}_h \setminus \partial\Omega$:

$$\{\!\{\psi\}\!\} = \frac{1}{2}\left(\psi|_{K^+} + \psi|_{K^-}\right), \tag{3.192}$$

where $\psi|_{K^\pm}$ denotes the restriction of ψ to the $K^\pm$-side of a facet $e \in \mathcal{E}_h^\circ$. The operator $[\![\cdot]\!]$ is the usual jump operator for the normal components of vector fields as defined in previous discussions.

The facet term $[\![\theta \boldsymbol{w}]\!]\{\!\{\Pi_h\}\!\}$ on $\mathcal{E}_h^\circ$ does not vanish because we take $\theta \in W_h^\theta$ and $\Pi_h \in W_h^3$; θ is therefore not continuous in the horizontal direction and Π_h is completely discontinuous across elements. However, the jump term $[\![\theta \boldsymbol{w}]\!]$ vanishes on horizontal interfaces (top/bottom facets of the cells) within a column. This is due to the fact that θ is constructed in a vertically-continuous finite element space.

We can use the continuity of θ to our advantage. Observe that the velocity space in (3.187) admits the following decomposition: $W_h^2 = W_h^{2,\text{horiz.}} \oplus W_h^{2,\text{vert.}}$. Since the balance relation (3.189) is purely vertical, we may consider only test functions in $W_h^{2,\text{vert.}}$.

Now let $\mathcal{C} \subset \mathcal{T}_h$ denote a vertical column of cells. We write $W_h^{2,\text{vert.}}(\mathcal{C})$, $W_h^\theta(\mathcal{C})$ and $W_h^3(\mathcal{C})$ to denote the restrictions of the finite element spaces to the column $\mathcal{C}$. Now we multiply (3.189) by $\boldsymbol{w} \in \mathring{W}_h^{2,\text{vert.}}(\mathcal{C})$. After integrating by parts within a column, we obtain the following finite element problem: find $\Pi_h \in W_h^3(\mathcal{C})$ such that

$$\begin{aligned} \int_{\mathcal{C}} c_p \nabla_h \cdot (\theta \boldsymbol{w}) \Pi_h \,\mathrm{d}\boldsymbol{x} = &- \int_{\partial\mathcal{C}_t} c_p \theta \boldsymbol{w} \cdot \boldsymbol{n} \Pi_0 \,\mathrm{d}S \\ &- \int_{\mathcal{C}} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \qquad \forall \boldsymbol{w} \in \mathring{W}_h^{2,\text{vert.}}(\mathcal{C}), \end{aligned} \tag{3.193}$$

where $\mathring{W}_h^{2,\text{vert.}}(\mathcal{C}) \subset W_h^{2,\text{vert.}}$ is the subspace with vanishing normal components on $\partial\mathcal{C}_b$. There is no coupling between adjacent columns since all arguments are discontinuous in the horizontal direction. Coupling only occurs in the vertical due to the continuity of $\boldsymbol{w} \cdot \boldsymbol{n}$ and θ through the top/bottom facets in $\mathcal{C}$. An illustration is provided in Figure 3.13.

[12] We are not considering the effects of the centrifugal force, $\boldsymbol{\Omega} \times \left(\boldsymbol{\Omega} \times \hat{\boldsymbol{k}}\right)$, since it is measurably small compared to all other terms. For this reason, it is typically neglected in atmospheric models.

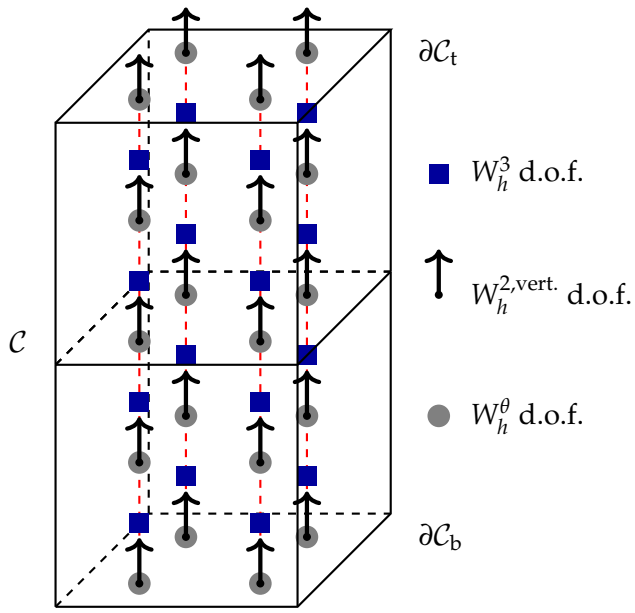


FIGURE 3.13: An illustration of the degrees of freedom (d.o.f.) for $\boldsymbol{w} \in W_h^{2,\mathrm{vert.}}$, $\theta \in W_h^\theta$, and $\Pi_h \in W_h^3$ in a single column $\mathcal{C}$ with two levels. The vertical element $W_h^{2,\mathrm{vert.}}$ corresponds to the vertical component of a Raviart-Thomas-Nédélec element of order $q = 2$. The element W_h^θ is the scalar version of $W_h^{2,\mathrm{vert.}}$ and W_h^3 is a dQ_1 element.

Currently, (3.193) is a system with different test and trial functions. As noted by Natale, Shipton, and Cotter (2016), observe that if $\nabla_h \cdot (\theta \boldsymbol{v}_h) = 0$, for some $\boldsymbol{v}_h \in \mathring{W}_h^{2,\mathrm{vert.}}(\mathcal{C})$, then we have $\boldsymbol{v}_h = \boldsymbol{0}$ since the potential temperature θ is strictly positive and $\boldsymbol{v}_h \cdot \boldsymbol{n} = 0$ on $\partial\mathcal{C}_\mathrm{b}$. This means we can add a term involving $\boldsymbol{v}_h$ without changing the solution for Π_h.

We can therefore convert (3.193) into an equivalent mixed system for $(\boldsymbol{v}_h, \Pi_h) \in \mathring{W}_h^{2,\mathrm{vert.}}(\mathcal{C}) \times W_h^3(\mathcal{C})$ satisfying

$$\int_\mathcal{C} \left(\boldsymbol{w} \cdot \boldsymbol{v}_h - c_p \nabla_h \cdot (\theta \boldsymbol{w}) \Pi_h\right) \mathrm{d}\boldsymbol{x} = -\int_{\partial\mathcal{C}_\mathrm{t}} \theta \boldsymbol{w} \cdot \boldsymbol{n} \Pi_0 \,\mathrm{d}S - \int_\mathcal{C} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \tag{3.194}$$

$$\int_\mathcal{C} c_p \phi \nabla_h \cdot (\theta \boldsymbol{v}_h) \,\mathrm{d}\boldsymbol{x} = 0, \tag{3.195}$$

for all $\boldsymbol{w} \in \mathring{W}_h^{2,\mathrm{vert.}}(\mathcal{C})$ and $\phi \in W_3$. This defines a mixed finite element problem in each vertical column; it is similar to the mixed formulation of the Poisson equation with the modified divergence operator $\boldsymbol{v}_h \mapsto \nabla_h \cdot (\theta \boldsymbol{v}_h)$. Natale, Shipton, and Cotter (2016) gave a uniqueness proof for $(\boldsymbol{v}_h, \Pi_h)$, demonstrating that (3.194)–(3.195) is well-posed and that Π_h solves (3.193). The advantage of using (3.194)–(3.195) over (3.191) is that we now have a well-posed square system, which is column-wise independent. It does, however, require the solution of a saddle-point system in each column. We shall remedy this by introducing a new hybridizable method for the mixed problem.

A vertically-oriented hybridizable method

We proceed in a similar fashion to the development of a hybridizable method for the shallow water equation. First, let us summarize the *global* problem for (3.194)–(3.195). We wish to develop a hybridizable method for the following mixed finite element problem: find $(\boldsymbol{v}_h, \Pi_h) \in \mathring{W}_h^{2,\text{vert.}} \times W_h^3$ such that

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{v}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} c_p \nabla_h \cdot (\theta \boldsymbol{w}) \Pi_h \,\mathrm{d}\boldsymbol{x} = -\int_{\partial\Omega_t} \theta \boldsymbol{w} \cdot \boldsymbol{n} \Pi_0 \,\mathrm{d}S - \int_{\mathcal{T}_h} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \tag{3.196}$$

$$\int_{\mathcal{T}_h} c_p \phi \nabla_h \cdot (\theta \boldsymbol{v}_h) \,\mathrm{d}\boldsymbol{x} = 0, \tag{3.197}$$

for all $\boldsymbol{w} \in \mathring{W}_h^{2,\text{vert.}}$ and $\phi \in W_3$. The global finite element spaces are defined as

$$\mathring{W}_h^{2,\text{vert.}} = \{\boldsymbol{w} \in H(\text{div}; \Omega) : \boldsymbol{w}|_K \in W_2^v(K), \forall K \in \mathcal{T}_h, \boldsymbol{w} \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega_b\}, \tag{3.198}$$

$$W_h^3 = \{\phi \in L^2(\Omega) : \phi|_K \in W_3(K), \forall K \in \mathcal{T}_h\}, \tag{3.199}$$

where the local spaces of shape functions have the form

$$W_2^v(K) = [U_2(K^{\text{horiz.}})\hat{\boldsymbol{k}}] \otimes V_0(K^{\text{vert.}}), \quad W_3(K) = U_2(K^{\text{horiz.}}) \otimes V_1(K^{\text{vert.}}). \tag{3.200}$$

Here, we are using the definition of $\otimes$ for finite elements, as detailed in Section 2.4.3. The spaces $W_2^v(K)$ and $W_3(K)$ are defined on *product cells* of the form $K = K^{\text{horiz.}} \times K^{\text{vert.}}$. The cells $K^{\text{horiz.}}$ and $K^{\text{vert.}}$ denote the domains of the local polynomial spaces U_i and V_j of U_h^i and V_h^j respectively.

Since $W_h^{2,\text{vert.}}$ is an $H(\text{div})$ finite element, functions in $W_h^{2,\text{vert.}}$ are mapped to the reference cell via the contravariant Piola transform. W_h^3 is mapped via the usual change of coordinate mapping. Note that on product cells, the change of coordinates is no longer affine (Natale, Shipton, and Cotter, 2016). We refer the interested reader to McRae et al. (2016) for more details on the definition and implementation of the tensor product spaces discussed here (see also Table 2.2 in Section 2.4.4 for a general reference).

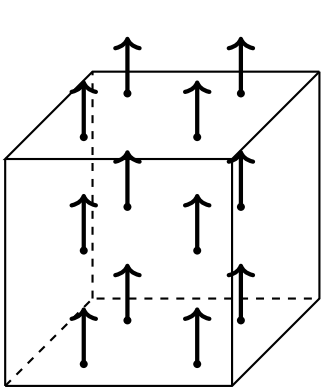
First, we introduce the discontinuous variant of $W_h^{2,\text{vert.}}$, defined as the set

$$\widehat{W}_h^{2,\text{vert.}} = \{\boldsymbol{w} \in [L^2(\Omega)]^d : \boldsymbol{w}|_K \in W_2^v(K), \forall K \in \mathcal{T}_h\}. \tag{3.201}$$

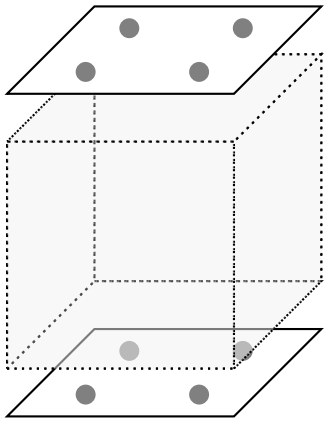
Similar to the standard hybridization method in Section 3.1.2, functions in $\widehat{W}_h^{2,\text{vert.}}$ are no longer required to have continuous normal components across the horizontal facets of the mesh. With $\mathcal{E}_h^v$ denoting the set of horizontally-aligned (top/bottom) facets of all $K \in \mathcal{T}_h$, we define the space of approximate traces as the set

$$W_h^{\text{tr,vert.}} = \{\gamma \in L^2(\mathcal{E}_h^v) : \gamma|_e \in M(e), \forall e \in \mathcal{E}_h^v\}, \tag{3.202}$$

where $M(e)$ is a polynomial space on the facet e. The degree of $M(e)$ is determined by the polynomial degree of $\boldsymbol{w} \cdot \boldsymbol{n}|_e$, for $\boldsymbol{w} \in W_h^{2,\text{vert.}}$, as illustrated in Figure 3.14. Functions in $W_h^{\text{tr,vert.}}$ only have support on the horizontal facets between neighbouring elements in the same column.



(A) The vertical velocity element: $W_h^{2,\text{vert.}}$



(B) The vertical trace element: $W_h^{\text{tr,vert.}}$

FIGURE 3.14: A vertical velocity element is shown in 3.14A with normal components on the horizontally-aligned facets belonging to $\mathcal{Q}_1(e)$. The corresponding trace element on the horizontal facets is shown in 3.14B.

Now we discretize using test functions in the broken $H(\text{div})$ space $\widehat{W}_h^{2,\text{vert.}}$. With $\mathcal{E}^v(\mathcal{C})$ denoting the set of horizonally-aligned facets in a column $\mathcal{C} \subset \mathcal{T}_h$, we have the following extended finite element problem in $\mathcal{C}$: find $(\boldsymbol{v}_h, \Pi_h, \lambda_h) \in \widehat{W}_h^{2,\text{vert.}}(\mathcal{C}) \times W_h^3(\mathcal{C}) \times W_h^{\text{tr,vert.}}(\mathcal{E}^v(\mathcal{C}))$ such that

$$\begin{aligned}\int_{\mathcal{C}} \widehat{\boldsymbol{w}} \cdot \boldsymbol{v}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{C}} c_p \nabla_h \cdot (\theta \widehat{\boldsymbol{w}}) \Pi_h \,\mathrm{d}\boldsymbol{x} \\ + \int_{\mathcal{E}^v(\mathcal{C}) \setminus \partial\mathcal{C}_\mathrm{t}} c_p \lambda_h [\![\theta \widehat{\boldsymbol{w}}]\!] \,\mathrm{d}S = -\int_{\partial\mathcal{C}_\mathrm{t}} \theta \widehat{\boldsymbol{w}} \cdot \boldsymbol{n} \Pi_0 \,\mathrm{d}S \\ - \int_{\mathcal{C}} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \end{aligned} \tag{3.203}$$

$$\int_{\mathcal{C}} c_p \phi \nabla_h \cdot (\theta \boldsymbol{v}_h) \,\mathrm{d}\boldsymbol{x} = 0, \tag{3.204}$$

$$\int_{\mathcal{E}^v(\mathcal{C}) \setminus \partial\mathcal{C}_\mathrm{t}} c_p \gamma [\![\theta \boldsymbol{v}_h]\!] \,\mathrm{d}S = 0, \tag{3.205}$$

for all $(\widehat{\boldsymbol{w}}, \phi, \gamma) \in \widehat{W}_h^{2,\text{vert.}}(\mathcal{C}) \times W_h^3(\mathcal{C}) \times W_h^{\text{tr,vert.}}(\mathcal{E}^v(\mathcal{C}))$. The surface term now reappears due to the fact that $\widehat{\boldsymbol{w}}$ has discontinuous normal components. The Lagrange multipler λ_h appears in the surface terms as an approximation to Π on the facets of $\mathcal{E}^v(\mathcal{C})$. Since $c_p, \theta > 0$, (3.205) is nothing more than the jump condition enforcing the continuity of $\boldsymbol{v}_h \cdot \boldsymbol{n}$ on $\mathcal{E}^v(\mathcal{C})$. Note that (3.205) is also enforcing the boundary condition $\boldsymbol{v}_h \cdot \boldsymbol{n} = 0$ on $\partial\mathcal{C}_\mathrm{b}$ by construction:

$$\int_{\mathcal{E}^v(\mathcal{C}) \setminus \partial\mathcal{C}_\mathrm{t}} c_p \gamma [\![\theta \boldsymbol{v}_h]\!] \,\mathrm{d}S = \int_{\mathcal{E}^v(\mathcal{C}) \setminus \partial\mathcal{C}} c_p \gamma [\![\theta \boldsymbol{v}_h]\!] \,\mathrm{d}S + \int_{\partial\mathcal{C}_\mathrm{b}} c_p \gamma \theta \boldsymbol{v}_h \cdot \boldsymbol{n} \,\mathrm{d}S = 0, \tag{3.206}$$

for all $\gamma \in W_h^{\text{tr,vert.}}(\mathcal{E}^v(\mathcal{C}))$. The system (3.203)–(3.205) is now not just column-wise independent, it is *cell-local* by our choice of finite element spaces. We can therefore use static condensation to eliminate $(\boldsymbol{v}_h, \Pi_h)$, producing a reduced system for λ_h in each column.

It can be shown that the solutions $(\boldsymbol{v}_h, \Pi_h)$ of (3.203)–(3.205) also satisfy the original compatible finite element discretization in (3.194)–(3.195). This follows from an *identical* argument made in the proof of Proposition 3. Noting that $\mathring{W}_h^{2,\text{vert.}} \subset \widehat{W}_h^{2,\text{vert.}}$,

taking $\widehat{\boldsymbol{w}} = \boldsymbol{w} \in \mathring{W}_h^{2,\text{vert.}}$ in (3.203) reduces to (3.194) since $[\![\theta \boldsymbol{w}]\!] = 0$ on $\mathcal{E}^v(\mathcal{C})$. By Lemma 2, the jump condition implies that the solution $\boldsymbol{v}_h$ to the hybridizable formulation is actually in $W_h^{2,\text{vert.}}$. Since the boundary condition is also satisfied through (3.205), we have that $\boldsymbol{v}_h$ belongs to $\mathring{W}_h^{2,\text{vert.}}$ as desired. So we have the hybridizable solutions $(\boldsymbol{v}_h, \Pi_h)$ satisfying all equations of (3.194)–(3.195). The uniqueness of solutions follows from Natale, Shipton, and Cotter (2016, Theorem 7.1).

The solution strategy for solving (3.203)–(3.205) is summarized as follow. The global matrix system has the form:

$$\begin{bmatrix} \boldsymbol{M}_2^v & -\boldsymbol{B}^T & \boldsymbol{K}^T \\ \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{K} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{V} \\ \boldsymbol{\Pi} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} -\boldsymbol{F}(g, \Pi_0) \\ \boldsymbol{0} \\ \boldsymbol{0} \end{Bmatrix}, \tag{3.207}$$

where $\boldsymbol{V}$, $\boldsymbol{\Pi}$, and $\boldsymbol{\Lambda}$ are the coefficient vectors for $\boldsymbol{v}_h$, Π_h, and λ_h respectively. The matrices and vectors are defined through the usual finite element manner. With $\widehat{\mathcal{B}}_2^v$, $\mathcal{B}_3$ and $\mathcal{B}_2^{\text{tr},v}$ denoting bases for $\widehat{W}_h^{2,\text{vert.}}$, W_h^3, and $W_h^{\text{tr,vert.}}$ respectively, the operators are defined via:

$$(\boldsymbol{M}_2^v)_{ij} = \int_{\mathcal{T}_h} \boldsymbol{\Psi}_j \cdot \boldsymbol{\Psi}_i \, \mathrm{d}\boldsymbol{x}, \qquad \boldsymbol{\Psi}_i, \boldsymbol{\Psi}_j \in \widehat{\mathcal{B}}_2^v, \tag{3.208}$$

$$(\boldsymbol{B})_{ij} = \int_{\mathcal{T}_h} c_p \Phi_j \nabla_h \cdot (\theta \boldsymbol{\Psi}_i) \, \mathrm{d}\boldsymbol{x}, \qquad (\boldsymbol{\Psi}_i, \Phi_j) \in \widehat{\mathcal{B}}_2^v \times \mathcal{B}_3, \tag{3.209}$$

$$(\boldsymbol{K})_{ij} = \int_{\mathcal{E}_h^v \setminus \partial\Omega_t} c_p \xi_j [\![\theta \boldsymbol{\Psi}_i]\!] \, \mathrm{d}S, \qquad (\boldsymbol{\Psi}_i, \xi_j) \in \widehat{\mathcal{B}}_2^v \times \mathcal{B}_2^{\text{tr},v}, \tag{3.210}$$

$$(\boldsymbol{F}(g, \Pi_0))_j = \int_{\mathcal{T}_h} g \boldsymbol{\Psi}_j \cdot \hat{\boldsymbol{k}} \, \mathrm{d}\boldsymbol{x} + \int_{\partial\Omega_t} \theta \boldsymbol{\Psi}_j \cdot \boldsymbol{n} \Pi_0 \, \mathrm{d}S, \qquad \boldsymbol{\Psi}_j \in \widehat{\mathcal{B}}_2^v. \tag{3.211}$$

Since all prognostic variables are local to the cell, we can eliminate $\boldsymbol{V}$ and $\boldsymbol{\Pi}$ element-wise to produce the condensed problem:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_2^v & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{K}^T \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_2^v & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{Bmatrix} c\boldsymbol{F}(g, \Pi_0) \\ \boldsymbol{0} \end{Bmatrix}. \tag{3.212}$$

Since the Lagrange multipliers are only coupled through vertical columns, (3.212) can be inverted column-wise. Once $\boldsymbol{\Lambda}$ is determined, both $\boldsymbol{V}$ and $\boldsymbol{\Pi}$ can be recovered element-by-element:

$$\begin{Bmatrix} \boldsymbol{V} \\ \boldsymbol{\Pi} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{M}_2^v & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix}^{-1} \left(\begin{Bmatrix} c\boldsymbol{F}(g, \Pi_0) \\ \boldsymbol{0} \end{Bmatrix} - \begin{bmatrix} \boldsymbol{K}^T \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda} \right). \tag{3.213}$$

The hybridizable method shown in this section was first presented by Gibson et al. (2019a) as an alternative to solving the mixed formulation in (3.194)–(3.195). It was further extended for use within a compatible finite element model for a moist compressible atmosphere by Bendall et al. (2019). All experiments agree with previous studies on standard test cases by Skamarock and Klemp (1994), Bryan and Fritsch (2002), and Ullrich, Reed, and Jablonowski (2015). We will feature the application of this new hybridizable method in the numerical experiments of Chapter 5.

3.4.2 Linear gravity wave system

In this section, we consider a three-dimensional linear compressible Boussinesq model. This simplified equation set exhibits non-hydrostatic and compressible effects while avoiding the full complexity of the pressure gradient term, equation of state, and nonlinearity of the full compressible Euler system. This model was used by Skamarock and Klemp (1994), for example, to explore aspects of time integration in a simplified setting. This same system also provides an effective test case for the development of implicit solvers, as shown by Mitchell and Müller (2016).

The full linear model is summarized as the following system of PDEs:

$$\frac{\partial \boldsymbol{u}}{\partial t} + f\hat{\boldsymbol{k}} \times \boldsymbol{u} = -\nabla p + b\hat{\boldsymbol{k}}, \tag{3.214}$$

$$\frac{\partial p}{\partial t} = -c^2 \nabla \cdot \boldsymbol{u}, \tag{3.215}$$

$$\frac{\partial b}{\partial t} = -N^2 \boldsymbol{u} \cdot \hat{\boldsymbol{k}}, \tag{3.216}$$

where $\boldsymbol{u}$ is the velocity, p is the pressure, b is the buoyancy (how light a parcel of fluid is compared to a reference value), f is the Coriolis parameter, $\hat{\boldsymbol{k}}$ is the upward normal, c the speed of sound, and N the buoyancy frequency. We call these equations the *linear gravity wave system*. For simplicity we assume that both c and N are constant, and enforce the slip boundary condition:

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0, \tag{3.217}$$

at the upper and lower boundary of the atmosphere ($\partial\Omega_\mathrm{t}$ and $\partial\Omega_\mathrm{b}$ respectively).

Compatible finite element formulation

Following similarly to our development of a hybridizable method for the shallow water equations in Section 3.1, we start by discretizing (3.214)–(3.216) in time using the implicit midpoint rule:

$$\frac{\boldsymbol{u}^n - \boldsymbol{u}^{n-1}}{\Delta t} + f\hat{\boldsymbol{k}} \times \left(\frac{\boldsymbol{u}^n + \boldsymbol{u}^{n-1}}{2}\right) + \nabla\left(\frac{p^n + p^{n-1}}{2}\right) - \left(\frac{b^n + b^{n-1}}{2}\right)\hat{\boldsymbol{k}} = 0, \tag{3.218}$$

$$\frac{p^n - p^{n-1}}{\Delta t} + c^2 \nabla \cdot \left(\frac{\boldsymbol{u}^n + \boldsymbol{u}^{n-1}}{2}\right) = 0, \tag{3.219}$$

$$\frac{b^n - b^{n-1}}{\Delta t} + N^2 \left(\frac{\boldsymbol{u}^n + \boldsymbol{u}^{n-1}}{2}\right) \cdot \hat{\boldsymbol{k}} = 0. \tag{3.220}$$

To simplify our notation, we introduce the definitions: $\delta\boldsymbol{u} := \boldsymbol{u}^n - \boldsymbol{u}^{n-1}$, $\delta p := p^n - p^{n-1}$, $\delta b := b^n - b^{n-1}$ and set $\boldsymbol{u}^0 = \boldsymbol{u}^{n-1}$, $p^0 = p^{n-1}$, and $b^0 = b^{n-1}$. Then (3.218)–(3.220) can be rewritten as the following semi-discrete system:

$$\delta\boldsymbol{u} + \frac{\Delta t}{2} f\hat{\boldsymbol{k}} \times \delta\boldsymbol{u} + \nabla\delta p - \delta b\hat{\boldsymbol{k}} = -\Delta t f\hat{\boldsymbol{k}} \times \boldsymbol{u}^0 - \Delta t \nabla p^0 + \Delta t b^0 \hat{\boldsymbol{k}} =: \boldsymbol{r}_{\boldsymbol{u}} \tag{3.221}$$

$$\delta p + \frac{\Delta t}{2} c^2 \nabla \cdot \delta\boldsymbol{u} = -\Delta t c^2 \nabla \cdot \boldsymbol{u}^0 =: r_p, \tag{3.222}$$

$$\delta b + \frac{\Delta t}{2} N^2 \delta\boldsymbol{u} \cdot \hat{\boldsymbol{k}} = -\Delta t N^2 \boldsymbol{u}^0 \cdot \hat{\boldsymbol{k}} =: r_b. \tag{3.223}$$

As noted by Mitchell and Müller (2016), the semi-discrete equation:

$$\delta b + \frac{\Delta t}{2} N^2 \delta \boldsymbol{u} \cdot \hat{\boldsymbol{k}} + \Delta t N^2 \boldsymbol{u}^0 \cdot \hat{\boldsymbol{k}} = 0 \tag{3.224}$$

holds even in the presence of orography. In this case, the upward-pointing normal $\hat{\boldsymbol{k}}$ will need to follow the terrain along the domain boundary $\partial\Omega_\mathrm{b}$. At any rate, we devise a two-stage solution process for solving (3.218)–(3.220). This is based on the approach taken by Mitchell and Müller (2016).

First, we eliminate the buoyancy variable δb from the semi-discrete equations (3.221)–(3.223). This produces a system for $\delta \boldsymbol{u}$ and δp:

$$\begin{aligned} \delta \boldsymbol{u} + \frac{\Delta t}{2} f \hat{\boldsymbol{k}} \times \delta \boldsymbol{u} + \frac{\Delta t}{2} N^2 \hat{\boldsymbol{k}} \left(\delta \boldsymbol{u} \cdot \hat{\boldsymbol{k}} \right) + \nabla \delta p &= \boldsymbol{r}_{\boldsymbol{u}} - \Delta t N^2 \hat{\boldsymbol{k}} \left(\boldsymbol{u}^0 \cdot \hat{\boldsymbol{k}} \right) \\ &= \boldsymbol{r}_{\boldsymbol{u}} + r_b \hat{\boldsymbol{k}} =: \tilde{\boldsymbol{r}}_{\boldsymbol{u}}, \end{aligned} \tag{3.225}$$

$$\delta p + \frac{\Delta t}{2} c^2 \nabla \cdot \delta \boldsymbol{u} = r_p. \tag{3.226}$$

We now discretize in space over a mesh $\mathcal{T}_h$ consisting of prismatic cells. The compatible finite element method seeks the velocity, pressure, and buoyancy updates in the spaces:

$$\delta \boldsymbol{u}_h \in \mathring{W}_h^2 = W_h^{2,\mathrm{horiz.}} \oplus \mathring{W}_h^{2,\mathrm{vert.}}, \quad \delta p_h \in W_h^3, \quad \delta b_h \in W_h^\theta, \tag{3.227}$$

as constructed in (3.187) and (3.188). $\mathring{W}_h^2$ denotes the subspace of W_h^2 with vanishing normal components on $\partial\Omega_\mathrm{t} \cup \partial\Omega_\mathrm{b}$. Starting with (3.225)–(3.226), the compatible finite element discretization reads as follows: find $(\delta \boldsymbol{u}_h, \delta p_h) \in \mathring{W}_h^2 \times W_h^3$ satisfying

$$\begin{aligned} &\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \delta \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_{\mathcal{T}_h} \boldsymbol{w} \cdot f \left(\hat{\boldsymbol{k}} \times \delta \boldsymbol{u}_h \right) \,\mathrm{d}\boldsymbol{x} \\ &+ N^2 \frac{\Delta t^2}{4} \int_{\mathcal{T}_h} \hat{\boldsymbol{k}} \cdot \boldsymbol{w} \hat{\boldsymbol{k}} \cdot \delta \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2} \int_{\mathcal{T}_h} \delta p_h \nabla_h \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \tilde{\boldsymbol{r}}_{\boldsymbol{u}}^h \,\mathrm{d}\boldsymbol{x} =: \widetilde{R}_{\boldsymbol{u}}[\boldsymbol{w}], \end{aligned} \tag{3.228}$$

$$\int_{\mathcal{T}_h} \phi \delta p_h \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} c^2 \int_{\mathcal{T}_h} \phi \nabla_h \cdot \delta \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \phi r_p^h \,\mathrm{d}\boldsymbol{x} =: R_p[\phi], \tag{3.229}$$

for all $(\boldsymbol{w}, \phi) \in \mathring{W}_h^2 \times W_h^3$. The discrete residual functions are defined as

$$\tilde{\boldsymbol{r}}_{\boldsymbol{u}}^h = -\Delta t f \hat{\boldsymbol{k}} \times \boldsymbol{u}_h^0 + \widetilde{\nabla}_h \delta p_h^0 + \Delta t b_h^0 \hat{\boldsymbol{k}} - \Delta t N^2 \hat{\boldsymbol{k}} \left(\boldsymbol{u}_h^0 \cdot \hat{\boldsymbol{k}} \right) \tag{3.230}$$

$$r_p^n = -\Delta t c^2 \nabla_h \cdot \boldsymbol{u}_h^0, \tag{3.231}$$

where $\widetilde{\nabla}_h$ is the operator dual to $\nabla_h \cdot$ through integration by parts (see (3.20)).

Once (3.228)–(3.229) is solved, the buoyancy perturbation δb_h can then be recovered using $\delta \boldsymbol{u}_h$. To do this, we discretize (3.223) to produce the following finite element problem: find $\delta b_h \in W_h^\theta$ such that

$$\int_{\mathcal{T}_h} \mu \delta b_h \,\mathrm{d}\boldsymbol{x} = -\Delta t N^2 \int_{\mathcal{T}_h} \mu \boldsymbol{u}_h^0 \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2} N^2 \int_{\mathcal{T}_h} \mu \delta \boldsymbol{u}_h \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}, \tag{3.232}$$

for all $\mu \in W_h^\theta$. Since the coupling of functions in W_h^θ is purely vertical, the matrix system associated with (3.232) can be inverted column-wise. It is also well-conditioned (independent of the grid resolution/time-step) and can be inverted with a small number of conjugate gradient iterations.

The system (3.228)–(3.229) was first considered by Mitchell and Müller (2016) without the Coriolis term ($f = 0$). It was presented as a simplified test case for the development of linear solvers for atmospheric dynamical cores. Solving (3.228)–(3.229) efficiently using a Schur-complement preconditioner requires a good *sparse* approximation to the dense Schur-complement on the pressure space (similar in construction to (3.30)).

In the study conducted by Mitchell and Müller (2016), a diagonal approximation of the mass matrix coupling velocity degrees of freedom was used to construct a sparse approximation to the pressure Schur-complement. A multigrid method was used to invert the preconditioned pressure system within a non-symmetric Krylov method (GMRES) operating on the velocity-pressure mixed system. They demonstrated excellent performance up to 6,144 compute cores, with both mesh- and Δt-independent solver convergence.

As we will see in Chapter 4, introducing the Coriolis term in the linearized equations causes significant problems for preconditioning. It was demonstrated by Gibson et al. (2019b) that the sparse preconditioner of Mitchell and Müller (2016) is no longer parameter robust when $f \neq 0$. Instead, we develop a hybridizable method for (3.228)–(3.229), which allows for easy incorporation of Coriolis effects. This is due to the fact that we no longer require an approximation to a dense pressure operator; the hybridized system obtained after element-wise static condensation is *sparse*. We can therefore focus our efforts on efficiently inverting the condensed system.

Hybridization of the velocity-pressure system

The hybridization of (3.228)–(3.229) was presented in Gibson et al. (2019a) and Gibson et al. (2019b). It is obtained by rendering the velocity updates discontinuous and reinforcing continuity weakly through functions on cell facets. As we have seen with the hybridization of the hydrostatic system (3.194)–(3.195), we shall also use the trace functions to enforce the slip boundary conditions for $\delta \boldsymbol{u}_h$.

Proceeding similarly as before, we let $\widehat{W}_h^2$ denote the broken variant of the $H(\mathrm{div})$ finite element space W_h^2. Next, we introduce the trace space on the skeleton $\mathcal{E}_h$:

$$W_h^{\mathrm{tr}} = \{\gamma \in L^2(\mathcal{E}_h) : \gamma|_e \in M(e), \forall e \in \mathcal{E}_h\}, \tag{3.233}$$

with $M(e)$ chosen to be the polynomial space of order q, where $\boldsymbol{w} \cdot \boldsymbol{n}|_e \in \mathcal{P}_q(e)$ for all $e \in \mathcal{E}_h$. The hybridizable system is then obtained in the usual way. After multiplying (3.225) by a test function $\widehat{\boldsymbol{w}} \in \widehat{W}_h^2$, integrating by parts yields:

$$\begin{aligned} \int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot \delta \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot f \left(\hat{\boldsymbol{k}} \times \delta \boldsymbol{u}_h\right) \,\mathrm{d}\boldsymbol{x} \\ + N^2 \frac{\Delta t^2}{4} \int_{\mathcal{T}_h} \hat{\boldsymbol{k}} \cdot \widehat{\boldsymbol{w}} \hat{\boldsymbol{k}} \cdot \delta \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} \\ - \frac{\Delta t}{2} \int_{\mathcal{T}_h} \delta p_h \nabla_h \cdot \widehat{\boldsymbol{w}} \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \lambda_h [\![\widehat{\boldsymbol{w}}]\!] \,\mathrm{d}S = \widetilde{R}_{\boldsymbol{u}}[\widehat{\boldsymbol{w}}], \end{aligned} \tag{3.234}$$

where the Lagrange multiplier $\lambda_h \in W_h^{\text{tr}}$ is an approximation to $\frac{\Delta t}{2}\delta p|_{\mathcal{E}_h}$. Equation (3.226) remains unchanged. Now, we need a jump condition to close the system. To enforce continuity of the normal components of $\delta \boldsymbol{u}_h \in \widehat{W}_h^2$, we use functions $\gamma \in W_h^{\text{tr}}$ via the condition:

$$\int_{\mathcal{E}_h} \gamma [\![\delta \boldsymbol{u}_h]\!] \, \mathrm{d}S = \int_{\mathcal{E}_h \setminus \partial\Omega} \gamma [\![\delta \boldsymbol{u}_h]\!] \, \mathrm{d}S + \int_{\partial\Omega} \gamma \delta \boldsymbol{u}_h \cdot \boldsymbol{n} \, \mathrm{d}S = 0, \tag{3.235}$$

for all $\gamma \in W_h^{\text{tr}}$. In this way, we also have (3.235) enforcing the boundary condition $\delta \boldsymbol{u}_h \cdot \boldsymbol{n} = 0$ on $\partial\Omega = \partial\Omega_{\text{t}} \cup \partial\Omega_{\text{b}}$.

The full hybridizable method is summarized as follows. Find $(\delta \boldsymbol{u}_n, \delta p_h, \lambda_h) \in \widehat{W}_h^2 \times W_h^3 \times W_h^{\text{tr}}$ satisfying the equations

$$\begin{aligned}
\int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot \delta \boldsymbol{u}_h \, \mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_{\mathcal{T}_h} \widehat{\boldsymbol{w}} \cdot f \left(\hat{\boldsymbol{k}} \times \delta \boldsymbol{u}_h \right) \, \mathrm{d}\boldsymbol{x} \\
+ N^2 \frac{\Delta t^2}{4} \int_{\mathcal{T}_h} \hat{\boldsymbol{k}} \cdot \widehat{\boldsymbol{w}} \hat{\boldsymbol{k}} \cdot \delta \boldsymbol{u}_h \, \mathrm{d}\boldsymbol{x} \\
- \frac{\Delta t}{2} \int_{\mathcal{T}_h} \delta p_h \nabla_h \cdot \widehat{\boldsymbol{w}} \, \mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \lambda_h [\![\widehat{\boldsymbol{w}}]\!] \, \mathrm{d}S = \widetilde{R}_{\boldsymbol{u}}[\widehat{\boldsymbol{w}}], \qquad (3.236) \\
\int_{\mathcal{T}_h} \phi \delta p_h \, \mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} c^2 \int_{\mathcal{T}_h} \phi \nabla_h \cdot \delta \boldsymbol{u}_h \, \mathrm{d}\boldsymbol{x} = R_p[\phi], \qquad (3.237) \\
\int_{\mathcal{E}_h} \gamma [\![\delta \boldsymbol{u}_h]\!] \, \mathrm{d}S = 0, \qquad (3.238)
\end{aligned}$$

for all $(\widehat{\boldsymbol{w}}, \phi, \gamma) \in \widehat{W}_h^2 \times W_h^3 \times W_h^{\text{tr}}$. The corresponding matrix system for (3.236)–(3.238) has the form:

$$\begin{bmatrix} \widetilde{\boldsymbol{M}}_2 & -\frac{\Delta t}{2}\boldsymbol{D}^T & \boldsymbol{K}^T \\ \frac{\Delta t c^2}{2}\boldsymbol{D} & \boldsymbol{M}_3 & \boldsymbol{0} \\ \boldsymbol{K} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{U} \\ \delta \boldsymbol{P} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{R}_{\boldsymbol{u}} \\ \boldsymbol{R}_p \\ \boldsymbol{0} \end{Bmatrix}. \tag{3.239}$$

As we have seen in previous hybridization methods, all degrees of freedom are now local to the cell. Therefore, it is easy to eliminate both $\delta \boldsymbol{U}$ and $\delta \boldsymbol{P}$ to obtain a sparse system for $\boldsymbol{\Lambda}$ via static condensation:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{M}}_2 & -\frac{\Delta t}{2}\boldsymbol{D}^T \\ \frac{\Delta t c^2}{2}\boldsymbol{D} & \boldsymbol{M}_3 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{K}^T \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{M}}_2 & -\frac{\Delta t}{2}\boldsymbol{D}^T \\ \frac{\Delta t c^2}{2}\boldsymbol{D} & \boldsymbol{M}_3 \end{bmatrix}^{-1} \begin{Bmatrix} \boldsymbol{R}_{\boldsymbol{u}} \\ \boldsymbol{R}_p \end{Bmatrix} \tag{3.240}$$

Both $\delta \boldsymbol{U}$ and $\delta \boldsymbol{P}$ are recovered locally as before once $\boldsymbol{\Lambda}$ is determined.

Neither the original system (3.228)–(3.229) or the hybridizable system (3.236)–(3.238) has undergone rigorous analysis yet. It is, however, easily verifiable that the solutions $(\delta \boldsymbol{u}_n, \delta p_h) \in \widehat{W}_h^2 \times W_h^3$ of (3.236)–(3.238) also solve (3.228)–(3.229). The argument follows identically from the discussion on (3.203)–(3.204). A more detailed analysis of the hybridizable equations and the corresponding matrix system for λ_h is a subject of on-going work. Some preliminary studies have already been performed in Section 4.5.3 of Chapter 4, which demonstrates numerical evidence that (3.240) can be inverted effectively using multigrid approaches employed in numerical weather prediction models (Fulton, Ciesielski, and Schubert, 1986). Additionally, hybridization allows for the incorporation of Coriolis effects in the implicit equations without relying on approximate Schur-complement factorizations.

3.5 Chapter summary

In this chapter, we present hybridizable formulations of various compatible finite element discretizations for simplified geophysical models. Emphasis is placed on the formulation, discretization, and characterization of the methods. A complete characterization result is proven for the linear shallow water model. The objective behind that exercise is to better understand, using more familiar constructions, the nature of the Lagrange multiplier and the resulting discrete system. In performing the analysis of the multiplier for the shallow water system, we explicitly constructed the local operators which permits the cell-wise reconstruction of prognostic variables after performing local eliminations. Moreover, the results obtained provide a good starting point for furthering the analysis of other hybridizable discretizations.

In Section 3.3, we showed how hybridization can be used within a nonlinear method. The procedure we outlined mirrors the strategies employed by operational dynamical cores, with a particular interest in the strategies used in the UK Met Office models (Wood et al., 2014; Thuburn, 2016; Melvin et al., 2019; Adams et al., 2019). The results obtained for the nonlinear shallow water tests agree with existing studies.

We purposefully omitted the technical details on the implementation aspects of hybridization so as to not obfuscate the discussion and analysis. Therefore, the next chapter will focus on the computational and performance aspects of these methods. In particular, we present a new framework for automatically generating low-level code for performing static condensation and local reconstructions. The abstraction known as "Slate," is the central topic of Chapter 4. It is sufficiently general to apply to a wide range of problems outside of hybridization. Every hybridizable discretization presented in this Chapter, and throughout the rest of this dissertation, can be implemented rapidly and effectively using this new computational abstraction.

4 An automated framework for hybridization and static condensation

4.1 Introduction

The development of simulation software is an increasingly important aspect of modern scientific computing, in the geosciences in particular. Such software requires a vast range of knowledge spanning several disciplines, ranging from applications expertise to mathematical analysis to high-performance computing and low-level code optimization. Software projects developing automatic code generation systems have become quite popular in recent years, as such systems help create a separation of concerns which focuses on a particular complexity independent from the rest. This allows for agile collaboration between computer scientists with hardware and software expertise, computational scientists with numerical algorithm expertise, and domain scientists such as meteorologists, oceanographers and climate scientists. Examples of such projects in the domain of finite element methods include FreeFEM++ (Hecht, 2012), Sundance (Long, Kirby, and Bloemen Waanders, 2010), the FEniCS Project (Logg, Mardal, and Wells, 2012), Feel++ (Prud'homme et al., 2012), and Firedrake (Rathgeber et al., 2017).

The finite element method (FEM) is a mathematically robust framework for computing numerical solutions of partial differential equations (PDEs) that has become increasingly popular in fluids and solids models across the geosciences, with a formulation that is highly amenable to code-generation techniques. A description of the weak formulation of the PDEs, together with appropriate discrete function spaces, is enough to characterize the finite element problem. Both the FEniCS and Firedrake projects employ the *Unified Form Language* (UFL) (Alnæs et al., 2014) to specify the finite element integral forms and discrete spaces necessary to properly define the finite element problem. UFL is a highly expressive domain-specific language (DSL) embedded in Python, which provides the necessary abstractions for code generation systems.

In Chapter 3, we presented some hybridizable methods for mixed finite element discretizations of equations relevant for atmospheric modeling. In Section 3.3, we presented some numerical results with no mention of the software implementation. The effective implementation of hybridization and static condensation requires invasive intervention in standard finite element codes. In particular, we must inject dense linear algebra routines during finite element assembly in order to perform such operations. To do this in a general way, which can be effectively applied to a wide array of problems, requires software composition using high-level abstractions of the desired mathematics.

In this chapter, we present a novel computational framework; we introduce a simple yet effective high-level abstraction for localized dense linear algebra on systems derived from finite element problems. Using embedded DSL technology, we provide a means to enable the rapid development of hybridization and static condensation techniques within an automatic code-generation framework. In other words, the main contribution of this chapter is in solving the problem of automatically translating from the mathematics of static condensation and hybridization to compiled code. This automated translation facilitates the separation of concerns between applications scientists and computational/computer scientists, and facilitates the automated optimization of compiled code. This work is implemented in the Firedrake finite element library and the PETSc solver library (Balay et al., 1997; Balay et al., 2016), accessed via the Python interface petsc4py (Dalcin et al., 2011).

4.1.1 The Firedrake finite element library

It will first be useful to put everything in context by describing the composition of abstractions employed by Firedrake and how it is used in practice. The main contribution of Firedrake, as a piece of mathematical software, is that it extends the decomposition of the finite element method into automated abstractions further than previous approaches. It is the unification of the following high-level frameworks:

1. A Python-embedded domain-specific language for the specification of PDE weak formulations: UFL (Alnæs et al., 2014).
2. The efficient evaluation of finite element basis functions and numerical quadrature: FIAT (Kirby, 2004) and FInAT (Homolya, Kirby, and Ham, 2017).
3. The automatic generation of local assembly code for evaluating finite element integrals, i.e., the Firedrake *compiler* for variational forms: The Two-Stage Form Compiler (TSFC) (Homolya et al., 2018).
4. A uniform abstraction for the specification of iterations over general meshes and global finite element assembly: PyOP2 (Rathgeber et al., 2012).
5. Platform-specific optimizations of generated code specific for finite element methods (COFFEE) (Luporini et al., 2015).[1]
6. An advanced solvers library for the solution of linear and nonlinear systems of equations, PETSc (Balay et al., 1997; Balay et al., 2016), through its Python interface: petsc4py (Dalcin et al., 2011). This includes the PETSc abstraction known as DMPlex (Knepley and Karpeev, 2009), which allows Firedrake to relate mesh entities to finite element degrees of freedom (nodes).

Firedrake provides a complete problem-solving environment in the form of a Python module, which seamlessly interfaces with each of these frameworks. The benefit of this composition of abstractions is that it enables a separation of concerns, allowing for targeted development and relatively easy incorporation of new features and code-generation technology. It also allows Firedrake to benefit directly from contributions made in third party libraries, such as PETSc. Figure 4.1 illustrates the complete Firedrake tool-chain.

[1] Currently in the process of being phased-out due to the migration of equivalent code-generation technology to the form compiler: TSFC.

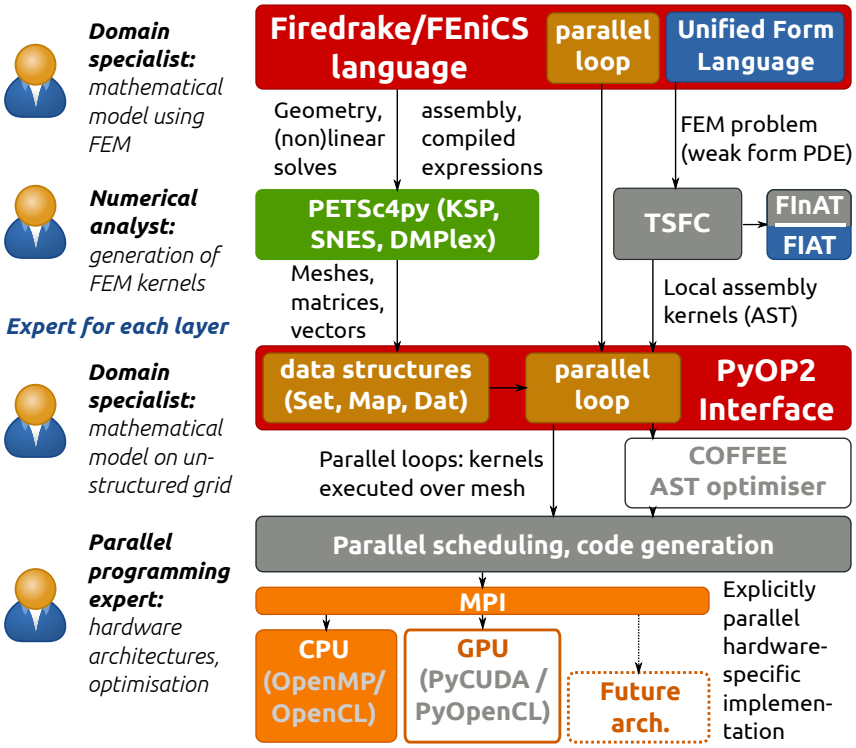


FIGURE 4.1: The Firedrake tool-chain illustrating the composition of abstractions and the separation of concerns that this creates. Interface layers are represented in red, whereas tools adopted from the PETSc and FEniCS projects are in green and blue respectively. PyOP2 objects for controlling the movement of data on unstructured meshes are in brown. Code-generation layers are in grey, and the execution platform on computer hardware is shown in orange. This an updated diagram based on Rathgeber et al. (2017, Figure 1). All frameworks in red, grey, and brown are directly maintained by the Firedrake Project. The frameworks in blue are maintained cooperatively by both the Firedrake and FEniCS projects.

A mixed Poisson example

As a simple demonstration of the high-level problem-solving environment provided by Firedrake, consider the model Poisson problem:

$$-\nabla\cdot\nabla p = f \text{ in } \Omega = [0,1]^2, \quad p = 0 \text{ on } \partial\Omega, \tag{4.1}$$

with a source function given as $f(x,y) = 10\exp\left\{-100\left(\left(x-\frac{1}{2}\right)^2+\left(y-\frac{1}{2}\right)^2\right)\right\}$. Then the mixed formulation of (4.1) is obtained by introducing the negative flux $\boldsymbol{u} = -\nabla p$ and substituting into the problem:

$$\boldsymbol{u} + \nabla p = 0 \text{ in } \Omega, \tag{4.2}$$
$$\nabla\cdot\boldsymbol{u} = f \text{ in } \Omega, \tag{4.3}$$
$$p = 0 \text{ on } \partial\Omega. \tag{4.4}$$

Then given a mesh $\mathcal{T}_h$ of Ω, the mixed finite element formulation of (4.2)–(4.4) seeks approximations $\boldsymbol{u}_h$ and p_h in the finite element subspaces $U_h \times V_h \subset H(\mathrm{div};\Omega) \times$

$L^2(\Omega)$, defined by:

$$U_h = \{\boldsymbol{w} \in H(\mathrm{div};\Omega) : \boldsymbol{w}|_K \in U(K), \forall K \in \mathcal{T}_h\}, \tag{4.5}$$
$$V_h = \{\phi \in L^2(\Omega) : \phi|_K \in V(K), \forall K \in \mathcal{T}_h\}. \tag{4.6}$$

The space U_h consists of $H(\mathrm{div})$-conforming piecewise vector polynomials, where choices of $U(K)$ typically include the Raviart-Thomas (RT), Brezzi-Douglas-Marini (BDM), or Brezzi-Douglas-Fortin-Marini (BDFM) elements (Raviart and Thomas, 1977; Nédélec, 1980; Brezzi, Douglas, and Marini, 1985; Brezzi et al., 1987b). The space V_h is the Lagrange family of discontinuous polynomials. These define the same families of compatible finite element spaces discussed throughout Sections 2.4.2 and 2.4.4.

The discrete finite element problem is arrived at in the usual way. After multiplying (4.2)–(4.3) by test functions and integrating by parts, we have the following discrete problem: find $(\boldsymbol{u}_h, p_h) \in U_h \times V_h$ satisfying

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} p_h \nabla \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x} = 0, \quad \forall \boldsymbol{w} \in U_h, \tag{4.7}$$
$$\int_{\mathcal{T}_h} \phi \nabla \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \phi f \,\mathrm{d}\boldsymbol{x}, \quad \forall \phi \in V_h. \tag{4.8}$$

The matrix system is obtained by expanding the solutions in terms of the finite element bases:

$$\boldsymbol{u}_h = \sum_{i=1}^{N_u} U_i \boldsymbol{\Psi}_i, \quad p_h = \sum_{i=1}^{N_p} P_i \xi_i, \tag{4.9}$$

where $\{\boldsymbol{\Psi}_i\}_{i=1}^{N_u}$ and $\{\xi_i\}_{i=1}^{N_p}$ are bases for U_h and V_h respectively. Here, U_i and P_i are the coefficients to be determined. As per standard Galerkin-based finite element methods, taking $\boldsymbol{w} = \boldsymbol{\Psi}_j$, $j \in \{1, \cdots, N_u\}$ and $\phi = \xi_j$, $j \in \{1, \cdots, N_p\}$ in (4.7)–(4.8) produces the saddle point system:

$$\mathcal{A}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{0} \\ \boldsymbol{F} \end{Bmatrix} = \boldsymbol{b}. \tag{4.10}$$

where $\boldsymbol{U} = \{U_i\}_{i=1}^{N_u}$, $\boldsymbol{P} = \{P_i\}_{i=1}^{N_p}$ are the coefficient vectors, and

$$(\boldsymbol{A})_{ij} = \int_{\mathcal{T}_h} \boldsymbol{\Psi}_i \cdot \boldsymbol{\Psi}_j \,\mathrm{d}\boldsymbol{x}, \quad (\boldsymbol{B})_{ij} = \int_{\mathcal{T}_h} \xi_i \nabla \cdot \boldsymbol{\Psi}_j \,\mathrm{d}\boldsymbol{x}, \quad (\boldsymbol{F})_j = \int_{\mathcal{T}_h} \xi_j f \,\mathrm{d}\boldsymbol{x}. \tag{4.11}$$

The problem described here can be formulated, discretized, and solved in Firedrake with only a few lines of Python code. Listing 4.1 displays a *complete* Firedrake program for solving the model problem (4.2)–(4.4). We shall quickly break down the program here.

Formulating the problem in UFL

A mesh is created in line 3 using one of several built-in utility meshes. This creates a triangular mesh of the unit square by dividing each quadrilateral cell of a uniform 32×32 grid into two triangles. The result is a structured mesh of 2048 triangular cells. Firedrake is also capable of reading meshes generated by third-party libraries,

LISTING 4.1: A complete Firedrake program for solving (4.7)–(4.8) using a Raviart-Thomas mixed method.

```
1  from firedrake import *
2
3  mesh = UnitSquareMesh(32, 32)
4  U = FunctionSpace(mesh, "RT", degree=2)
5  V = FunctionSpace(mesh, "DG", degree=1)
6  W = U * V
7
8  u, p = TrialFunctions(W)
9  w, phi = TestFunctions(W)
10
11 x, y = SpatialCoordinate(mesh)
12 f = 10*exp(-100*((x - 0.5)**2 + (y - 0.5)**2))
13
14 a = dot(w, u)*dx - div(w)*p*dx + phi*div(u)*dx
15 L = phi*f*dx
16
17 w = Function(W, name="solution")
18 solve(a == L, w,
19       solver_parameters={"ksp_type": "gmres",
20                          "ksp_rtol": 1e-8,
21                          "pc_type": "fieldsplit",
22                          "pc_fieldsplit_type": "schur",
23                          "pc_fieldsplit_schur_fact_type": "full",
24                          "fieldsplit_0_ksp_type": "preonly",
25                          "fieldsplit_0_pc_type": "lu",
26                          "fieldsplit_1_ksp_type": "preonly",
27                          "fieldsplit_1_pc_type": "lu"})
```

such as Gmsh (Geuzaine and Remacle, 2009), CGNS (Poirier et al., 1998), Triangle (Shewchuk, 1996), and Exodus II (Schoof and Yarberry, 1994).

The following lines demonstrate how one can express finite element problems in UFL. Lines 4–6 define the finite element spaces for a standard $H(\mathrm{div}) \times L^2$ mixed method using the Raviart-Thomas (`"RT"`) and discontinuous Lagrange (`"DG"`) spaces on the given mesh. The degrees of the spaces are chosen based on the finite element de-Rham complex (2.201) presented in Section 2.4.2. Test and trial functions are defined in lines 8–9, and the source function is expressed in lines 11–12 as a function of the spatial coordinates. The bilinear and linear forms for (4.38)–(4.39) are shown in lines 14–15. A function is created to store the solution of the mixed problem in line 17. The resemblance to the mathematical formulation is immediately apparent.

Code-generation and operator assembly

Up until this point, no code-generation has occurred; everything written is the previous lines are purely symbolic UFL expressions. Once `solve` is called in line 18, all information contained in the UFL expressions gets translated into low-level C code (kernels) by the form compiler: TSFC. The underlying UFL representation is able to provide the form compiler with estimated quadrature rules[2] (Ølgaard and Wells,

[2]This can also be prescribed manually by providing an appropriate quadrature degree to the UFL measure, e.g., `dx(degree=q)`.

2010) along with information about the reference cell and its pullback. Pullbacks are determined by the type of finite element space(s) the test and trial functions are defined on. See Section 2.2.1 for details on the pullbacks of various finite element families.

Using all information contained in the high-level UFL representation, TSFC is able to inspect the UFl forms and generate a local assembly kernel for all the element matrices/vectors in (4.11). During this process, TSFC uses FIAT (The FInite element Automated Tabulator) to pre-compute evaluations of finite element basis functions on the reference cell at quadrature point locations, and incorporates the result into the generated assembly kernel. The code is further optimized by FInAT, which allows Firedrake to produce assembly code with optimal order complexity, such as using sum factorization for finite elements on cuboid cells. This is particularly important for high-order discretizations. More information on this is discussed in great detail by Homolya, Kirby, and Ham (2017).

Once local assembly code is generated, the output result is then mapped into a PETSc sparse data structure (`Mat`/`Vec` objects (Balay et al., 2016, §1.4)) via a generated kernel iterated over the mesh by PyOP2. All generated code is compiled *at runtime*, and stored (cached) to be reused if necessary (for example, in time-dependent simulations). It is worth noting here that PyOP2 actually has no concept of the mesh topology; it works only with indirection maps between sets of mesh entities (cells, faces/edges, vertices) and sets of degrees of freedom. Firedrake derives the required indirection maps for a given mesh through PETSc's DMPlex interface. This is discussed in more detail by Lange et al. (2016).

Solving the linear system and configuring PETSc

Solving (4.7)–(4.8) occurs in line 18, where an optional Python dictionary of solvers options can be passed directly to PETSc. Since configuring the linear solver will be a central topic in this chapter, we have provided an explicit example in lines 19–27. We shall break these commands down for the purpose of exposition. Before we do so, it is worth explaining the *language* PETSc uses to configure linear solvers.

PETSc Options: At first glance, lines 19–27 can be overwhelming. This therefore warrants a more in depth explanation. All PETSc objects are configurable at *runtime*, which allows for rapid experimentation of different solvers without drastically changing the application code. The primary mechanism for controlling the behavior of PETSc solvers is through the *PETSc options database* (Balay et al., 2016, §4). The PETSc "programming language" used to control these options consists of *two* operations:

1. Value assignment; and
2. string concatenation.

Each object has an associated *options prefix*, which can be used to distinguish it from other solver objects. A linear solver object in PETSc is given the nomenclature "`KSP`" (Krylov subspace method) and its options prefix is "`ksp_`". A *preconditioner* is a "`PC`" object, with corresponding options prefix "`pc_`". Looking now at lines 19–27, we can

translate what is actually happening. The first three lines:

```
"ksp_type":   "gmres",
"ksp_rtol":   1e-8,
"pc_type":   "fieldsplit",
```

controls the type of `KSP` and `PC` objects for the full matrix system (4.10), as well as the termination criterial for the `KSP`. The first line says: "Set the type of the KSP object to GMRES," while the second line tells PETSc to terminate the solver when the residual $\boldsymbol{b} - \boldsymbol{\mathcal{A}}\boldsymbol{x}$ has been reduced by a factor of 10^8. The preconditioner wrapped in a `PC` object is configured identically; the third line tells PETSc to set the preconditioner for the `KSP` to be of type: "`fieldsplit`."

A `fieldsplit` preconditioner (Brown et al., 2012) splits the operator at the *global* level into blocks (corresponding to each field). The splitting type is controlled by the options prefix "`pc_fieldsplit_type`." In our example, we chose to use a full Schur-complement factorization of the block matrix:

```
"pc_fieldsplit_type":   "schur",
"pc_fieldsplit_schur_fact_type":   "full",
```

where "`pc_fieldsplit_schur_fact_type`" configures the type of Schur-complement factorization. Mathematically, this produces a preconditioning operator $\mathcal{P}$ of the form:

$$\mathcal{P} = \mathcal{A}^{-1} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}^{-1}\boldsymbol{B}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{B}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{bmatrix} \tag{4.12}$$

where $\boldsymbol{S} = \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$. Schur-complement preconditioners require two global inversions (solves): one for $\boldsymbol{A}$, and another for the Schur-complement $\boldsymbol{S}$. In PETSc, each inversion is *another* solver object, with its own `KSP` and `PC` objects independent from the main "outer" `KSP`.

Each solver for (4.12) are configured using the prefixes: "`fieldsplit_0_`" for inverting $\boldsymbol{A}$, and "`fieldsplit_1_`" for $\boldsymbol{S}$. The last few lines do exactly just that:

```
"fieldsplit_0_ksp_type":   "preonly",
"fieldsplit_0_pc_type":   "lu",
"fieldsplit_1_ksp_type":   "preonly",
"fieldsplit_1_pc_type":   "lu".
```

When we set the "`ksp_type`" to "`preonly`", we are specifically telling PETSc to *not* use an iterative method; instead we just apply the preconditioner. In both cases, we invert $\boldsymbol{A}$ and $\boldsymbol{S}$ using LU factorizations.

Each of these solvers can be configured *exactly* like any other PETSc solver. This enables, quite literally, the arbitrary composition of linear solvers and preconditioners. Although this system may appear cumbersome, it provides an extremely flexible and precise environment for specifying linear solvers. More importantly, switching out linear solvers requires *no modification* of the core Firedrake code itself (lines 1–18 do not change).

Since we inverting both $\boldsymbol{A}$ and $\boldsymbol{S}$ exactly, the options we have specified in Listing 4.1 are actually a *direct method*. In other words, GMRES will converge in one iteration

since the factorization (4.12) is evaluated exactly (up to rounding errors). However, these are not the most prudent choices of solvers options. Recall from our previous discussion on mixed methods that $\boldsymbol{S}$ is globally dense due to the dense inverse $\boldsymbol{A}^{-1}$. Therefore, explicitly forming and inverting $\boldsymbol{S}$ becomes computationally impractical for larger problems. However, this discussion serves as a nice introduction for programming linear solvers in Firedrake. We will use this type of language frequently throughout this chapter.

Extending Firedrake's solver capabilities

Throughout this dissertation, we have been advocating the hybridization method as an efficient alternative to solving compatibled (mixed) finite element problems. While Firedrake provides access to a wide-range of options for solving PDEs, it unfortunately lacks support for the more invasive procedures required to statically condense finite element systems. This is critical for the efficient solution of hybridizable formulations of finite element discretizations. More specifically, prior to the work presented in this chapter, Firedrake has no immediate mechanism for injecting new code within the operator assembly process.

In order to perform static condensation and produce the desired condensed system, we require Firedrake to be able to intervene in global operator assembly to generate a new local expression for the statically condensed system. This now requires computational kernels that not only assemble element-wise tensors, but also *algebraically* manipulating them to form a new element tensor. This is precisely the scope of this chapter. Here, we present a new abstraction framework for these types of procedures: Slate.

4.2 Slate: a system for linear algebra on element tensors

The language, which we call *Slate*, provides typical mathematical operations performed on matrices and vectors, hence the input syntax is comparable to high-level linear algebra software such as MATLAB. The Slate language provides basic abstract building blocks which can be used by a specialized compiler for linear algebra to generate low-level code implementations.

Slate is heavily influenced by UFL (Alnæs et al., 2014; Logg, Mardal, and Wells, 2012), and is functionally designed to be compatible with any UFL form expressions. As we have previously discussed, UFL forms can be compiled by a *form compiler*, which translates UFL into low level code for the local assembly of a form over the cells and facets of a mesh. In a similar manner, Slate expressions are compiled to low level code that performs the requested linear algebra element-wise on a mesh.

4.2.1 An overview of Slate

We begin by establishing notation used throughout this chapter. Let $\mathcal{T}_h$ denote a tessellation of $\Omega \subset \mathbb{R}^n$, the computational domain, consisting of polygonal elements K associated with a mesh size parameter h, exterior boundary $\partial\Omega$, and $\mathcal{E}_h = \{e \subset \partial K \text{ for all } K \in \mathcal{T}_h\}$ the set of facets of $\mathcal{T}_h$. In a slight abuse of notation, we define the set of facets *interior* to the domain by $\mathcal{E}_h^\circ = \mathcal{E}_h \setminus \partial\Omega$. Similarly, we denote the set

of *exterior* facets as $\mathcal{E}_h^\partial = \mathcal{E}_h \cap \partial\Omega$. If the domain boundary is decomposed into the regions $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, then we denote the subset of exterior facets lying on the subdomains of the boundary as $\mathcal{E}_{h,D}^\partial = \mathcal{E}_h^\partial \cap \partial\Omega_D$ (similarly for $\mathcal{E}_{h,N}^\partial$).

To clarify conventions and the scope of Slate, we now introduce our notation for a general finite element form following the convention of Alnæs et al. (2014). We define a real-valued *multi-linear form* as an operator which maps a list of *arguments* $\boldsymbol{v} = (v_0, \cdots, v_{\alpha-1}) \in V_0 \times \cdots \times V_{\alpha-1}$ into $\mathbb{R}$:

$$a : V_0 \times \cdots \times V_{\alpha-1} \to \mathbb{R}, \quad a \mapsto a(v_0, \cdots, v_{\alpha-1}) = a(\boldsymbol{v}), \tag{4.13}$$

where a is linear in each argument v_k. The *arity* of a form is α, an integer denoting the total number of form arguments. In traditional finite element nomenclature (for $\alpha \leq 2$), V_0 is referred to as the space of *test functions* and V_1 as the space of *trial functions*. Each V_k are referred to as *argument spaces*. Forms with arity $\alpha = 0, 1$ or 2 are best interpreted as the more familiar mathematical objects: scalars (0-forms), linear forms or functionals (1-forms), and bilinear forms (2-forms) respectively.

If a given form a is parameterized by one or more *coefficients*, say $\boldsymbol{c} = (c_0, \cdots, c_q) \in C_0 \times \cdots \times C_q$ where $\{C_k\}_{k=0}^q$ are *coefficient spaces*, then we write:

$$a : C_0 \times \cdots \times C_q \times V_0 \times \cdots \times V_{\alpha-1} \to \mathbb{R}, \tag{4.14}$$

$$a \mapsto a(c_0, \cdots, c_q; v_0, \cdots, v_{\alpha-1}) = a(\boldsymbol{c}; \boldsymbol{v}). \tag{4.15}$$

From here on, we shall work exclusively with forms that are linear in $\boldsymbol{v}$ and possibly non-linear in the coefficients $\boldsymbol{c}$. This is reasonable since non-linear methods based on Newton iterations produces linear problems via Gâteaux differentiation of a non-linear form corresponding to a PDE (also known as the *form Jacobian*). We refer the interested reader to Alnæs et al. (2014, §2.1.2) for more details. For clarity, we present examples of multi-linear forms of arity $\alpha = 0, 1$ and 2 that frequently appear in finite element discretizations using our notation:

$$a(\kappa; v, u) := \sum_{K \in \mathcal{T}_h} \int_K \nabla v \cdot (\kappa \nabla u) \, \mathrm{d}\boldsymbol{x}, \qquad \alpha = 2, \quad q = 1, \tag{4.16}$$

$$a(f; v) := \sum_{K \in \mathcal{T}_h} \int_K v \, f \, \mathrm{d}\boldsymbol{x}, \qquad \alpha = 1, \quad q = 1, \tag{4.17}$$

$$a(f, g;) := \sum_{K \in \mathcal{T}_h} \int_K |f - g|^2 \, \mathrm{d}\boldsymbol{x}, \qquad \alpha = 0, \quad q = 2, \tag{4.18}$$

$$a(\gamma, \boldsymbol{\sigma}) := \sum_{e \in \mathcal{E}_h^\circ} \int_e \gamma \, [\![\boldsymbol{\sigma}]\!] \, \mathrm{d}S + \sum_{e \in \mathcal{E}_h^\partial} \int_e \gamma \, \boldsymbol{\sigma} \cdot \boldsymbol{n} \, \mathrm{d}S, \qquad \alpha = 2, \quad q = 0. \tag{4.19}$$

In general, a finite element form will consist of integrals over various geometric domains: integration over cells $\mathcal{T}_h$, interior facets $\mathcal{E}_h^\circ$, and exterior facets $\mathcal{E}_h^\partial$. Therefore, we express a general multi-linear form in terms of integrals over each set of geometric entities:

$$a(\boldsymbol{c}; \boldsymbol{v}) = \sum_{K \in \mathcal{T}_h} \int_K \mathcal{I}_K^{\mathcal{T}}(\boldsymbol{c}; \boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} + \sum_{e \in \mathcal{E}_h^\circ} \int_e \mathcal{I}_e^{\mathcal{E},\circ}(\boldsymbol{c}; \boldsymbol{v}) \, \mathrm{d}S + \sum_{e \in \mathcal{E}_h^\partial} \int_e \mathcal{I}_e^{\mathcal{E},\partial}(\boldsymbol{c}; \boldsymbol{v}) \, \mathrm{d}S, \tag{4.20}$$

where $\mathcal{I}_K^{\mathcal{T}}$ denotes a cell integrand on $K \in \mathcal{T}_h$, $\mathcal{I}_e^{\mathcal{E},\circ}$ is an integrand on the interior facet $e \in \mathcal{E}_h^\circ$, and $\mathcal{I}_e^{\mathcal{E},\partial}$ is an integrand defined on the exterior facet $e \in \mathcal{E}_h^\partial$. The form

$a(\boldsymbol{c};\boldsymbol{v})$ describes a finite element form *globally* over the entire problem domain.

Here, we will consider the case where the interior facet integrands $\mathcal{I}_e^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})$ can be decomposed into two independent parts on each interior facet e: one for the positive restriction $(+)$ and the negative restriction $(-)$. That is, for each $e \in \mathcal{E}_h^\circ$, we may write: $\mathcal{I}_e^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v}) = \mathcal{I}_{e^+}^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v}) + \mathcal{I}_{e^-}^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})$. This allows us to express the integral over an interior facet e connecting two adjacent elements, say K^+ and K^-, as the sum of integrals:

$$\int_{e\subset\partial K^+\cup\partial K^-} \mathcal{I}_e^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}S = \int_{e\subset\partial K^+} \mathcal{I}_{e^+}^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}S + \int_{e\subset\partial K^-} \mathcal{I}_{e^-}^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}S. \tag{4.21}$$

Then the local contribution of (4.20) in each cell K, along with its associated facets $e \subset \partial K$, is simply

$$a_K(\boldsymbol{c};\boldsymbol{v}) = \int_K \mathcal{I}_K^{\mathcal{T}}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}\boldsymbol{x} + \sum_{e\subset\partial K\setminus\partial\Omega}\int_e \mathcal{I}_e^{\mathcal{E},\circ}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}S + \sum_{e\subset\partial K\cap\partial\Omega}\int_e \mathcal{I}_e^{\mathcal{E},\partial}(\boldsymbol{c};\boldsymbol{v})\,\mathrm{d}S. \tag{4.22}$$

We call (4.22) the *cell-local* contribution of $a(\boldsymbol{c};\boldsymbol{v})$, with

$$a(\boldsymbol{c};\boldsymbol{v}) = \sum_{K\in\mathcal{T}_h} a_K(\boldsymbol{c};\boldsymbol{v}). \tag{4.23}$$

To make matters concrete, let us suppose $a(\boldsymbol{c};\boldsymbol{v})$ is a bilinear form with arguments $\boldsymbol{v} = (v_0, v_1) \in V_0 \times V_1$. Now let $\{\Phi_i\}_{i=1}^N$ and $\{\Psi_i\}_{i=1}^M$ denote bases for V_0 and V_1 respectively. Then the global $N \times M$ matrix $\boldsymbol{A}$ corresponding to $a(\boldsymbol{c}; v_0, v_1)$ has its entries defined via

$$(\boldsymbol{A})_{ij} = a\left(\boldsymbol{c};\Phi_i,\Psi_j\right) = \sum_{K\in\mathcal{T}_h}(\boldsymbol{A}_K)_{ij}, \quad (\boldsymbol{A}_K)_{ij} = a_K\left(\boldsymbol{c};\Phi_i,\Psi_j\right). \tag{4.24}$$

By construction, $(\boldsymbol{A}_K)_{ij} \neq 0$ if and only if Φ_i and Ψ_j take non-zero values in K. Now we introduce the *cell-node map* $i = e(K,\hat{\imath})$ as the mapping from the local node number $\hat{\imath}$ in K to the global node number i. Suppose there are n and m nodes defining the degrees of freedom for V_0 and V_1, respectively, in K. Then all non-zero entries of $\boldsymbol{A}_K$ arise from integrals involving basis functions with local indices corresponding to the global indices i, j:

$$\left(\boldsymbol{A}^K\right)_{\hat{\imath}\hat{\jmath}} := a_K\left(\boldsymbol{c};\Phi_{e(K,\hat{\imath})},\Psi_{e(K,\hat{\jmath})}\right), \quad \hat{\imath}\in\{1,\cdots,n\}, \quad \hat{\jmath}\in\{1,\cdots,m\}. \tag{4.25}$$

These local contributions are collected in the $n \times m$ dense matrix $\boldsymbol{A}^K$, which we call the *element tensor*. Then the global matrix $\boldsymbol{A}$ is directly assembled from the collection of element tensors:

$$(\boldsymbol{A})_{ij} \xleftarrow{e(K,\cdot)} \left(\boldsymbol{A}^K\right)_{\hat{\imath}\hat{\jmath}}, \quad \forall K \in \mathcal{T}_h. \tag{4.26}$$

For details on the general evaluation of finite element basis functions and multilinear forms, we refer the reader to Kirby (2004), Kirby and Logg (2006), Logg et al. (2012), and Homolya et al. (2018). Further details on the global assembly of finite element operators, with a particular focus on code-generation, are summarized in the work of Logg and Wells (2010) and Markall et al. (2012).

In Firedrake, the element tensor is mapped entry-wise into a global sparse array using the cell-node map $e(K,\cdot)$. This operation is handled by PyOP2 (Rathgeber et

al., 2012) and serves as the main user-facing abstraction for global finite element assembly. However, to perform static condensation, we need to produce a new global operator by algebraically manipulating different element tensors. This is relatively invasive in numerical code, as it requires bypassing direct operator assembly in order to produce the new tensor. This is precisely the scope of Slate.

Like UFL, Slate relies on the grammar of the host-language: Python. The entire Slate language is implemented as a Python module which defines its types (classes) and operations on said types. Together, this forms a high-level language for expressing dense linear algebra on element tensors. The Slate language consists of two primary abstractions for linear algebra:

1. terminal element tensors corresponding to multi-linear integral forms (matrices, vectors, and scalars), or assembled data (for example, coefficient vectors of a finite element function); and
2. expressions consisting of algebraic operations on terminal tensors.

The composition of binary and unary operations on terminal tensors produces a *Slate expression*. Such expressions can be composed with other Slate objects in arbitrary ways, resulting in concise representations of complex algebraic operations on locally assembled arrays. We summarize all currently supported Slate abstractions here.

Terminal tensors

In Slate, one associates a tensor with data on a cell either by using a multi-linear form, or assembled coefficient data:

- `Tensor`$(a(\boldsymbol{c};\boldsymbol{v}))$
  associates a form, expressed in UFL, with its local element tensor:

$$\boldsymbol{A}^K \leftarrow a_K(\boldsymbol{c};\boldsymbol{v}), \text{ for all } K \in \mathcal{T}_h. \tag{4.27}$$

  The form arity α of $a_K(\boldsymbol{c};\boldsymbol{v})$ determines the *rank* of the corresponding `Tensor`, i.e. scalars, vectors, and matrices are produced from scalars, linear forms, and bilinear forms respectively.[3] The *shape* of the element tensor is determined by both the number of arguments, and total number of degrees of freedom local to the cell.

- `AssembledVector`(f)
  where f is some finite element function. The function $f \in V$ is expressed in terms of the finite element basis of V: $f(x) = \sum_{i=1}^{N} f_i \Phi_i(x)$. The result is the local coefficient vector of f on K:

$$\boldsymbol{F}^K_{\hat{i}} = \left\{ f_{e(K,\hat{i})} \right\}_{\hat{i}=1}^{n}, \tag{4.28}$$

  where $e(K,\hat{i})$ is the local node numbering and n is the number of nodes local to the cell K.

[3] Similarly to UFL, Slate is capable of abstractly representing arbitrary rank tensors. However, only rank ≤ 2 tensors are typically used in most finite element applications and therefore we currently only generate code for those ranks.

Symbolic linear algebra

Slate supports typical binary and unary operations in linear algebra, with a high-level syntax close to mathematics. At the time of this work, these include:

- `A + B`, the addition of two equal shaped tensors: $\boldsymbol{A}^K + \boldsymbol{B}^K$.
- `A * B`, a contraction over the last index of `A` and the first index of `B`. This is the usual multiplicative operation on matrices, vectors, and scalars: $\boldsymbol{A}^K \boldsymbol{B}^K$.
- `-A`, the additive inverse (negation) of a tensor: $-\boldsymbol{A}^K$.
- `A.T`, the transpose of a tensor: $\left(\boldsymbol{A}^K\right)^T$.
- `A.inv`, the inverse of a square tensor: $\left(\boldsymbol{A}^K\right)^{-1}$.
- `A.solve(B, decomposition="...")`, the result, $\boldsymbol{X}^K$, of solving a local linear system $\boldsymbol{A}^K \boldsymbol{X}^K = \boldsymbol{B}^K$, optionally specifying a factorization strategy.
- `A.blocks[indices]`, where `A` is a tensor from a mixed finite element space. This allows for the extraction of subblocks, which are indexed by field (slices are allowed). For example, if a matrix $\boldsymbol{A}$ corresponds to the bilinear form $a : V \times W \to \mathbb{R}$, where $V = V_0 \times \cdots \times V_n$ and $W = W_0 \times \cdots \times W_m$ are product spaces consisting of finite element spaces $\{V_i\}_{i=0}^n$, $\{W_i\}_{i=0}^m$, then the element tensors have the form:

$$\boldsymbol{A}^K = \begin{bmatrix} \boldsymbol{A}^K_{00} & \boldsymbol{A}^K_{01} & \cdots & \boldsymbol{A}^K_{0m} \\ \boldsymbol{A}^K_{10} & \boldsymbol{A}^K_{11} & \cdots & \boldsymbol{A}^K_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^K_{n0} & \boldsymbol{A}^K_{n1} & \cdots & \boldsymbol{A}^K_{nm} \end{bmatrix}. \tag{4.29}$$

  The submatrix of (4.29) with block indices $\boldsymbol{i} = (\boldsymbol{p}, \boldsymbol{q})$, $\boldsymbol{p} = \{p_1, \cdots, p_r\}$, $\boldsymbol{q} = \{q_1, \cdots, q_c\}$, is

$$\boldsymbol{A}^K_{\boldsymbol{pq}} = \begin{bmatrix} \boldsymbol{A}^K_{p_1 q_1} & \cdots & \boldsymbol{A}^K_{p_1 q_c} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}^K_{p_r q_1} & \cdots & \boldsymbol{A}^K_{p_r q_c} \end{bmatrix} = \boldsymbol{A}^K\texttt{.blocks[}\boldsymbol{p}, \boldsymbol{q}\texttt{]}, \tag{4.30}$$

  where $\boldsymbol{p} \subseteq \{0, \cdots, n\}$, $\boldsymbol{q} \subseteq \{0, \cdots, m\}$.

Each `Tensor` instantiated knowns all the information about the underlying UFL form that defines it, such as form arguments, coefficients, and the underlying finite element space(s) it operates on. This information is propagated through as unary or binary transformations are applied. All unary and binary operations shown here provides the necessary algebraic framework for a large class of problems, some of which we present in this paper.

In Firedrake, Slate expressions are transformed into low-level parallelizable code by a *linear algebra compiler*. The compiler interprets Slate expressions as a *syntax tree*, where the tree is parsed to identify what local arrays need to be assembled and the sequence of array operations. At the time of this work, our compiler generates C++ code, using the templated library Eigen (Guennebaud and Jacob, 2010) for dense linear algebra. The translation from Slate to C++ is fairly straightforward, as all operations supported by Slate have a one-to-one representation in Eigen.

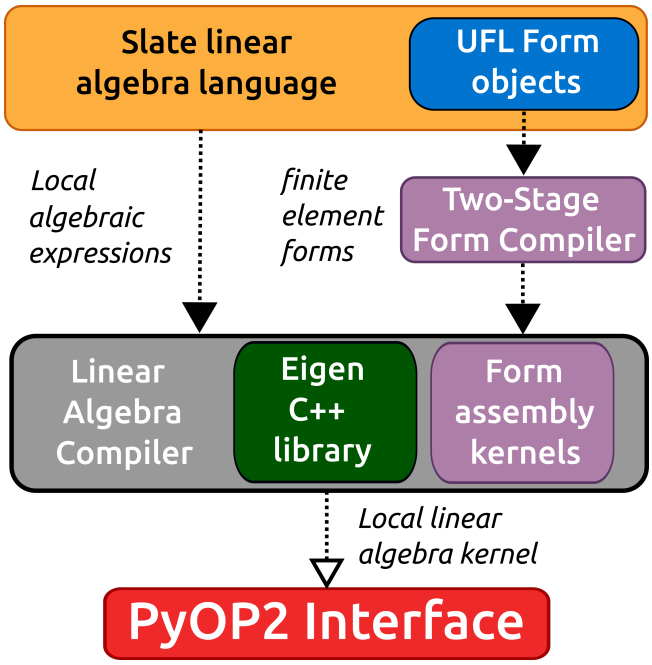


FIGURE 4.2: The Slate language wraps UFL objects describing the finite element system. The resulting Slate expressions are passed to a specialized linear algebra compiler, which produces a single "macro" kernel assembling the local contributions and executes the dense linear algebra represented in Slate. The kernels are passed to the Firedrake's PyOP2 interface, which wraps the Slate kernel in a mesh-iteration kernel. Parallel scheduling, code generation, and compilation occurs after the PyOP2 layer.

The compiler pass will generate a single "macro" kernel, which performs the dense linear algebra operations represented in Slate. The resulting code will also include (often multiple) function calls to local assembly kernels generated by TSFC (Homolya et al., 2018) in order to assemble all necessary sub-blocks of an element tensor. All code generated by the linear algebra compiler conforms to the application programming interface (API) of the PyOP2 framework, as detailed by Rathgeber et al. (2012, §3). Figure 4.2 provides an illustration of the complete tool-chain.

4.3 Examples

We now present a few examples and discuss solution methods which require element-wise manipulations of finite element systems and their specification in Slate. We stress here that Slate is not limited to these model problems; rather these examples were chosen for clarity and to demonstrate key features of the Slate language. For our discussion, we use a model elliptic equation defined in Ω, the computational domain.

Consider the second-order elliptic PDE with Dirichlet and Neumann boundary conditions:

$$-\nabla\cdot(\kappa\nabla p)+cp=f \text{ in } \Omega,\quad p=p_0 \text{ on } \partial\Omega_D,\quad -\kappa\nabla p\cdot\boldsymbol{n}=g \text{ on } \partial\Omega_N, \tag{4.31}$$

where $\partial\Omega_D\cup\partial\Omega_N=\partial\Omega$ and $\kappa, c:\Omega\to\mathbb{R}^+$ are positive-valued coefficients. To obtain a mixed formulation of (4.31), we introduce the auxiliary velocity variable

$\boldsymbol{u} = -\kappa \nabla p$. Setting $\mu = \kappa^{-1}$, we then obtain the first-order system of PDEs:

$$\mu \boldsymbol{u} + \nabla p = 0 \quad \text{in } \Omega, \tag{4.32}$$
$$\nabla \cdot \boldsymbol{u} + cp = f \quad \text{in } \Omega, \tag{4.33}$$
$$p = p_0 \quad \text{on } \partial\Omega_D, \tag{4.34}$$
$$\boldsymbol{u} \cdot \boldsymbol{n} = g \quad \text{on } \partial\Omega_N. \tag{4.35}$$

4.3.1 Hybridization of mixed methods

Following the model Poisson equation from Section 4.1.1, the mixed method for (4.32)–(4.35) follows identically. Methods of this type seek approximations $(\boldsymbol{u}_h, p_h)$ in the finite-dimensional subspaces $U_h \times V_h \subset H(\text{div}; \Omega) \times L^2(\Omega)$, defined by:

$$U_h = \{\boldsymbol{w} \in H(\text{div}; \Omega) : \boldsymbol{w}|_K \in U(K), \forall K \in \mathcal{T}_h, \boldsymbol{w} \cdot \boldsymbol{n} = g \text{ on } \partial\Omega_N\}, \tag{4.36}$$
$$V_h = \{\phi \in L^2(\Omega) : \phi|_K \in V(K), \forall K \in \mathcal{T}_h\}, \tag{4.37}$$

where the U_h and V_h are the usual $H(\text{div}) \times L^2$-pairing of finite element spaces listed in Sections 2.4.2 and 2.4.4. The resulting mixed finite element problem reads as follows: find $(\boldsymbol{u}_h, p_h) \in U_h \times V_h$ satisfying

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \mu \boldsymbol{u}_h \, \mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} p_h \nabla_h \cdot \boldsymbol{w} \, \mathrm{d}\boldsymbol{x} = -\int_{\mathcal{E}^{\partial}_{h,D}} p_0 \boldsymbol{w} \cdot \boldsymbol{n} \, \mathrm{d}S, \quad \forall \boldsymbol{w} \in U_{h,0}, \tag{4.38}$$
$$\int_{\mathcal{T}_h} \phi \nabla_h \cdot \boldsymbol{u}_h \, \mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \phi \, c p_h \, \mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \phi \, f \, \mathrm{d}\boldsymbol{x}, \quad \forall \phi \in V_h, \tag{4.39}$$

where $U_{h,0}$ is the subspace of U_h with functions whose normal components vanish on $\partial\Omega_N$. The matrix equation is obtained in an identical fashion to (4.10) and has the form:

$$\begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{D} \end{bmatrix} \begin{Bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F}_0 \\ \boldsymbol{F}_1 \end{Bmatrix}. \tag{4.40}$$

As we have stated at in Section 4.1.1, inverting (4.40) using a preconditioner based on a Schur-complement factorization requires two global inversions: one for $\boldsymbol{A}$ and another for the elliptic[4] Schur-complement $\boldsymbol{S} = \boldsymbol{D} + \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$. Inverting $\boldsymbol{A}$ is typically not a problem; the operator is symmetric, positive-definite, and sparse. However, forming $\boldsymbol{S}$ explicitly is impractical due to the dense inverse $\boldsymbol{A}^{-1}$.

In this case, one typically uses a *sparse approximation* of $\boldsymbol{S}$, say $\widetilde{\boldsymbol{S}}$. A common approach in fluid dynamics is to use a diagonal approximation of $\boldsymbol{A}$ when forming $\widetilde{\boldsymbol{S}}$ (for example, see Benzi, Golub, and Liesen (2005, §10.1.3)):

$$\widetilde{\boldsymbol{S}} = \boldsymbol{D} + \boldsymbol{B}\widetilde{\boldsymbol{A}}^{-1}\boldsymbol{B}^T, \quad \widetilde{\boldsymbol{A}} = \text{Diag}(\boldsymbol{A}). \tag{4.41}$$

Since $\widetilde{\boldsymbol{A}}$ is diagonal, its inverse is trivial to compute and $\widetilde{\boldsymbol{S}}$ becomes a sparse operator. However, the resulting Schur-complement is no longer exact. Therefore, an outer Krylov method, such as GMRES, is required to control the reduction of the problem residual. The rate at which GMRES converges will then depend on how well $\widetilde{\boldsymbol{S}}$ approximates $\boldsymbol{S}$ (Siefert and Sturler, 2006).

[4] The Schur-complement is often called an elliptic operator since $\boldsymbol{S}$ here can be interpreted as the discretization of the second-order operator: $cp - \nabla \cdot (\kappa \nabla p)$ (Benzi, Golub, and Liesen, 2005, §10.1.3).

As discussed in Chapter 3, the hybridization technique avoids handling globally dense operators by replacing the original system (4.38)–(4.39) with a discontinuous variant. This is done by replacing the discrete solution space for $\boldsymbol{u}_h$ with the "broken" space $\widehat{U}_h$. The approximation space for p_h remains unchanged. Lagrange multipliers are then introduced as an auxiliary variable in the space M_h, defined only on cell-interfaces:

$$M_h = \{\gamma \in L^2(\mathcal{E}_h) : \gamma|_e \in M(e), \forall e \in \mathcal{E}_h\}. \tag{4.42}$$

The space $M(e)$ is chosen to be a polynomial space on e of the *same* degree as $\boldsymbol{u}_h \cdot \boldsymbol{n}|_e$. See Section 3.1.2 for a review of the function spaces for the hybridizable mixed methods.

The resulting hybridizable formulation reads: find $(\widehat{\boldsymbol{u}}_h, p_h, \lambda_h) \in \widehat{U}_h \times V_h \times M_{h,0}$ such that

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \mu \widehat{\boldsymbol{u}}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} p_h \nabla_h \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h \setminus \partial\Omega_D} \lambda_h [\![\boldsymbol{w}]\!] \,\mathrm{d}S = -\int_{\mathcal{E}^\partial_{h,D}} p_0 \boldsymbol{w} \cdot \boldsymbol{n} \,\mathrm{d}S, \tag{4.43}$$

$$\int_{\mathcal{T}_h} \phi \nabla_h \cdot \widehat{\boldsymbol{u}}_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \phi \, c p_h \,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{T}_h} \phi \, f \,\mathrm{d}\boldsymbol{x}, \tag{4.44}$$

$$\int_{\mathcal{E}_h \setminus \partial\Omega_D} \gamma [\![\widehat{\boldsymbol{u}}_h]\!] \,\mathrm{d}S = \int_{\mathcal{E}^\partial_{h,N}} \gamma \, g \,\mathrm{d}S, \tag{4.45}$$

for all $(\boldsymbol{w}, \phi, \gamma) \in \widehat{U}_h \times V_h \times M_{h,0}$, where $M_{h,0}$ denotes the subspace of traces vanishing on $\partial\Omega_D$. The jump condition in (4.45) enforces both continuity of the normal components of $\widehat{\boldsymbol{u}}_h$ across elemental boundaries, as well as the boundary condition on $\partial\Omega_N$. As discussed in Section 3.1.2, the "broken" velocity $\widehat{\boldsymbol{u}}_h$ will coincide with its $H(\mathrm{div})$ counterpart $\boldsymbol{u}_h \in U_h$; the formulations (4.43)–(4.44) and (4.38)–(4.39) are solving equivalent problems (Arnold and Brezzi, 1985).

Notwithstanding a minor break in convention from Section 3.2, we reintroduce our previous notation for the broken velocity $\widehat{\boldsymbol{u}}_h$. This is due to the fact that the discrete coefficient vectors for $\widehat{\boldsymbol{u}}_h$ and $\boldsymbol{u}_h$ in a computer implementation are of different sizes. We therefore distinguish between the two fields to illustrate this fact in our discussion throughout this chapter.

With $\widehat{\boldsymbol{U}}$, $\boldsymbol{P}$, and $\boldsymbol{\Lambda}$ denoting the coefficient vectors for $(\widehat{\boldsymbol{u}}_h, p_h, \lambda_h)$, the matrix system arising from (4.43)–(4.45) has the general form:

$$\begin{bmatrix} \boldsymbol{A}_{00} & \boldsymbol{A}_{01} & \boldsymbol{A}_{02} \\ \boldsymbol{A}_{10} & \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{20} & \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \begin{Bmatrix} \widehat{\boldsymbol{U}} \\ \boldsymbol{P} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{F}_0 \\ \boldsymbol{F}_1 \\ \boldsymbol{F}_2 \end{Bmatrix}, \tag{4.46}$$

where the matrix system is produced by expanding test and trial functions in terms of the finite element bases for $\widehat{U}_h$, V_h, and M_h. Despite the *dramatic* increase in total number of unknowns to determine, (4.46) has a considerable advantage over (4.40) in the following ways:

1. Since both $\widehat{U}_h$ and V_h are discontinuous spaces, $\widehat{\boldsymbol{u}}_h$ and p_h are coupled only within the cell. This allows us to simultaneously eliminate both $\widehat{\boldsymbol{U}}$ and $\boldsymbol{P}$ via *element-wise* static condensation. This produces a significantly smaller global problem for $\boldsymbol{\Lambda}$ (the hybridized problem):

$$\boldsymbol{S}_\lambda \boldsymbol{\Lambda} = \boldsymbol{E}, \tag{4.47}$$

where $\boldsymbol{S}_\lambda$ and $\boldsymbol{E}$ are assembled element-wise by computing the local element tensors $\{\boldsymbol{S}_\lambda^K\}_{K\in\mathcal{T}_h}$ and $\{\boldsymbol{E}^K\}_{K\in\mathcal{T}_h}$ respectively:

$$\boldsymbol{S}_\lambda^K = \boldsymbol{A}_{22}^K - \begin{bmatrix}\boldsymbol{A}_{20}^K & \boldsymbol{A}_{21}^K\end{bmatrix}\begin{bmatrix}\boldsymbol{A}_{00}^K & \boldsymbol{A}_{01}^K\\ \boldsymbol{A}_{10}^K & \boldsymbol{A}_{11}^K\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{A}_{02}^K\\ \boldsymbol{A}_{12}^K\end{bmatrix}, \tag{4.48}$$

$$\boldsymbol{E}^K = \boldsymbol{F}_2^K - \begin{bmatrix}\boldsymbol{A}_{20}^K & \boldsymbol{A}_{21}^K\end{bmatrix}\begin{bmatrix}\boldsymbol{A}_{00}^K & \boldsymbol{A}_{01}^K\\ \boldsymbol{A}_{10}^K & \boldsymbol{A}_{11}^K\end{bmatrix}^{-1}\begin{Bmatrix}\boldsymbol{F}_0^K\\ \boldsymbol{F}_1^K\end{Bmatrix}. \tag{4.49}$$

2. The matrix $\boldsymbol{S}_\lambda$ is sparse, symmetric, positive-definite, and spectrally equivalent to the dense Schur-complement $\boldsymbol{S} = \boldsymbol{D} + \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T$ from (4.40) of the original mixed system (Gopalakrishnan, 2003; Cockburn, Gopalakrishnan, and Lazarov, 2009).

3. Once $\boldsymbol{\Lambda}$ is computed, both $\widehat{\boldsymbol{U}}$ and $\boldsymbol{P}$ can be recovered locally in each element. This can be accomplished in a number of ways. For example, one can compute $\boldsymbol{P}^K$ by solving:

$$\begin{aligned}\left(\boldsymbol{A}_{11}^K - \boldsymbol{A}_{10}^K\left(\boldsymbol{A}_{00}^K\right)^{-1}\boldsymbol{A}_{01}^K\right)\boldsymbol{P}^K = \boldsymbol{F}_1^K - \boldsymbol{A}_{10}^K\left(\boldsymbol{A}_{00}^K\right)^{-1}\boldsymbol{F}_0^K\\ - \left(\boldsymbol{A}_{12}^K - \boldsymbol{A}_{10}^K\left(\boldsymbol{A}_{00}^K\right)^{-1}\boldsymbol{A}_{02}^K\right)\boldsymbol{\Lambda}^K,\end{aligned} \tag{4.50}$$

followed by solving for $\widehat{\boldsymbol{U}}^K$:

$$\boldsymbol{A}_{00}^K\widehat{\boldsymbol{U}}^K = \boldsymbol{F}_0^K - \boldsymbol{A}_{01}^K\boldsymbol{P}^K - \boldsymbol{A}_{02}^K\boldsymbol{\Lambda}^K. \tag{4.51}$$

Similarly, one could rearrange the order in which each variable is reconstructed.

4. Arnold and Brezzi (1985) showed that λ_h is actually an approximation to p on the mesh skeleton. They further demonstrated that λ_h can be used to locally post-process p_h, yielding a new approximation $p_h^\star$ to p with superconvergent properties. Local post-processing was studied further by Brezzi, Douglas, and Marini (1985), Bramble and Xu (1989), Stenberg (1991), and Cockburn et al. (2010). We highlight two post-processing methods for $\widehat{\boldsymbol{u}}_h$ and p_h, respectively, in Section 4.3.3.

The hybridization of (4.38)–(4.39) can be interpreted as a *static-condensation-amenable* mixed method. As noted in early analysis on the cost of static condensation for hybridizable mixed methods (Marini, 1985; Arbogast and Chen, 1995), the cost of local static condensation is small compared to inverting (4.47). This is also reflected in our numerical experiments in Section 4.5.2. Therefore, the efficiency of hybridizable mixed methods relies on fast solution techniques for inverting $\boldsymbol{S}_\lambda$.

Gopalakrishnan (2003) showed that $\boldsymbol{S}_\lambda$ is spectrally equivalent to an elliptic operator. As a result, this has galvanized investigations on applying algebraic multigrid (AMG) methods directly on the condensed operator (Brunner and Kolev, 2011; Kalchev et al., 2016; Dobrev et al., 2019). This also inspired our choice of solver algorithms for hybridized systems resulting in a nonsymmetric $\boldsymbol{S}_\lambda$ due to the implicit-treatment of the Coriolis term in geophysical models.[5] We shall elaborate further on AMG strategies in Section 4.4, including potential avenues for future work.

[5] See, for example, the hybridizable mixed methods for the linear shallow water equations in Section 3.2 and the gravity wave system in Section 3.4.

LISTING 4.2: Abridged Firedrake code for solving (4.46) via static condensation and local recovery. Arguments of the mixed space $\widehat{U}_h \times V_h \times M_h$ are indexed by 0, 1, and 2 respectively. The `mesh`, `degree`, and function data `mu`, `c`, `f`, `g`, and `p0` are assumed to have been defined.

```
U = FunctionSpace(mesh, "DRT", degree)        # Broken RT space
V = FunctionSpace(mesh, "DG", degree - 1)     # DG space
M = FunctionSpace(mesh, "DGT", degree - 1)    # Trace space
W = U * V * M
[...]  # User-defined data: μ,c,f,g,p0

u, p, lmbd = TrialFunctions(W)
w, phi, gmma = TestFunctions(W)
n = FacetNormal(mesh)

# Bilinear and linear forms for (4.43)-(4.45)
a = (dot(w,mu*u)*dx - div(w)*p*dx + lmbd('+')*jump(w,n=n)*dS
     + lmbd*dot(w,n)*ds(neumann_ids)
     + phi*div(u)*dx + phi*c*p*dx
     + gmma('+')*jump(u,n=n)*dS + gmma*dot(u,n)*ds(neumann_ids))
L = (dot(w,n)*p0*ds(dirichlet_ids) + phi*f*dx
     + gmma*g*ds(neumann_ids))

# Blocks of element tensors defining the local 3-by-3 system
A = Tensor(a).blocks
F = Tensor(L).blocks

# Slate expressions for (4.48) and (4.49)
Sexp = A[2, 2] - A[2, :2] * A[:2, :2].inv * A[:2, 2]
Eexp = F[2] - A[2, :2] * A[:2, :2].inv * F[:2]

S = assemble(Sexp, bcs=[...])          # Evaluate (4.48)
E = assemble(Eexp)                     # Evaluate (4.49)
lambda_h = Function(M)                 # Function for λ_h

# Solve (4.47)
solve(S, lambda_h, E,
      solver_parameters={"ksp_type":"preonly", "pc_type":"lu"})

p_h = Function(V)                      # Function for p_h
u_h = Function(U)                      # Function for û_h
Lambda = AssembledVector(lambda_h)     # Coefficient vector for λ_h
P = AssembledVector(p_h)               # Coefficient vector for p_h

# Intermediate expressions
Sd = A[1, 1] - A[1, 0] * A[0, 0].inv * A[0, 1]
Sl = A[1, 2] - A[1, 0] * A[0, 0].inv * A[0, 2]

# Slate expressions for (4.50) and (4.51)
p_expr = Sd.solve(F[1] - A[1, 0] * A[0, 0].inv * F[0]
                  - Sl * Lambda, decomposition="PartialPivLu")
u_expr = A[0, 0].solve(F[0] - A[0, 1] * P - A[0, 2] * Lambda,
                       decomposition="PartialPivLu")

assemble(p_expr, p_h)                  # Evaluate (4.50)
assemble(u_expr, u_h)                  # Evaluate (4.51)
```

Listing 4.2 demonstrates how Slate can be used directly by a Firedrake user. Lines 1–3 define the spaces for the hybridizable RT method on a given mesh (whose definition is omitted for brevity). Test and trial functions for the finite element discretization are defined in the usual way. Lines 12 and 16 demonstrate the capability of UFL for expressing far more than just cell integrals (`dx`). In particular, surface terms now appear on the interior facets (`dS`) due to the jump terms.[6]

The exterior facet terms (`ds`) appear on regions of the mesh boundary for the Dirichlet and Neumann conditions. Every built-in Firedrake mesh contains a unique set of boundary markers (typically a list of integers) denoting specific regions along the boundary. Exterior facet integrals along the boundary containing Dirichlet data are denoted by providing the exterior facet measure with a list of boundary markers: `ds(dirichlet_ids)` (similarly for `ds(neumann_ids)`). If a mesh is provided from a third-party library, the boundary markers must be specified in the corresponding mesh-input file.[7]

Lines 20–51 contain Slate-specific Firedrake code. Symbolic expressions for (4.48) and (4.49) are shown in lines 24 and 25 respectively. Similarly with the Firedrake demonstration in Section 4.1.1, code-generation only occurs after attempting to numerically evaluate the expression (in this case, calling "`assemble`" in lines 27–28). Any vanishing conditions for the trace variables can be provided at the point of assembly. A global linear solve for the Lagrange multiplier is performed in line 32. Programming the linear solver is done by providing an appropriate Python dictionary of PETSc options.

Lines 45 and 47 are symbolic expressions for (4.50) and (4.51). In this case, we have provided the argument "`decomposition`" which instructs the linear algebra compiler (illustrated in Figure 4.2) to insert an external function call to Eigen for performing an LU decomposition. Inverses in lines 24 and 25 could also be replaced with similar solve calls, allowing the user to specify directly how matrix inverses should be computed (by default, "`A.inverse()`" performs an LU factorization).

Remark 9. *The purpose of Listing 4.2 is to demonstrate the high-level syntax Slate provides; it is not necessarily the most convenient way to solve a mixed method using hybridization. In fact, Section 4.4 illustrates a far more compact and composable way to statically condense finite element systems in the form of a PETSc "`PC`" object.*

4.3.2 Hybridization of discontinuous Galerkin methods

The hybridizable discontinuous Galerkin (HDG) method is a natural extension of discontinuous Galerkin (DG) discretizations. Similarly with hybridizable mixed methods, and HDG method is a static-condensation-amenable DG method with *numerical fluxes* expressed in terms of the numerical trace of the scalar variable. Here, we consider a specific HDG discretization, namely the LDG-H method (Cockburn,

[6] For syntactical reasons, UFL requires all arguments defined on interior facets to have a facet restriction (`'+'`) or (`'-'`) (denoting two sides of any given facet). This is handled implicitly by the UFL operator: `jump`. However, the trace functions must also be restricted. They have arbitrarily been given the (`'+'`) restriction.

[7] A public demonstration on providing a Firedrake-compatible mesh using Gmsh (Geuzaine and Remacle, 2009) can be found here: https://www.firedrakeproject.org/demos/immersed_fem.py.html (Date accessed: November 9, 2019).

Gopalakrishnan, and Sayas, 2010). Other forms of HDG that involve local lifting operators can also be implemented in this software framework by the introduction of additional local (i.e., discontinuous) variables in the definition of the local solvers.

Deriving the LDG-H formulation follows exactly from standard DG methods. All prognostic variables are sought in the discontinuous spaces $U_h \times V_h \subset [L^2(\Omega)]^n \times L^2(\Omega)$, defined as:

$$U_h = \{\boldsymbol{w} \in [L^2(\Omega)]^n : \boldsymbol{w}|_K \in U(K), \forall K \in \mathcal{T}_h\}, \tag{4.52}$$

$$V_h = \{\phi \in L^2(\Omega) : \phi|_K \in V(K), \forall K \in \mathcal{T}_h\}, \tag{4.53}$$

where $U(K)$ and $V(K)$ are vector and scalar polynomial spaces, respectively, of degree $k \geq 0$. Then the DG method can be formulated as the problem of finding $\boldsymbol{u}_h \in U_h$ and $p_h \in V_h$ such that for all $K \in \mathcal{T}_h$:

$$\int_K \boldsymbol{w} \cdot \mu \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} - \int_K p_h \nabla \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x} + \int_{\partial K} \widehat{p}\boldsymbol{w} \cdot \boldsymbol{n} \,\mathrm{d}S = 0, \tag{4.54}$$

$$-\int_K \nabla\phi \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} + \int_{\partial K} \phi\widehat{\boldsymbol{u}} \cdot \boldsymbol{n} \,\mathrm{d}S + \int_K \phi\, c p_h \,\mathrm{d}\boldsymbol{x} = \int_K \phi\, f \,\mathrm{d}\boldsymbol{x}, \tag{4.55}$$

for all $\boldsymbol{w} \in U(K)$ and $\phi \in V(K)$, where $\widehat{p}$ and $\widehat{\boldsymbol{u}}$ are the scalar and vector numerical fluxes approximating p and $\boldsymbol{u} = -\kappa\nabla p$, respectively, on the boundary of K.

To construct an HDG discretization, we define the numerical fluxes to be functions of the trial unknowns and a new independent unknown in the trace space M_h, the space of trace functions whose restriction to a facet e is a polynomial of degree at most k. For the LDG-H method, the fluxes take the form:

$$\widehat{\boldsymbol{u}}(\boldsymbol{u}_h, p_h, \lambda_h; \tau) = \boldsymbol{u}_h + \tau\,(p_h - \widehat{p})\,\boldsymbol{n}, \tag{4.56}$$

$$\widehat{p}(\lambda_h) = \lambda_h, \tag{4.57}$$

where $\lambda_h \in M_h$ is a function approximating the trace of p on $\mathcal{E}_h$ and τ is a positive stablization function that may vary on each facet $e \in \mathcal{E}_h$. Following Cockburn, Gopalakrishnan, and Sayas (2010), we enforce λ_h to satisfy the Dirichlet condition on $\partial\Omega_D$ in an L^2-projection sense.

The full LDG-H formulation now reads as follows. Find $(\boldsymbol{u}_h, p_h, \lambda_h) \in U_h \times V_h \times M_h$ such that

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \mu \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} p_h \nabla_h \cdot \boldsymbol{w} \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \lambda_h [\![\boldsymbol{w}]\!] \,\mathrm{d}S = 0, \tag{4.58}$$

$$-\int_{\mathcal{T}_h} \nabla_h\phi \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \phi\, c p_h \,\mathrm{d}\boldsymbol{x}$$

$$+\int_{\mathcal{E}_h} \phi\, [\![\boldsymbol{u}_h + \tau\,(p_h - \lambda_h)\,\boldsymbol{n}]\!] \,\mathrm{d}S = \int_{\mathcal{T}_h} \phi\, f \,\mathrm{d}\boldsymbol{x}, \tag{4.59}$$

$$\int_{\mathcal{E}_h \setminus \partial\Omega_D} \gamma\, [\![\boldsymbol{u}_h + \tau\,(p_h - \lambda_h)\,\boldsymbol{n}]\!] \,\mathrm{d}S = \int_{\mathcal{E}^{\partial}_{h,N}} \gamma\, g \,\mathrm{d}S, \tag{4.60}$$

$$\int_{\mathcal{E}^{\partial}_{h,D}} \gamma\, \lambda_h \,\mathrm{d}S = \int_{\mathcal{E}^{\partial}_{h,D}} \gamma\, p_0 \,\mathrm{d}S, \tag{4.61}$$

for all $(\boldsymbol{w}, \phi, \gamma) \in U_h \times V_h \times M_h$. Equation (4.60) is the jump condition, which enforces the continuity of $\widehat{\boldsymbol{u}} \cdot \boldsymbol{n}$ on $\mathcal{E}_h$ and (4.61) ensures λ_h satisfies the Dirichlet condition. By virtue of the jump condition, the numerical flux is single-valued on the

facets. Hence, the LDG-H method defines a *conservative* DG method (Cockburn, Gopalakrishnan, and Sayas, 2010). The discretization is well-posed and stable for all $\tau > 0$. The stability parameter τ does, however, have a significant influence on the rates of convergence in the computed solutions. This is especially apparent during local post-processing.

The LDG-H method retains the advantages of standard DG methods while also enabling the assembly of reduced linear systems through static condensation. The matrix system arising from (4.58)–(4.61) has the same general form as the hybridized mixed method in (4.46), except all sub-blocks are now populated with non-zero entries due to the coupling of trace functions with both p_h and $\boldsymbol{u}_h$. The Slate expressions for the local elimination and reconstruction operations will be identical to those in Listing 4.2. For the interested reader, a unified analysis of hybridization methods (both mixed and DG) for second-order elliptic equations is presented in Cockburn, Gopalakrishnan, and Lazarov (2009) and Cockburn (2016).

4.3.3 Local post-processing

For both mixed (Arnold and Brezzi, 1985; Brezzi, Douglas, and Marini, 1985; Bramble and Xu, 1989; Stenberg, 1991) and DG methods (Cockburn, Gopalakrishnan, and Sayas, 2010; Cockburn, Guzmán, and Wang, 2009), it is possible to locally post-process solutions to obtain superconvergent approximations (gaining one order of accuracy over the unprocessed solution). These methods can be expressed as local solves on each element, and so this fits well within the scope of Slate. In this section, we present two post-processing techniques: one for scalar fields, and another for the vector unknown. The Slate code follows naturally from previous discussions in Sections 4.3.1 and 4.3.2, using the standard set of operations on element tensors summarized in Section 4.2.1.

Post-processing of the scalar solution

Our first example is a modified version of the procedure presented by Stenberg (1991) for enhancing the accuracy of the scalar solution. This was also highlighted within the context of hybridizing eigenproblems by Cockburn et al. (2010). This post-processing technique can be used for both the hybridizable mixed and LDG-H methods. We proceed by posing the finite element systems cell-wise.

Let $\mathcal{P}_k(K)$ denote a polynomial space of degree $\leq k$ on a cell $K \in \mathcal{T}_h$. Then for a given pair of computed solutions $\boldsymbol{u}_h, p_h$ of the hybridizable methods, we define the post-processed scalar $p_h^\star \in \mathcal{P}_{k+1}(K)$ as the unique solution of the local problem:

$$\int_K \nabla w \cdot \nabla p_h^\star \,\mathrm{d}\boldsymbol{x} = -\int_K \nabla w \cdot \kappa^{-1} \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} \tag{4.62}$$

$$\int_K v\, p_h^\star \,\mathrm{d}\boldsymbol{x} = \int_K v\, p_h \,\mathrm{d}\boldsymbol{x}, \tag{4.63}$$

for all $w \in \mathcal{P}_{k+1}^{\perp,l}(K)$ and $v \in \mathcal{P}_l(K)$, $0 \leq l \leq k$. Here, the space $\mathcal{P}_{k+1}^{\perp,l}(K)$ denotes the L^2-orthogonal complement of $\mathcal{P}_l(K)$. This post-processing method directly uses the definition of the flux $\boldsymbol{u}_h$, the approximation of $-\kappa\nabla p$. In practice, the space $\mathcal{P}_{k+1}^{\perp,l}(K)$ may be constructed using an orthogonal hierarchical basis, and solving (4.62)–(4.63) amounts to inverting a symmetric positive definite system in each cell of the mesh.

LISTING 4.3: Example of local post-processing using Firedrake and Slate. Here, we locally solve the mixed system defined in (4.62)–(4.63). The corresponding symbolic local tensors are defined in lines 12 and 17. The Slate expression for directly inverting the local system is written in line 20. In line 24, a Slate-generated kernel is produced which solves the resulting linear system in each cell. Since we are not interested in the multiplier, we only return the block corresponding to the post-processed scalar solution.

```
1  # Define spaces for the higher-order scalar approximation
2  # and Lagrange multipliers
3  DGk1 = FunctionSpace(mesh, "DG", degree + 1)
4  DG0 = FunctionSpace(mesh, "DG", 0)
5
6  # Test and trial functions
7  W = DGk1 * DG0
8  p, psi = TrialFunctions(W)
9  w, phi = TestFunctions(W)
10
11 # Create local Slate tensors for the post-processing system
12 K = Tensor((inner(grad(p), grad(w)) + inner(psi, w)
13             + inner(p, phi))*dx)
14
15 # Use the computed pressure p_h and flux u_h
16 # in the right-hand side
17 F = Tensor((-inner(u_h, grad(w)) + inner(p_h, phi))*dx)
18
19 # Solve (4.64)-(4.65)
20 E = K.inv * F
21
22 # Function for the post-processed scalar p_h^*
23 p_star = Function(DGk1, name="Post-processed scalar")
24 assemble(E.blocks[0], p_star)       # Return only p_h^*
```

At the time of this work, Firedrake does not support the construction of such a finite element basis. However, we can introduce Lagrange multipliers to enforce the orthogonality constraint. The resulting local problem then becomes the following mixed system: find $(p_h^\star, \Psi) \in \mathcal{P}_{k+1}(K) \times \mathcal{P}_l(K)$

$$\int_K \nabla w \cdot \nabla p_h^\star \,\mathrm{d}\boldsymbol{x} + \int_K w\,\Psi \,\mathrm{d}\boldsymbol{x} = -\int_K \nabla w \cdot \kappa^{-1}\boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x}, \tag{4.64}$$

$$\int_K \phi\, p_h^\star \,\mathrm{d}\boldsymbol{x} = \int_K \phi\, p \,\mathrm{d}\boldsymbol{x}, \tag{4.65}$$

for all $w \in \mathcal{P}_{k+1}(K)$ and $\phi \in \mathcal{P}_l(K)$, $0 \leq l \leq k$. The local problems (4.64)–(4.65) and (4.62)–(4.63) are equivalent, with the Lagrange multiplier Ψ enforcing orthogonality of test functions in $\mathcal{P}_{k+1}(K)$ with functions in $\mathcal{P}_l(K)$.

This post-processing method produces a new approximation which superconverges at a rate of $k+2$ for hybridized mixed methods (Stenberg, 1991; Cockburn et al., 2010). For the LDG-H method, $k+2$ superconvergence is achieved when $\tau = \mathcal{O}(1)$ and $\tau = \mathcal{O}(h)$, but only $k+1$ convergence is achieved when $\tau = \mathcal{O}(1/h)$ (Cockburn, Guzmán, and Wang, 2009; Cockburn, Gopalakrishnan, and Sayas, 2010). We demonstrate the increased accuracy in computed solutions in Section 4.5.1. An abridged example using Firedrake and Slate to solve the local linear systems is provided in Listing 4.3.

Post-processing of the flux

Our second example illustrates a procedure that uses the numerical flux of an HDG discretization for (4.32)–(4.35). Within the context of the LDG-H method, we can use the numerical trace in (4.56) to produce a vector field that is in $H(\mathrm{div})$. Specifically, the new flux $\boldsymbol{u}_h^\star$ has continuous normal components on $\mathcal{E}_h$. The technique we outline here follows that of Cockburn, Guzmán, and Wang (2009).

Let $\mathcal{T}_h$ be a mesh consisting of simplices. On each cell $K \in \mathcal{T}_h$, we define $\boldsymbol{u}_h^\star$ to be the unique element of the local Raviart-Thomas space $[\mathcal{P}_k(K)]^n + \boldsymbol{x}\mathcal{P}_k(K)$ satisfying

$$\int_K \boldsymbol{r} \cdot \boldsymbol{u}_h^\star \,\mathrm{d}\boldsymbol{x} = \int_K \boldsymbol{r} \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x}, \tag{4.66}$$

$$\int_e \mu \boldsymbol{u}_h^\star \cdot \boldsymbol{n} \,\mathrm{d}S = \int_e \mu \widehat{\boldsymbol{u}} \cdot \boldsymbol{n} \,\mathrm{d}S, \tag{4.67}$$

for all facets e on ∂K, where $\widehat{\boldsymbol{u}}$ is the numerical flux defined in (4.56). This local problem produces a new velocity $\boldsymbol{u}_h^\star$ with the following properties:

1. $\boldsymbol{u}_h^\star$ converges at the *same* rate as $\boldsymbol{u}_h$ for all choices of τ producing a solvable system for (4.58)–(4.61). However,
2. $\boldsymbol{u}_h^\star \in H(\mathrm{div}; \Omega)$. That is, $[\![\boldsymbol{u}_h^\star]\!]_e = 0$, $\forall e \in \mathcal{E}_h^\circ$.
3. Additionally, the divergence of $\boldsymbol{u}_h^\star$ convergences at a rate of $k+1$.

The Firedrake implementation using Slate is similar to the scalar post-processing example (see Listing 4.3); the cell-wise linear systems (4.66)–(4.67) can be expressed in UFL, and therefore the necessary Slate expressions to invert the local systems follows naturally from the set of operations presented in Section 4.2.1. We use the very sensitive parameter dependency in the post-processing methods to validate our software implementation in Zenodo/Tabula-Rasa (2019, "Code-verification").

4.4 Static condensation as a preconditioner

We have already demonstrated the flexible composition of solvers and preconditioners PETSc provides in Section 4.1.1. Firedrake already relies heavily on PETSc through its Python interface: petsc4py (Dalcin et al., 2011). In addition to having access to a wide range of third-party libraries through PETSc (MUMPS (Amestoy, Duff, and L'Excellent, 2000), UMFPACK (Davis, 2004), hypre (Baker et al., 2011), and many more), Firedrake can provide *operator-based preconditioners* for solving linear systems as Python-implemented (through petsc4py) `PC` objects. These specialized preconditioners use the underlying UFL representation of the PDE in a Python context accessible to PETSc. The result is a `PC` capable of accessing the PDE-level information as needed.

Slate enables static condensation approaches to be expressed very concisely. Nonetheless, application of static condensation to different finite element systems using Slate still requires a certain amount of code repetition; solving different PDE systems may require re-deriving the necessary Slate expressions to condense the system and recover unknowns. By formulating each form of static condensation as a preconditioner, code can be written *once* and then applied to any mathematically suitable problem. Rather than writing the static condensation by hand, in many cases, it is

sufficient to just select the appropriate, Slate-based, preconditioner. Before we proceed, it will be useful to quickly explain our notation for linear solvers.

Suppose we wish to solve a linear system: $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. We can think of (left) preconditioning the system in residual form:

$$\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{A}, \boldsymbol{b}) \equiv \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \tag{4.68}$$

by an operator $\mathcal{P}$ (`PC`), which may not necessarily be linear, as a transformation into an equivalent system of the form

$$\mathcal{P}\boldsymbol{r} = \mathcal{P}\boldsymbol{b} - \mathcal{P}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}. \tag{4.69}$$

Given a current iterate $\boldsymbol{x}_i$ the residual at the i-th iteration is simply $\boldsymbol{r}_i \equiv \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$, and $\mathcal{P}$ acts on the residual to produce an approximation to the error $\boldsymbol{\epsilon}_i \equiv \boldsymbol{x} - \boldsymbol{x}_i$. If $\mathcal{P}$ is an application of an exact inverse, the residual is converted into an exact (up to numerical round-off) error.

We will denote the application of a particular Krylov subspace method (`KSP`) for the linear system (4.68) as $\mathcal{K}_x(\boldsymbol{r}(\boldsymbol{A}, \boldsymbol{b}))$. Upon preconditioning the system via $\mathcal{P}$ as in (4.69), we write

$$\mathcal{K}_x(\mathcal{P}\boldsymbol{r}(\boldsymbol{A}, \boldsymbol{b})). \tag{4.70}$$

If (4.70) is solved directly via $\mathcal{P} = \boldsymbol{A}^{-1}$, then $\mathcal{P}\boldsymbol{r}(\boldsymbol{A}, \boldsymbol{b}) = \boldsymbol{A}^{-1}\boldsymbol{b} - \boldsymbol{x}$. So (4.70) then becomes $\mathcal{K}_x(\boldsymbol{r}(\boldsymbol{I}, \boldsymbol{A}^{-1}\boldsymbol{b}))$, producing the exact solution of (4.68) in a single iteration of $\mathcal{K}$. Having established notation, we now present our implementation of static condensation via Slate by defining the appropriate operator, $\mathcal{P}$.

4.4.1 Interfacing with PETSc via custom preconditioners

The implementation of preconditioners in Firedrake requires manipulation not of assembled matrices, but rather their symbolic representation. To do this, we use the preconditioning infrastructure developed by Kirby and Mitchell (2018b), which gives preconditioners written in Python access to the symbolic problem description. In Firedrake, this means all derived preconditioners have direct access to the UFL representation of the PDE system. From this mathematical specification, we manipulate this appropriately via Slate and provide operators assembled from Slate expressions to PETSc for further algebraic preconditioning. Using this approach, we have developed a static condensation interface for the hybridization of $H(\mathrm{div}) \times L^2$ mixed problems, and a generic interface for statically condensing finite element systems. The advantage of writing even the latter as a preconditioner is the ability to switch out the solution scheme for the system, even when nested inside a larger set of coupled equations or non-linear solver (Newton-based methods) at *runtime*.

PETSc supports user-defined preconditioners through petc4py. To use a Python `PC` in Firedrake, one only needs to set the appropriate solver options. Setting "`pc_type`" to be of type "`python`" tells PETSc that a custom petsc4py preconditioner is being provided. Then one needs to specify the type of Python preconditioner through the solver option: "`pc_python_type`." We shall provide explicit examples for the preconditioners considered in this section.

4.4.2 A general-purpose static condensation preconditioner

As discussed in Sections 4.3.1 and 4.3.2, one of the main advantages of using a hybridizable variant of a DG or mixed method is that such systems permit the use of cell-wise static condensation. To facilitate this, we provide a PETSc `PC` static condensation interface: `firedrake.SCPC`. This preconditioner takes *any* static-condensation-amenable system (such as (4.46)) and performs the local elimination and recovery procedures. Slate expressions are generated from the underlying UFL problem description.

As an abstract example, suppose we have a block linear system of the form:

$$\left[\begin{array}{ccc|ccc} \boldsymbol{A}_{e_1,e_1} & \cdots & \boldsymbol{A}_{e_1,e_n} & \boldsymbol{A}_{e_1,c_1} & \cdots & \boldsymbol{A}_{e_1,c_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{e_n,e_1} & \cdots & \boldsymbol{A}_{e_n,e_n} & \boldsymbol{A}_{e_n,c_1} & \cdots & \boldsymbol{A}_{e_n,c_m} \\ \hline \boldsymbol{A}_{c_1,e_1} & \cdots & \boldsymbol{A}_{c_1,e_n} & \boldsymbol{A}_{c_1,c_1} & \cdots & \boldsymbol{A}_{c_1,c_m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{c_m,e_1} & \cdots & \boldsymbol{A}_{c_m,e_n} & \boldsymbol{A}_{c_m,c_1} & \cdots & \boldsymbol{A}_{c_m,c_m} \end{array}\right] \left\{\begin{array}{c} \boldsymbol{X}_{e_1} \\ \vdots \\ \boldsymbol{X}_{e_n} \\ \hline \boldsymbol{X}_{c_1} \\ \vdots \\ \boldsymbol{X}_{c_m} \end{array}\right\} = \left\{\begin{array}{c} \boldsymbol{R}_{e_1} \\ \vdots \\ \boldsymbol{R}_{e_n} \\ \hline \boldsymbol{R}_{c_1} \\ \vdots \\ \boldsymbol{R}_{c_m} \end{array}\right\}, \tag{4.71}$$

where $\boldsymbol{X}_{e_i}$, $\boldsymbol{X}_{c_j}$ are coefficient vectors for the fields with indices e_i and c_j, respectively, and $\boldsymbol{R}_{e_i}$, $\boldsymbol{R}_{c_j}$ are the associated right-hand sides. More compactly, we can rewrite (4.71) using block-matrix notation

$$\begin{bmatrix} \boldsymbol{A}_{e,e} & \boldsymbol{A}_{e,c} \\ \boldsymbol{A}_{c,e} & \boldsymbol{A}_{c,c} \end{bmatrix} \begin{Bmatrix} \boldsymbol{X}_e \\ \boldsymbol{X}_c \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{R}_e \\ \boldsymbol{R}_c \end{Bmatrix}, \tag{4.72}$$

where $\boldsymbol{X}_e$, $\boldsymbol{X}_c$, $\boldsymbol{R}_e$, and $\boldsymbol{R}_c$ are block-vectors. The system (4.72) is assumed to be static-condensation-amenable in the sense that all fields with indices $e_1, \cdots, e_n$ are *cell-local*. This allows for a reordering of (4.71) such that the matrix associated with the cell-local degrees of freedom $\boldsymbol{A}_{e,e}$ is rendered block-diagonal (cf. Figures 3.5A and 3.5B in Section 3.1.2).

Mathematically, the `firedrake.SCPC` preconditioner can be interpreted as a Schur-complement method for (4.72) of the form:

$$\boldsymbol{\mathcal{P}} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{A}_{e,e}^{-1}\boldsymbol{A}_{e,c} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{e,e}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_c^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}_{c,e}\boldsymbol{A}_{e,e}^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{4.73}$$

where $\boldsymbol{S}_c = \boldsymbol{A}_{c,c} - \boldsymbol{A}_{c,e}\boldsymbol{A}_{e,e}^{-1}\boldsymbol{A}_{e,c}$ is the Schur-complement operator for the $\boldsymbol{X}_c$ system. The distinction here from block preconditioners via the PETSc `fieldsplit` option, described in Section 4.1.1, is that the application of $\boldsymbol{\mathcal{P}}$ does *not* require two global inversions. In fact, the global system (4.72) is not even required to be explicitly assembled. By construction, $\boldsymbol{A}_{e,e}^{-1}$ is *sparse* and can be evaluated cell-wise. As a result, $\boldsymbol{S}_c$ is sparse and can be assembled *exactly* (up to numerical round-off) by computing the element tensors $\boldsymbol{S}^K$:

$$\boldsymbol{S}_c^K = \boldsymbol{A}_{c,c}^K - \boldsymbol{A}_{c,e}^K \left(\boldsymbol{A}_{e,e}^K\right)^{-1} \boldsymbol{A}_{e,c}^K. \tag{4.74}$$

The only globally coupled system requiring iterative inversion is $\boldsymbol{S}$:

$$\boldsymbol{\mathcal{K}}_{X_c}(\boldsymbol{\mathcal{P}}_s \boldsymbol{r}(\boldsymbol{S}_c, \boldsymbol{R}_s)), \tag{4.75}$$

Algorithm 2 : `firedrake.SCPC`

Require: Static-condensation-amenable finite element system:

$$\begin{bmatrix} A_{e,e} & A_{e,c} \\ A_{c,e} & A_{c,c} \end{bmatrix} \begin{Bmatrix} X_e \\ X_c \end{Bmatrix} = \begin{Bmatrix} R_e \\ R_c \end{Bmatrix}$$

1: Evaluate S cell-wise: $S_c^K = A_{c,c}^K - A_{c,e}^K \left(A_{e,e}^K\right)^{-1} A_{e,c}^K$ ▷ Static condensation
2: Evaluate R_s cell-wise: $R_s^K = R_c^K - A_{c,e}^K \left(A_{e,e}^K\right)^{-1} R_e^K$ ▷ Forward elimination
3: Globally solve for X_c:

$$\mathcal{K}_{X_c}(\mathcal{P}_s r(S_c, R_s))$$

4: Compute X_e cell-wise: $X_e^K = \left(A_{e,e}^K\right)^{-1} \left(R_c^K - A_{e,c}^K X_c^K\right)$ ▷ Back-substitution
return $X = \begin{Bmatrix} X_e & X_c \end{Bmatrix}^T$

LISTING 4.4: Example invocation of the Firedrake preconditioner: `firedrake.SCPC`.

```
-pc_type python
-pc_python_type firedrake.SCPC
-pc_sc_eliminate_fields 0, 1
-condensed_field_ksp_type cg
-condensed_field_pc_type ilu
```

where R_s is the condensed right-hand side, evaluated locally via:

$$R_s^K = R_c^K - A_{c,e}^K \left(A_{e,e}^K\right)^{-1} R_e^K, \tag{4.76}$$

and $\mathcal{P}_s$ is a preconditioner for S. Once X_c is computed, X_e is recovered via back-substitution. This is achieved by inverting the linear system element-by-element:

$$A_{e,e}^K X_e^K = R_c^K - A_{e,c}^K X_c^K. \tag{4.77}$$

All matrix inversions are computed using an LU factorization. Algorithm 2 describes the entire application procedure. Static condensation, forward elimination, and back-substitution are all performed via Slate-generated kernels.

The solver option "`pc_sc_eliminate_fields`" controls which unknowns are locally eliminated by providing a list of field indices. Eliminating the specified unknowns cell-wise will produce a condensed system for the remaining unknowns. The linear solver for the condensed system can be configured just like any PETSc solver object. This can be programmed directly using the options prefix: "`condensed_field`." For example, suppose we have a three-field problem with indices 0, 1, and 2. Then to configure this `PC` to eliminate fields "`0,1`" and solve the condensed problem using a preconditioned conjugate gradient method (ILU), one might use options like in Listing 4.4.

This preconditioner is suitable for both hybridized mixed and HDG discretizations. It can also be used within other contexts, such as the static condensation of continuous Galerkin discretizations (Guyan, 1965; Irons, 1965) or primal-hybrid methods (Devloo et al., 2018). Optimal choices of $\mathcal{P}_s$ for the condensed system will depend heavily on the application. The advantage of using `firedrake.SCPC` is that it provides a flexible environment for rapid experimentation with different choices of $\mathcal{P}_s$.

4.4.3 Preconditioning mixed methods via hybridization

The preconditioner `firedrake.HybridizationPC` expands on the previous one. This time, we take an $H(\mathrm{div}) \times L^2$ discretization of a mixed method for a scalar elliptic equation and automatically form the hybridizable problem. This is accomplished through manipulating the UFL objects representing the discretized PDE. This includes replacing argument spaces with their discontinuous counterparts, introducing test functions on an appropriate trace space, and providing operators assembled from Slate expressions in a similar manner as described in Section 4.4.2.

As a concrete example, consider the mixed finite element problem in Section 4.3.1, specifically equations (4.38)–(4.39). The problem we originally have (in matrix form) is:

$$\boldsymbol{\mathcal{A}}\boldsymbol{X} = \boldsymbol{R}, \quad \boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{D} \end{bmatrix}, \quad \boldsymbol{X} = \begin{Bmatrix} \boldsymbol{U} \\ \boldsymbol{P} \end{Bmatrix}, \quad \boldsymbol{R} = \begin{Bmatrix} \boldsymbol{R}_U \\ \boldsymbol{R}_P \end{Bmatrix}, \tag{4.78}$$

where $\boldsymbol{\mathcal{A}}$ is the matrix given in (4.40) and $\boldsymbol{R}$ is the right-hand side information. The process of transforming (4.78) into its equivalent hybridizable form is mathematically straightforward.

Since all Firedrake Python preconditioners have direct access to the UFL representation of the PDE, we know precisely what function spaces are present. In particular, the preconditioner knows about the mixed function spaces $U_h \times V_h \subset H(\mathrm{div}) \times L^2$ and the UFL forms describing the problem. From this information, the preconditioner can perform the following:

1. Given U_h, we know the family of the $H(\mathrm{div})$-finite element, its degree, and the mesh its defined on. Using this information, its broken version $\widehat{U}_h$ can immediately be constructed.[8]
2. From the degree of the $H(\mathrm{div})$-finite element space, we can determine the degree of the corresponding trace space M_h.
3. Given the UFL forms describing the mixed formulation, we can create a new UFL expression by *replacing* its arguments with test/trial functions from $\widehat{U}_h$.
4. The preconditioner has access to any strong boundary conditions imposed in U_h and its associated boundary markers on the mesh. Using this information together with the trace space M_h, the UFL form is extended by adding interior and exterior facet integrals containing the Lagrange multiplier.
5. The jump condition enforcing normal-component continuity of the trial function in $\widehat{U}_h$ is added to the new UFL problem.

Items 1–5 require nothing more than targeted inspection and manipulation of UFL objects. The allows the preconditioner to form the hybridizable system:

$$\left[\begin{array}{cc|c} \widehat{\boldsymbol{A}} & -\widehat{\boldsymbol{B}}^T & \boldsymbol{C}^T \\ \widehat{\boldsymbol{B}} & \boldsymbol{D} & \boldsymbol{0} \\ \hline \boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} \end{array}\right] \left\{\begin{array}{c} \widehat{\boldsymbol{U}} \\ \boldsymbol{P} \\ \hline \boldsymbol{\Lambda} \end{array}\right\} = \left\{\begin{array}{c} \widehat{\boldsymbol{R}}_U \\ \boldsymbol{R}_P \\ \hline \boldsymbol{R}_g \end{array}\right\}, \tag{4.79}$$

where $\widehat{\cdot}$ denotes modified matrices and vectors corresponding to forms with replaced arguments in $\widehat{U}_h$ ($\boldsymbol{D}$ and $\boldsymbol{R}_P$ are unchanged), $\widehat{\boldsymbol{U}}$ is the coefficient vector for

[8] Every finite element space in UFL can be made discontinuous using built-in UFL functions that topologically re-associates degrees of freedom with the cell only.

the discontinuous flux, $\boldsymbol{\Lambda}$ is the coefficient vector for the Lagrange multiplier. The last equation in (4.79) is the matrix-form of the jump condition in (4.45): $\boldsymbol{C}\widehat{\boldsymbol{U}} = \boldsymbol{R}_g$, where the vector $\boldsymbol{R}_g$ appears as a result of weakly incorporating strong boundary conditions on the flux variable. Equation (4.79) can be written more compactly as the block system:

$$\begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}} & \boldsymbol{\mathcal{C}}^T \\ \boldsymbol{\mathcal{C}} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \widehat{\boldsymbol{X}} \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \widehat{\boldsymbol{R}} \\ \boldsymbol{R}_g \end{Bmatrix}, \tag{4.80}$$

with

$$\widehat{\boldsymbol{\mathcal{A}}} = \begin{bmatrix} \widehat{\boldsymbol{A}} & -\widehat{\boldsymbol{B}}^T \\ \widehat{\boldsymbol{B}} & \boldsymbol{D} \end{bmatrix}, \quad \boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix}, \quad \widehat{\boldsymbol{X}} = \begin{Bmatrix} \widehat{\boldsymbol{U}} \\ \boldsymbol{P} \end{Bmatrix}, \quad \widehat{\boldsymbol{R}} = \begin{Bmatrix} \widehat{\boldsymbol{R}}_U \\ \boldsymbol{R}_P \end{Bmatrix}. \tag{4.81}$$

From this point, the application of `firedrake.HybridizationPC` follows identically to `firedrake.SCPC`; its application can be interpreted as a Schur-complement reduction of (4.80):

$$\widehat{\boldsymbol{\mathcal{P}}} = \begin{bmatrix} \boldsymbol{I} & -\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{C}}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_\lambda^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\mathcal{C}}\widehat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{4.82}$$

where $\boldsymbol{S}_\lambda$ is the Schur-complement matrix: $\boldsymbol{S}_\lambda = -\boldsymbol{\mathcal{C}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{C}}^T$. The inversion of $\widehat{\boldsymbol{\mathcal{A}}}$ does not require a global solver since $\widehat{\boldsymbol{U}}$ and $\boldsymbol{P}$ are cell-local. Similarly, (4.80) never needs to be explicitly assembled; only a single global problem for $\boldsymbol{\Lambda}$ is needed to completely solve the hybridizable system. The recovery of $\widehat{\boldsymbol{U}}$ and $\boldsymbol{P}$ happens in the same manner as in `firedrake.SCPC`.

As first shown by Arnold and Brezzi (1985), the flux solutions $\boldsymbol{u}_h$ for (4.78) and $\widehat{\boldsymbol{u}}_h$ for (4.79) approximate the same field using mathematically equivalent formulations. Since the input system provided to this preconditioner is the original mixed method (4.78), we need to project the computed solution into U_h. This ensures that the coefficient vector for the computed solution has the correct dimension.

To do this, we follow Arbogast and Chen (1995) and introduce an averaging operator $\pi_{\text{avg}} : \widehat{U}_h \to U_h$. Then for any $\widehat{\boldsymbol{w}} \in \widehat{U}_h$, we define $\pi_{\text{avg}}(\widehat{\boldsymbol{w}})$ as the function with:

$$\pi_{\text{avg}}(\widehat{\boldsymbol{w}})|_K = \widehat{\boldsymbol{w}}|_K, \quad K \in \mathcal{T}_h, \tag{4.83}$$

$$\pi_{\text{avg}}(\widehat{\boldsymbol{w}})|_K \cdot \boldsymbol{n}|_e = \frac{1}{2}\left(\widehat{\boldsymbol{w}}|_{K^+} \cdot \boldsymbol{n}|_e + \widehat{\boldsymbol{w}}|_{K^-} \cdot \boldsymbol{n}|_e\right), \quad e = \partial K^+ \cap \partial K^- \in \mathcal{E}_h^\circ, \tag{4.84}$$

$$\pi_{\text{avg}}(\widehat{\boldsymbol{w}})|_K \cdot \boldsymbol{n}|_e = \widehat{\boldsymbol{w}}|_K \cdot \boldsymbol{n}|_e, \quad e \in \mathcal{E}_h^\partial. \tag{4.85}$$

A straightforward calculation verifies that $\pi_{\text{avg}}(\widehat{\boldsymbol{w}})$ has continuous normal components on $\mathcal{E}_h^\circ$, hence $\pi_{\text{avg}}(\widehat{\boldsymbol{w}}) \in U_h$:

$$\begin{aligned} [\![\pi_{\text{avg}}(\widehat{\boldsymbol{w}})]\!]_e &= \frac{1}{2}\left(\widehat{\boldsymbol{w}}|_{K^+} \cdot \boldsymbol{n}^+|_e + \widehat{\boldsymbol{w}}|_{K^-} \cdot \boldsymbol{n}^+|_e\right) + \frac{1}{2}\left(\widehat{\boldsymbol{w}}|_{K^-} \cdot \boldsymbol{n}^-|_e + \widehat{\boldsymbol{w}}|_{K^+} \cdot \boldsymbol{n}^-|_e\right) \\ &= \frac{1}{2}\left(\widehat{\boldsymbol{w}}|_{K^+} + \widehat{\boldsymbol{w}}|_{K^-} - \widehat{\boldsymbol{w}}|_{K^-} - \widehat{\boldsymbol{w}}|_{K^+}\right) \cdot \boldsymbol{n}^+|_e = 0, \end{aligned} \tag{4.86}$$

for all $e = \partial K^+ \cap \partial K^-$, $K^+, K^- \in \mathcal{T}_h$. In terms of implementation, evaluating $\pi_{\text{avg}}(\widehat{\boldsymbol{u}}_h)$ can be done by simply averaging the nodal values $\widehat{\boldsymbol{U}}$. The coefficient vector for (4.78) is then updated via the matrix-vector product: $\boldsymbol{U} \leftarrow \boldsymbol{\Pi}_{\text{avg}}\widehat{\boldsymbol{U}}$. More specifically, the

operator $\mathbf{\Pi}_{\text{avg}}$ is an averaging matrix in $\mathbb{R}^{\dim U_h \times \dim \widehat{U}_h}$ with entries:

$$\left(\mathbf{\Pi}_{\text{avg}}\right)_{ij} = |\omega_i|^{-1}\chi(i;\widehat{\mathbf{\Psi}}_j), \tag{4.87}$$

where $|\omega_i|$ denotes the number of cells K sharing the global degree of freedom (node) n_i of U_h, and $\chi(i;\widehat{\mathbf{\Psi}}_j)$ is an indicator function taking a value of 0 if the basis function $\widehat{\mathbf{\Psi}}_j \in \widehat{U}_h$ vanishes at n_i and 1 otherwise.

Note that for any degree of freedom for U_h, there will be exactly *one* nodal basis function $\mathbf{\Psi}_i \in U_h$ taking a value of one at that node. However, if a degree of freedom is associated with an interior facet, there will be *two* basis functions $\widehat{\mathbf{\Psi}}_i^+, \widehat{\mathbf{\Psi}}_i^- \in \widehat{U}_h$ taking values of one at that node (each contributing to the node from both sides of the two cells sharing the facet). This due to the fact that $\widehat{U}_h$ is a globally discontinuous finite element space.

It is also worth mentioning that $\mathbf{\Pi}_{\text{avg}}$ is never explicitly assembled by this preconditioner; instead its matrix-vector product is evaluated by directly averaging the entries of $\widehat{\boldsymbol{U}}$. This process can be done cheaply and requires a negligible amount of computational work.

With $\widehat{\boldsymbol{\mathcal{P}}}$ as in (4.82), applying `firedrake.HybridizationPC` on the *original* system $\boldsymbol{\mathcal{A}}\boldsymbol{X} = \boldsymbol{R}$ can be described mathematically as:

$$\boldsymbol{\mathcal{P}} = \mathbf{\Pi}\widehat{\boldsymbol{\mathcal{P}}}\mathbf{\Pi}^T, \quad \mathbf{\Pi} = \begin{bmatrix} \mathbf{\Pi}_{\text{avg}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I} & \mathbf{0} \end{bmatrix}. \tag{4.88}$$

It is worth noting here that assembly of the right-hand side for the Lagrange multiplier system requires slightly more attention. We specifically need to address the co-vector $\boldsymbol{R}_U$ in the $H(\text{div})$-finite element space U_h.

The situation we are given is that we have $\boldsymbol{R}_U = \boldsymbol{R}_U(\boldsymbol{w})$ for $\boldsymbol{w} \in U_h$, but require $\widehat{\boldsymbol{R}}_U(\widehat{\boldsymbol{w}})$ for $\widehat{\boldsymbol{w}} \in \widehat{U}_h$. For consistency, we additionally require:

$$\widehat{\boldsymbol{R}}_U(\boldsymbol{w}) = \boldsymbol{R}_U(\boldsymbol{w}), \quad \forall \boldsymbol{w} \in U_h. \tag{4.89}$$

We can construct such a $\widehat{\boldsymbol{R}}_U$ in the following way. First, let n_i denote a global degree of freedom for U_h associated with a particular mesh entity (cell interiors or facets) and its associated basis function $\mathbf{\Psi}_i \in U_h$. Then by construction, we have:

$$\mathbf{\Psi}_i = \begin{cases} \widehat{\mathbf{\Psi}}_i|_e & \text{for } n_i \text{ associated with } e \in \mathcal{E}_h^\partial, \\ \widehat{\mathbf{\Psi}}_i^+|_e + \widehat{\mathbf{\Psi}}_i^-|_e & \text{for } n_i \text{ associated with } e \in \mathcal{E}_h^\circ, \\ \widehat{\mathbf{\Psi}}_i|_K & \text{for } n_i \text{ associated with } K \in \mathcal{T}_h, \end{cases} \tag{4.90}$$

where $\widehat{\mathbf{\Psi}}_i, \widehat{\mathbf{\Psi}}_i^\pm \in \widehat{U}_h$ are basis functions whose evaluations at n_i are non-zero. For degrees of freedom on interior facets, both $\widehat{\mathbf{\Psi}}_i^\pm$ contribute half of the total contribution on a given interior facet. Therefore, we define our "broken" right-hand side as the locally averaged expression:

$$\widehat{\boldsymbol{R}}_U(\widehat{\mathbf{\Psi}}_i)|_K = |\omega_i|^{-1}\boldsymbol{R}_U(\mathbf{\Psi}_i)|_K, \quad K \in \mathcal{T}_h, \tag{4.91}$$

where $|\omega_i|$ is as defined previously: the total number of cells sharing the degree of freedom n_i.

Algorithm 3 : `firedrake.HybridizationPC`

Require: Mixed formulation of a scalar elliptic equation: $\mathcal{A}X = R$

1: Formulate the hybridizable system:

$$\begin{bmatrix} \widehat{\mathcal{A}} & \mathcal{C}^T \\ \mathcal{C} & 0 \end{bmatrix} \begin{Bmatrix} \widehat{X} \\ \Lambda \end{Bmatrix} = \begin{Bmatrix} \widehat{R} \\ R_g \end{Bmatrix}$$

2: Evaluate S_λ cell-wise: $S_\lambda^K = -\mathcal{C}^K \left(\widehat{\mathcal{A}}^K\right)^{-1} \left(\mathcal{C}^K\right)^T$ ▷ Static condensation

3: Evaluate $R \to \widehat{R}$ ▷ Transfer right-hand side

4: Evaluate R_s cell-wise: $R_s^K = R_g^K - \mathcal{C}^K \left(\widehat{\mathcal{A}}^K\right)^{-1} \widehat{R}^K$ ▷ Forward elimination

5: Globally solve for Λ:

$$\mathcal{K}_\Lambda(\mathcal{P}_s r(S_\lambda, R_s))$$

6: Compute $\widehat{X}$ cell-wise: $\widehat{X}^K = \left(\widehat{\mathcal{A}}^K\right)^{-1} \left(\widehat{R}^K - \left(\mathcal{C}^K\right)^T \Lambda^K\right)$ ▷ Back-substitution

7: Evaluate $X \leftarrow \Pi \widehat{X}$ ▷ Project

return $X = \begin{Bmatrix} U & P \end{Bmatrix}^T$

LISTING 4.5: Example invocation of the preconditioner: `firedrake.HybridizationPC`.

```
-pc_type python
-pc_python_type firedrake.HybridizationPC
-hybridization_ksp_type cg
-hybridization_pc_type icc
```

Using (4.90), (4.91), and the fact that R_U is linear in its argument, we can verify that our construction of $\widehat{R}_U$ satisfies (4.89). Evaluating (4.91) can be performed similarly to the averaging of the coefficient vector $\widehat{U}$. We do this by just averaging the assembled dual vector directly using Π_{avg}, $\widehat{R}_U \leftarrow \Pi_{\text{avg}}^T R_U$.

Algorithm 3 describes the application of the `firedrake.HybridizationPC` solver object. It can be applied to a wide array of mixed methods which permit direct hybridization. The advantage of a preconditioner like this is that the actual hybridization occurs *automatically*; therefore one does not need to modify their original problem description to use this `PC`. For example, the initial mixed Poisson demonstration in Section 4.1.1 (Listing 4.1) can be solved using this preconditioner *without* actually having to write down the hybridizable formulation. Instead, this hybridization procedure can be readily applied by specifying the necessary solver options.

Listing 4.5 demonstrates how one might invoke this preconditioner on a suitable mixed discretization. The options prefix "`hybridization_`" is used to configure the linear solver object for the Lagrange multipliers. In the example, we chose to use a preconditioned conjugate gradient method using an incomplete Cholesky factorization. Having presented our static condensation preconditioners, it will now be useful to discuss preconditioning strategies for the Lagrange multipliers. Specifically, we argue that AMG-based preconditioning is effective, especially for the trace systems arising out of hybridizable mixed methods.

4.4.4 Preconditioning the Lagrange multiplier system

We shall focus now on the linear solver for the Lagrange multipliers:

$$\boldsymbol{\mathcal{K}}_{\boldsymbol{\Lambda}}(\boldsymbol{\mathcal{P}}_s \boldsymbol{r}(\boldsymbol{S}_\lambda, \boldsymbol{R}_s)), \tag{4.92}$$

where $\boldsymbol{S}_\lambda$ is the statically condensed trace operator, $\boldsymbol{R}_s$ is the right-hand side, and $\boldsymbol{\mathcal{P}}_s$ is the preconditioner. The choice of $\boldsymbol{\mathcal{P}}_s$ and $\boldsymbol{\mathcal{K}}$ will be the main focus of this section. We shall focus our discussion the case where (4.92) is the condensed system for the hybridizable mixed method.

For the hybridization of the model mixed problem (4.38)–(4.39) from Section 4.3.1, it was shown that $\boldsymbol{S}_\lambda$ is symmetric positive-definite and spectrally equivalent to a second-order elliptic operator (Gopalakrishnan, 2003; Cockburn and Gopalakrishnan, 2005). As a result, the optimal choice of $\boldsymbol{\mathcal{K}}$ is the conjugate gradient method.

It was shown by Brunner and Kolev (2011), both theoretically and demonstrated in practice, that classical AMG (Ruge and Stüben, 1987; Stüben, 2001) (where coarsening is performed using subsets of degrees of freedom) gives mesh-independent convergence for Schur-complement systems arising from mixed formulations of scalar elliptic equations. This naturally leads to considering similar approaches for (4.92).

To further motivate us in this direction, we summarize a key result by Cockburn and Gopalakrishnan (2005). Consider the following norm for the space of Lagrange multipliers, M_h:

$$|||\lambda|||^2 := \sum_{K \in \mathcal{T}_h} \frac{1}{|K|} \|\lambda - \mathscr{M}(\lambda)\|^2_{L^2(\partial K)}, \tag{4.93}$$

where $\mathscr{M}(\lambda) = \frac{1}{|\partial K|} \int_{\partial K} \lambda \, \mathrm{d}S$ denotes the best-approximating constant to $\lambda \in M_h$. Cockburn and Gopalakrishnan (2005, Theorem 3.6) showed that the energy-norm induced by $\boldsymbol{S}_\lambda$ is *equivalent* to $|||\cdot|||$ with mesh-independent constants. Given the definition of $|||\cdot|||$, this also shows that $\boldsymbol{S}_\lambda$ has a *near null-space* (null-space of the operator without boundary conditions) consisting of only constant trace functions. Since AMG methods know nothing about the geometry of the problem, they rely on the properties of the near null-space to develop effective coarsening strategies. To illustrate our point, we first briefly describe a generic AMG method.

A quick overview of algebraic multigrid (AMG)

Recall that the two main principles of any multigrid method are (Stüben, 2001):

1. Relaxation (or smoothing) principle: A simple iterative method such as Jacobi, Richardson, or successive over-relaxation (SOR)[9] methods are well-known to have strong smoothing properties. Therefore, a smooth error term can be produced with relative ease.
2. Coarse-grid correction principle: A smooth error term can be approximated well on a coarser grid, thus saving a substantial amount of computational work.

[9] Gauss-Seidel is a notable example of an SOR-type method.

In linear algebra terms, the "grids" for AMG are represented simply as the vector spaces $\mathbb{R}^n$ (fine-grid) and the lower-dimensional space (coarse) $\mathbb{R}^c$. *Prolongation* (interpolation) operators map data from coarse-grid to fine-grid as the full-rank $n \times c$ matrix $\mathcal{P}_n^c : \mathbb{R}^c \to \mathbb{R}^n$, and the associated *restriction* operator maps fine-grid data to the coarse-grid: $\mathcal{R}_c^n = (\mathcal{P}_n^c)^T$.[10]

Then a two-grid AMG method for solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric positive-definite, can be summarized as the following steps:

1. (Pre-smoothing): Perform ν_1 smoothing iterations $\tilde{\boldsymbol{x}} = \texttt{Smooth}(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b}; \nu_1)$.
2. (Evaluate the residual): $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{\epsilon}$.
3. (Restrict the residual): $\boldsymbol{r}^c = \mathcal{R}_c^n \boldsymbol{r}$.
4. (Coarse-grid correction): $\boldsymbol{\epsilon}^c = \texttt{Solve}(\boldsymbol{A}^c, \boldsymbol{r}^c)$.
5. (Prolong correction): $\hat{\tilde{\boldsymbol{x}}} = \tilde{\boldsymbol{x}} + \mathcal{P}_n^c \boldsymbol{\epsilon}^c$.
6. (Post-smoothing): Perform ν_2 smoothing iterations $\boldsymbol{x} = \texttt{Smooth}(\hat{\tilde{\boldsymbol{x}}}, \boldsymbol{A}, \boldsymbol{b}, \nu_2)$.

In practice, step 4 is solved recursively by applying steps 1–3 until the coarsest-grid is reached. After the coarse-grid solve, the error is prolonged back to the fine-grid, accompanied by post-smoothing after each transfer. This requires a hierarchy of AMG "grids," transfer operators, and coarse-grid matrices. Going straight from fine-grid to coarse-grid in one-sweep is referred to as a "V-cycle."

We refer to $\boldsymbol{\epsilon}$ as the error in the exact solution at the current iterate: $\boldsymbol{\epsilon} = \boldsymbol{A}^{-1}\boldsymbol{b} - \tilde{\boldsymbol{x}}$. In step 4, we solve (usually via a direct method) for the error on the coarsest-grid $\boldsymbol{\epsilon}^c$, which is an approximation to the error on the fine-grid. The coarse operator is defined using the Galerkin product: $\boldsymbol{A}^c = \mathcal{R}_c^n \boldsymbol{A} \mathcal{P}_n^c$. Note that the choice of $\mathcal{R}_c^n$ ensures that $\boldsymbol{A}^c$ remains symmetric:

$$(\boldsymbol{A}^c)^T = (\mathcal{R}_c^n \boldsymbol{A} \mathcal{P}_n^c)^T = (\mathcal{P}_n^c)^T \boldsymbol{A}^T \left((\mathcal{P}_n^c)^T\right)^T = \mathcal{R}_c^n \boldsymbol{A} \mathcal{P}_n^c = \boldsymbol{A}^c. \tag{4.94}$$

Arguably the most critical part of the AMG procedure is the choice of smoothing. This is important due to the fact that error not eliminated by the initial smoother must then be handled by the coarse-grid solver. The overall convergence of the AMG method relies on the smoother removing highly-oscillatory error, leaving a so-called *smooth error* which can be effectively approximated by the coarse-grid correction. If the smoother fails to remove the oscillatory data, then the AMG procedure is not guaranteed to converge (Fulton, Ciesielski, and Schubert, 1986; Ruge and Stüben, 1987; Stüben, 2001).

In general, smooth error corresponds to the eigenvectors of $\boldsymbol{A}$ whose eigenvalues are small in magnitude (small eigenmodes). It is therefore the primary job of the smoother to eliminate the error corresponding to large eigenvalues. That is, the smaller the eigenmodes, the more effective the coarse-grid correction will be (Falgout, 2006). The smallest of the eigenmodes are the elements of the *near null-space*, and therefore knowing the near null-space properties of the operator is critically important in the design of an effective AMG procedure.

[10] This is not the only way to construct the restriction operator. It is, however, a popular choice since it preserves symmetry of the coarse-grid operator.

AMG for the hybridizable mixed method

Classical AMG of Ruge and Stüben (1987) and Falgout, Jones, and Yang (2006) is based on the assumption that geometrically smooth functions (non-oscillatory) are in the near null-space of the operator. Since constant functions are geometrically smooth, the trace operator $\boldsymbol{S}_\lambda$ in (4.92) fits exactly within classical AMG heuristics. This was verified experimentally by Dobrev et al. (2019), where hypre (Baker et al., 2011), a widely-available classical AMG library, was applied on the trace system for a hybridized mixed method of a scalar-valued second-order elliptic problem. There, they showed that black-box AMG preconditioning for the Lagrange multipliers exhibits good weak scaling (up to 1,536 processes).

Additionally, the *smoothed aggregation* (SA) method (Vaněk, Mandel, and Brezina, 1996) is another highly successful AMG procedure. SA methods differ from classical AMG in that coarsening is defined by building aggregates of finer-level degrees of freedom rather than subsets. The SA method relies on the near null-space components to build the prolongation operators in way which preserves the components upon interpolation. This exact interpolation of near null-space components is an essential part of SA methods and is a necessary condition for mesh-independent convergence (Vaněk, Brezina, and Mandel, 2001).

In most of the numerical experiments presented in the rest of this chapter and Chapter 5, the SA multigrid approach is chosen to be preconditioner for the Lagrange multiplier systems. This was partly due to experimentation, as both the PETSc-native SA multigrid preconditioner GAMG (Balay et al., 2016, §4.4.5) and Trilinos' ML implementation of SA multigrid (Gee et al., 2006) consistently produce parameter-robust solver convergence (even for non-symmetric Lagrange multiplier systems). These will be discussed in more detail in Sections 4.5.3 and 5.4.4.

A remark on the HDG trace system

To conclude this section, it is worth mentioning possible strategies for the trace system associated the LDG-H method in Section 4.3.2. The LDG-H trace system is provably symmetric and positive-definite (Cockburn, Gopalakrishnan, and Sayas, 2010). Furthermore, it was shown by Cockburn et al. (2013, Theorem 3.3) that a spectrally equivalent preconditioner for the hybridizable mixed method could also be used to precondition the LDG-H trace system. However, strong non-diagonally dominant matrix structure was observed in the LDG-H trace operator, rendering the performance of direct AMG (on the trace space) unclear (Kronbichler and Wall, 2018). As a result, point-based smoothing like Jacobi iterations perform poorly in a multigrid method for the LDG-H trace system. Block-relaxation schemes like block-Jacobi or block-Gauss-Seidel should be considered instead.

A more promising alternative to direct AMG is summarized by Cockburn et al. (2013) and demonstrated in Kronbichler and Wall (2018). In that study, the trace system is restricted to a spectrally equivalent operator in a continuous P_1 finite element space. The hierarchy continues by means of an GMG solver for P_1-finite elements, where optimal multigrid performance was demonstrated. However, the P_1-nested AMG solver for the HDG method still falls short of the monolithic multigrid-based solvers for continuous Galerkin discretizations. Further investigations are needed to completely determine the practicality of HDG methods for elliptic equations.

4.5 Numerical studies

We now present results utilizing the Slate DSL and our static condensation preconditioners for a set of test problems. Since we are using the interfaces outlined in Section 4.4, Slate is accessed indirectly and requires no manually-written solver code for hybridization or static condensation/local recovery. All parallel results were obtained on a single fully-loaded compute node of dual-socket Intel E5-2630v4 (Xeon) processors with 2×10 cores (2 threads per core) running at 2.2GHz. In order to avoid potential memory effects due to the operating system migrating processes between sockets, we pin MPI processes to cores.

Verification of the generated code is performed using parameter-sensitive convergence tests. The study consists of running a variety of discretizations spanning the methods outlined in Section 4.3. Details, Firedrake programs, and numerical results are made public and can be viewed in Zenodo/Tabula-Rasa (2019). All results are in full agreement with the theory.

4.5.1 HDG method for a three-dimensional elliptic equation

In this section, we take a closer look at the LDG-H method for the model elliptic equation (sign-definite Helmholtz):

$$-\nabla \cdot \nabla p + p = f, \text{ in } \Omega = [0,1]^3, \quad p = g, \text{ on } \partial\Omega, \tag{4.95}$$

where f and g are chosen such that the analytic solution is:

$$p(x,y,z) = \exp\{\sin(\pi x)\sin(\pi y)\sin(\pi z)\}. \tag{4.96}$$

We use a regular mesh consisting $6 \cdot N^3$ tetrahedral elements ($N \in \{4, 8, 16, 32, 64\}$). First, we reformulate (4.95) as the first-order system:

$$\boldsymbol{u} + \nabla p = 0, \text{ in } \Omega \quad \nabla \cdot \boldsymbol{u} + p = f, \text{ in } \Omega \quad p = g, \text{ on } \partial\Omega. \tag{4.97}$$

We start with linear polynomial approximations, up to cubic, for the LDG-H discretization of (4.97). Additionally, we compute a post-processed scalar approximation $p_h^\star$ of the HDG solution. This raises the approximation order of the computed solution by an additional degree. In all numerical studies here, we set the HDG parameter $\tau = 1$. All results were computed in parallel, utilizing a single compute node (described previously).

A continuous Galerkin (CG) discretization of the primal problem (4.95) serves as a reference for this experiment. Due to the superconvergence in the post-processed solution for the HDG method, we use CG discretizations of polynomial order 2, 3, and 4. This takes into account the enhanced accuracy of the HDG solution, despite being initially computed as a lower-order approximation. We therefore expect both methods to produce equally accurate solutions to the model problem.

Our aim here is not to compare the performance of HDG and CG, which has been investigated elsewhere (for example, see Kirby, Sherwin, and Cockburn (2011) and Yakovlev et al. (2015)). Instead, we provide a reference that the reader might be more familiar with in order to evaluate whether our software framework produces a sufficiently performant HDG implementation relative to what might be expected.

LISTING 4.6: Solver options for configuring the primal CG solver. The relative tolerance for the Krylov solver is specified based on the mesh size.

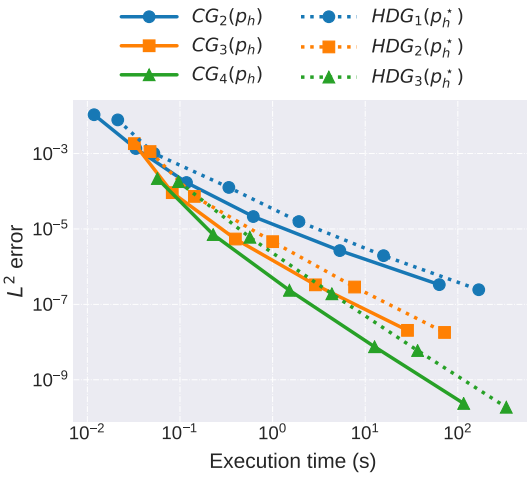
```
-ksp_type cg
-ksp_rtol rtol    # rtol changes with resolution
-pc_type hypre
-pc_hypre_type boomeramg
-pc_hypre_boomeramg_strong_threshold 0.75
-pc_hypre_boomeramg_agg_nl 2
```

LISTING 4.7: Solver options for configuring the primal HDG solver. The relative tolerance for the trace Krylov solver is specified based on the mesh size.

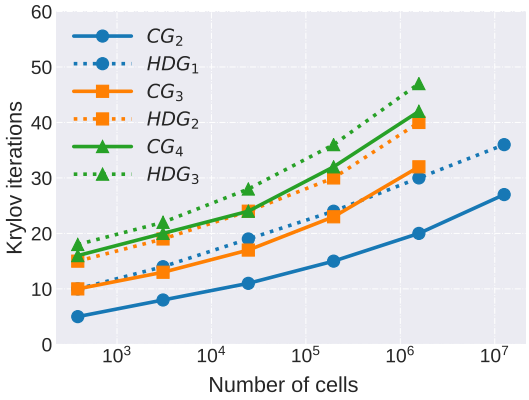
```
-mat_type matfree
-ksp_type preonly
-pc_type python
-pc_python_type firedrake.SCPC
-pc_sc_eliminate_fields 0, 1

# solver for the condensed system
-condensed_field_ksp_type cg
-condensed_field_ksp_rtol rtol    # rtol changes with resolution
-condensed_field_pc_type hypre
-condensed_field_pc_hypre_type boomeramg
-condensed_field_pc_hypre_boomeramg_strong_threshold 0.75
-condensed_field_pc_hypre_boomeramg_agg_nl 2
```

To invert the CG system, we use a preconditioned conjugate gradient solver with hypre's boomerAMG implementation of classical AMG as a preconditioner (Falgout, Jones, and Yang, 2006). For the HDG method, we use the static condensation preconditioner `firedrake.SCPC` described in Section 4.4.2. The statically condensed system is solved using the same preconditioned conjugate gradient strategy as the CG system. Based on the observations by Kronbichler and Wall (2018), the HDG trace system not diagonally dominant. Therefore, we can expect point-smoothers such as standard Jacobi or Gauss-Seidel to be largely ineffective. Instead, we use boomerAMG's "out of the box" modified block hybrid-Gauss-Seidel method as a smoother, following Kolev and Vassilevski (2009). This particular smoother is convergent for any symmetric positive-definite matrix (Baker et al., 2011). A full list of solver options used in this experiment is shown in Listing 4.6 and 4.7.

To avoid over-solving, we iterate to a relative tolerance such that the *discretization error* is minimal for a given mesh. In other words, the solvers are configured to terminate when there is no further reduction in the L^2-error of the computed solution compared with the analytic solution. This means we are not iterating to a fixed solver tolerance across all mesh resolutions. Therefore, we can expect the total number of Krylov iterations (for both the CG and HDG methods) to increase as the mesh resolution becomes finer. The rationale behind this approach is to directly compare the execution time to solve for the best possible approximation to the solution given a fixed resolution.



(A) A log-log plot showing the error against execution time for the CG and HDG with post-processing ($\tau = 1$) methods.



(B) A log-linear plot showing Krylov iterations of the AMG-preconditioned conjugate gradient algorithm (to reach discretization error) against number of cells.

FIGURE 4.3: Comparison of continuous Galerkin and LDG-H solvers for the model three-dimensional positive-definite Helmholtz equation.

Error versus execution time

The total execution time is recorded for the CG and HDG solvers, which includes the setup time for the AMG preconditioner, matrix assembly, and the time-to-solution for the Krylov method. In the HDG case, we include all setup costs, the time spent building the Schur-complement for the traces, local recovery of the scalar and flux approximations, and post-processing. The L^2-error against execution time is summarized in Figure 4.3A.

The HDG method of order $k-1$ (HDG_{k-1}) with post-processing, as expected, produces a solution which is as accurate as the CG method of order k (CG_k). While the full HDG system is never explicitly assembled, the larger execution time is a result of several factors. The primary factor is that the total number of trace unknowns for the HDG_1, HDG_2, and HDG_3 discretizations is roughly four, three, and two times larger (resp.) than the corresponding number of CG unknowns. Therefore, each iteration is more expensive. We also observe that the trace system requires more Krylov iterations to reach discretization error (see Figure 4.3B), which improves relative to the CG method as the approximation order increases. Further analysis on multigrid methods for HDG systems is required to draw further conclusions. The main computational bottleneck in HDG methods is the global linear solver. We therefore expect our implementation to be dominated by the cost associated with inverting the trace operator. If one considers just the time-to-solution, the CG method is clearly ahead of the HDG method. However, the superior scaling, local conservation, and stabilization properties of the HDG method make it a particularly appealing choice for fluid dynamics applications (Yakovlev et al., 2015; Kronbichler and Wall, 2018). Therefore, the development of good preconditioning strategies for the HDG method is critical for its competitive use.

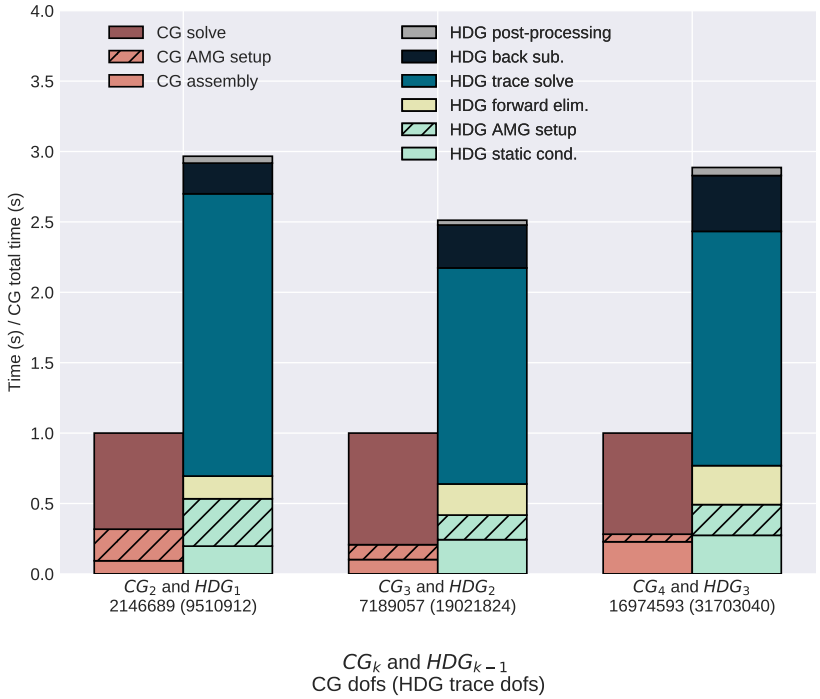


FIGURE 4.4: Break down of the CG_k and HDG_{k-1} execution times on a $6 \cdot 64^3$ simplicial mesh.

Break down of solver time

The HDG method requires solving for many more degrees of freedom than CG or primal DG methods. This is largely due to the fact that the HDG method simultaneously approximates the primal solution and its velocity. The global matrix for the traces is significantly larger than the one for the CG system at low polynomial order. The execution time for HDG is then compounded by a more expensive global solve.

Figure 4.4 displays a break down of total execution times on a simplicial mesh consisting of 1.5 million elements. The execution times have been normalized by the CG total time, showing that the HDG method is roughly 3 times more expensive than the CG method. This is expected given the larger degree-of-freedom count and expensive global solve. The raw numerical breakdown of the HDG and CG solvers are shown in Table 4.1. We isolate each component of the HDG method contributing to the total execution time: preconditioner setup costs, static condensation (trace operator assembly), forward elimination (right-hand side assembly for the trace system), backwards substitution, and local post-processing of the scalar solution. For all k, our HDG implementation is solver-dominated as expected.

Both trace operator and right-hand side assembly are dominated by the costs of inverting a local square mixed matrix coupling the scalar and velocity unknowns, which is performed directly via an LU factorization. This is also the case for backwards substitution. They should all therefore be of the same magnitude in time spent. We observe that this is the case across all degrees, with times ranging between approximately 6—10% of total execution time for assembling the condensed system. Back-substitution takes roughly the same time as the static condensation and forward elimination stages (approximately 12% of execution time on average). The slight increase in time is due to splitting the local equations into two local solver

TABLE 4.1: Breakdown of the raw timings for the HDG_{k-1} ($\tau = 1$) and CG_k methods, $k = 2$, 3, and 4. Each method corresponds to a mesh size $N = 64$ on a fully-loaded compute node.

Stage	HDG_1		HDG_2		HDG_3	
	t_{stage} (s)	% t_{total}	t_{stage} (s)	% t_{total}	t_{stage} (s)	% t_{total}
Static cond.	1.05	6.64 %	6.95	9.68 %	31.66	9.49 %
HDG AMG setup	1.79	11.34 %	5.00	6.95 %	25.21	7.56 %
Forward elim.	0.86	5.43 %	6.32	8.79 %	31.98	9.59 %
Trace solve	10.66	**67.59 %**	43.89	**61.09 %**	192.31	**57.65 %**
Back sub.	1.16	7.34 %	8.71	12.12 %	45.81	13.73 %
Post-process	0.26	1.65 %	0.98	1.36 %	6.62	1.98 %
HDG Total	15.77		71.85		333.58	
	CG_2		CG_3		CG_4	
	t_{stage} (s)	% t_{total}	t_{stage} (s)	% t_{total}	t_{stage} (s)	% t_{total}
Matrix assembly	0.50	9.32 %	2.91	10.19 %	26.37	22.81 %
CG AMG setup	1.19	22.43 %	3.02	10.55 %	6.21	5.38 %
Solve	3.63	**68.25 %**	22.67	**79.26 %**	82.99	**71.81 %**
CG Total	5.32		28.60		115.57	

passes: one for p_h and another for the velocity $\boldsymbol{u}_h$. Finally, the additional cost of post-processing accrues negligible time (roughly 2% of execution time across all degrees). This is a small pay-off for an increase in order of accuracy.

The setup times for the AMG preconditioner of both methods are displayed along side the rest of the components contributing to the total execution time. The setup time for the boomerAMG preconditioner grows steadily with the matrix size of the HDG trace system, but takes approximately the same proportion of execution time. Further investigations in the heuristics of classical and aggregation-type AMG methods for the HDG trace operator is required to make any concrete claims. Despite the uncertainty of direct AMG for the HDG method, the overall results we have shown here are quite similar to what has been observed in previous comparison studies with CG and HDG discretizations (Yakovlev et al., 2015; Kirby, Sherwin, and Cockburn, 2011; Kronbichler and Wall, 2018). However, geometric multigrid approaches combined with a P_1-nested multigrid method is already shown to be promising (Cockburn et al., 2013; Fabien et al., 2019). As a result, further investigations of this type of multigrid approach for more general HDG discretizations is a topic on on-going investigation.

We note that caching of local tensors does not occur. Each pass to perform the local eliminations and backwards reconstructions rebuilds the local element tensors. It is not clear at this time whether the performance gained from avoiding rebuilding the local operators will offset the memory costs of storing the local matrices. Moreover, in time-dependent problems where the operators may contain state-dependent variables, rebuilding local matrices will be necessary in each time-step regardless.

4.5.2 Hybridizable mixed methods for the shallow water equations

A primary motivator for our interest in hybridizable methods revolves around developing efficient solvers for problems in geophysical flows. In Section 3.3.1, we presented some results integrating the nonlinear shallow water equations on the sphere

TABLE 4.2: The number of unknowns to be determined are summarized for each compatible finite element method. Resolution is the same for both methods. A time-step size $\Delta t = 100$ seconds is prescribed for both compatible finite element methods.

Discretization properties					
Mixed method	# cells	Δx	Velocity unknowns	Depth unknowns	Total (millions)
$\mathrm{RT}_1 \times \mathrm{dP}_0$	327,680	≈ 43 km	491,520	327,680	0.8 M
$\mathrm{BDM}_2 \times \mathrm{dP}_1$			2,457,600	983,040	3.4 M

using test case 5 (flow past a mountain) from Williamson et al. (1992). We now revisit the same example and discuss the implementation details here. In particular, we demonstrate the advantage of using a hybridizable mixed method.

After discretizing in time and space using a semi-implicit scheme and Picard linearization, following Natale and Cotter (2017), we must solve a sequence of saddle point system at each time-step of the form:

$$\mathcal{A}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{M}_1 + \frac{\Delta t}{2}\boldsymbol{C} & -\frac{\Delta t}{2} g \boldsymbol{B}^T \\ \frac{\Delta t}{2} H \boldsymbol{B} & \boldsymbol{M}_2 \end{bmatrix} \begin{Bmatrix} \delta \boldsymbol{U}^k \\ \delta \boldsymbol{D}^k \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{R}_{\boldsymbol{u}}^k \\ \boldsymbol{R}_h^k \end{Bmatrix}, \tag{4.98}$$

where (4.98) is the result of linearizing the nonlinear equations (3.157)–(3.158) in Section 3.3.1 about a rest state with mean depth H. The linear updates $\delta \boldsymbol{U}^k$ and $\delta \boldsymbol{D}^k$ are sought in the mixed finite element spaces $U_h \times V_h \subset H(\mathrm{div}) \times L^2$, and update the nonlinear velocity and depth fields. Compatible pairings including the RT or BDM spaces such as $\mathrm{RT}_k \times \mathrm{dP}_{k-1}$ or $\mathrm{BDM}_k \times \mathrm{dP}_{k-1}$ fall within the set of compatible mixed spaces ideal for geophysical fluid simulations (see Section 2.4.2). In particular, the lowest-order RT method ($\mathrm{RTC}_1^f \times \mathrm{dQ}_0$) on a structured quadrilateral grid (such as the latitude-longitude grid used many operational dynamical cores) corresponds to the Arakawa C-grid finite difference discretization (Cotter and Shipton, 2012).

In staggered finite difference models, the standard approach for solving (4.98) is to neglect the Coriolis term $\boldsymbol{C}$ and eliminate the velocity unknown $\delta \boldsymbol{U}^k$ to obtain a discrete elliptic equation for $\delta \boldsymbol{D}^k$, where smoothers like Richardson iterations or relaxation methods are convergent. This is more problematic in the compatible finite element framework, since $\boldsymbol{M}_1$ has a dense inverse. Instead, we use the preconditioner described in Section 4.4.3 to form the equivalent hybridizable formulation, where both $\delta \boldsymbol{U}^k$ and $\delta \boldsymbol{D}^k$ are eliminated locally to produce a sparse elliptic equation for the Lagrange multipliers.

Profiling Williamson test case 5

The numerical procedure from Section 3.3.1 remains unchanged. Here, we focus on the linear solver for (4.98). Specifically, we profile the linear solver using two preconditioning strategies for a total of 25 time-steps, with a time-step size of $\Delta t = 100$ seconds. For this experiment, we consider two compatible finite element discretizations on a simplicial mesh using a lowest-order pairing $(\mathrm{RT}_1, \mathrm{dP}_0)$ and a higher-order pairing $(\mathrm{BDM}_2, \mathrm{dP}_1)$ for the velocity and depth spaces. The sphere mesh used in this section is generated from 7 refinements of an icosahedron, resulting in a triangulation consisting of 327,680 elements in total. The discretization information is summarized in Table 4.2.

At each time-step, we perform a fixed number of 4 Picard iterations as done by Shipton, Gibson, and Cotter (2018) and Melvin et al. (2019). We compare the overall simulation time using two different solver configurations for the implicit linear system. First, we use a flexible variant of GMRES [11] acting on the outer system with an approximate Schur complement preconditioner (via the `fieldsplit` option):

$$\mathcal{P}_{\mathrm{SC}} = \begin{bmatrix} \boldsymbol{I} & \frac{\Delta t}{2} g \widetilde{\boldsymbol{M}}_1^{-1} \boldsymbol{B}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widetilde{\boldsymbol{M}}_1^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \widetilde{\boldsymbol{S}}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\frac{\Delta t}{2} H \boldsymbol{B} \widetilde{\boldsymbol{M}}_1^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{4.99}$$

where $\widetilde{\boldsymbol{S}}$ is a symmetric[12] sparse operator, defined via a diagonal approximation to the matrix on the velocity space U_h:

$$\widetilde{\boldsymbol{S}} = \boldsymbol{M}_2 + \frac{\Delta t^2}{4} g H \boldsymbol{B} \mathrm{Diag}(\widetilde{\boldsymbol{M}}_1)^{-1} \boldsymbol{B}^T, \quad \widetilde{\boldsymbol{M}}_1 = \boldsymbol{M}_1 + \frac{\Delta t}{2} \boldsymbol{C}. \tag{4.100}$$

To control the reduction of the residual in the approximate Schur-complement system, we use GMRES configured with a relative tolerance of 10^{-8}. GMRES is accelerated using PETSc's native implementation of smoothed aggregation multigrid (GAMG) (Balay et al., 2016, §4.4.5). Following Adams et al. (2003), we employ block ILU-preconditioned Chebyshev iterations as the smoother in our AMG method. The velocity operator $\widetilde{\boldsymbol{M}}_1^{-1}$ in (4.99) is computed approximately using an incomplete LU factorization with zero in-fill (ILU(0)). Full solver configurations for this approach are shown in Listing 4.8.

Our second solver will use the Python preconditioner: `firedrake.HybridizationPC`. Rather than apply an outer Krylov method on the mixed system (4.98), we configure PETSc to *only* apply the preconditioner. The hybridization procedure replaces (4.98) with its hybridizable variant. The resulting three-field problem then has the form:

$$\begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}} & \boldsymbol{\mathcal{C}}^T \\ \boldsymbol{\mathcal{C}} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta \widehat{\boldsymbol{X}}^k \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \widehat{\boldsymbol{R}}^k_{\delta X} \\ \boldsymbol{0} \end{Bmatrix}, \tag{4.101}$$

where $\widehat{\boldsymbol{\mathcal{A}}}$ is the discontinuous operator coupling $\delta \widehat{\boldsymbol{X}}^k = \begin{Bmatrix} \delta \widehat{\boldsymbol{U}}^k & \delta \boldsymbol{D}^k \end{Bmatrix}^T$, $\boldsymbol{\mathcal{C}}$ is the matrix corresponding to the surface terms added after hybridization, and $\widehat{\boldsymbol{R}}^k_{\delta X} = \begin{Bmatrix} \widehat{\boldsymbol{R}}^k_{\boldsymbol{u}} & \boldsymbol{R}^k_h \end{Bmatrix}^T$ are the problem residuals. An exact Schur-complement factorization is performed on (4.101), using Slate to generate the local elimination kernels. We use a similar set of solver options for the inversion of $\widetilde{\boldsymbol{S}}$ in (4.99) to invert the Lagrange multiplier system. The increments $\delta \widehat{\boldsymbol{U}}^k$ and $\delta \boldsymbol{D}^k$ are recovered locally, using Slate-generated kernels. Once recovery is complete, $\delta \widehat{\boldsymbol{U}}^k$ is injected back into the $H(\mathrm{div})$ finite element space via $\delta \boldsymbol{U}^k \leftarrow \boldsymbol{\Pi}_{\mathrm{avg}} \delta \widehat{\boldsymbol{U}}^k$. Based on the discussion in Section 4.4.3, we apply:

$$\mathcal{P}_{\mathrm{hybrid}} = \boldsymbol{\Pi} \left(\begin{bmatrix} \boldsymbol{I} & -\widehat{\boldsymbol{\mathcal{A}}}^{-1} \boldsymbol{\mathcal{C}}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{\mathcal{A}}}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & S_\lambda^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{\mathcal{C}} \widehat{\boldsymbol{\mathcal{A}}}^{-1} & I \end{bmatrix} \right) \boldsymbol{\Pi}^T. \tag{4.102}$$

[11] We use a flexible version of GMRES on the outer system since we use an additional Krylov solver to iteratively invert the Schur-complement. This makes the application of the preconditioner *variable* (or nonlinear).

[12] The Coriolis matrix $\boldsymbol{C}$ has only values of zero along the matrix diagonal. Upon diagonalization, this gives $\mathrm{Diag}(\boldsymbol{M}_1 + \frac{\Delta t}{2} \boldsymbol{C}) = \mathrm{Diag}(\boldsymbol{M}_1)$, where $\boldsymbol{M}_1$ is the mass matrix on U_h.

TABLE 4.3: Preconditioner solve times for a 25-step run with $\Delta t = 100$s. These are cumulative times in each stage of the two preconditioners throughout the entire profile run. We display the average iteration count (rounded to the nearest integer) for both the outer and the inner Krylov solvers. The significant speedup when using hybridization is a direct result of eliminating the outer-most solver.

Preconditioner and solver details					
Mixed method	Preconditioner	t_{total} (s)	Avg. outer its.	Avg. inner its.	$\frac{t^{\text{SC}}_{\text{total}}}{t^{\text{hybrid.}}_{\text{total}}}$
$\text{RT}_1 \times \text{dP}_0$	approx. Schur.	15.137	2	8	3.413
	hybridization	4.434	None	2	
$\text{BDM}_2 \times \text{dP}_1$	approx. Schur.	300.101	4	9	5.556
	hybridization	54.013	None	6	

TABLE 4.4: Breakdown of the cost (average) of a single application of the preconditioned flexible GMRES algorithm and hybridization. Hybridization takes approximately the same time per iteration.

Preconditioner	Stage	$\text{RT}_1 \times \text{dP}_0$		$\text{BDM}_2 \times \text{dP}_1$	
		t_{stage} (s)	% t_{total}	t_{stage} (s)	% t_{total}
$\mathcal{P}_{\text{SC}}$	Schur solve	0.07592	**91.28 %**	0.78405	**93.53 %**
	Invert: $\widetilde{\boldsymbol{M}}_1$	0.00032	0.39 %	0.00678	0.81 %
	Apply inverse: $\widetilde{\boldsymbol{M}}_1^{-1}$	0.00041	0.49 %	0.00703	0.84 %
	gmres other	0.00652	7.84 %	0.04041	4.82 %
	Total	0.08317		0.83827	
$\mathcal{P}_{\text{hybrid}}$	Transfer: $\boldsymbol{R}_{\delta X} \to \widehat{\boldsymbol{R}}^k_{\delta X}$	0.00322	7.26 %	0.00597	1.10 %
	Eval.: $-\boldsymbol{\mathcal{C}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\widehat{\boldsymbol{R}}^k_{\delta X}$	0.00561	12.64 %	0.12308	22.79 %
	Invert: $\boldsymbol{S}_\lambda$	0.02289	**51.63 %**	0.28336	**52.46 %**
	Back sub.	0.00986	22.23 %	0.12220	22.62 %
	Projection: $\boldsymbol{\Pi}_{\text{avg}}\delta\widehat{\boldsymbol{U}}^k$	0.00264	5.96 %	0.00516	0.96 %
	Total	0.04434		0.54013	

To solve for the Lagrange multipliers, we invert $\boldsymbol{S}_\lambda$ using GMRES on the system due to the asymmetry of the Coriolis term. We accelerate GMRES using smoothed aggregation multigrid (GAMG) with ILU-preconditioned Richardson relaxation. Listing 4.9 shows the options configuration for the hybridization solver.

Table 4.3 displays a summary of our findings. The advantages of a hybridizable method versus a mixed method are more clearly realized in this experiment. When using hybridization, we observe a significant reduction in time spent in the implicit solver compared to the approximate Schur-complement approach. This is primarily because we have reduced the number of "outer" iterations to zero; the hybridization preconditioner is performing an *exact* factorization of the global hybridizable system. This is empirically supported when considering per-application solve times. The values reported in Table 4.4 show the average cost of a single outer GMRES iteration (which includes the application of $\mathcal{P}_{\text{SC}}$) and a single application of $\mathcal{P}_{\text{hybrid}}$. Hybridization and the approximate Schur complement preconditioner are comparable in terms of average execution time, with hybridization being slightly faster. This further demonstrates that the primary cause for the longer execution time of the latter is directly related to the additional outer iterations induced from using an approximate factorization. In terms of over all time-to-solution, the hybridizable

LISTING 4.8: Solver options for the approximate Schur-complement preconditioner in the rotating shallow water test case.

```
-ksp_type fgmres
-ksp_rtol 1e-8
-pc_type fieldsplit
-pc_fieldsplit_type full
-pc_fieldsplit_schur_precondition selfp

# velocity block solver
-fieldsplit_0_ksp_type preonly
-fieldsplit_0_pc_type bjacobi
-fieldsplit_0_sub_pc_type ilu

# Schur-complement block solver
-fieldsplit_1_ksp_type gmres
-fieldsplit_1_ksp_rtol 1e-8
-fieldsplit_1_pc_type gamg
-fieldsplit_1_pc_gamg_reuse_interpolation True
-fieldsplit_1_pc_gamg_sym_graph True
-fieldsplit_1_mg_levels_ksp_type chebyshev
-fieldsplit_1_mg_levels_ksp_max_it 2
-fieldsplit_1_mg_levels_pc_type bjacobi
-fieldsplit_1_mg_levels_sub_pc_type ilu
```

LISTING 4.9: Solver options for the hybridization preconditioner in the rotating shallow water test case.

```
-ksp_type preonly
-mat_type matfree
-pc_type python
-pc_python_type firedrake.HybridizationPC

# solver for the Lagrange multipliers
-hybridization_ksp_type gmres
-hybridization_ksp_max_it 100
-hybridization_ksp_rtol 1e-8
-hybridization_pc_type gamg
-hybridization_pc_gamg_reuse_interpolation True
-hybridization_pc_gamg_sym_graph True
-hybridization_mg_levels_ksp_type richardson
-hybridization_mg_levels_ksp_max_it 2
-hybridization_mg_levels_pc_type bjacobi
-hybridization_mg_levels_sub_pc_type ilu
```

methods are clearly ahead of the original mixed methods.

We also measure the relative reductions in the problem residual of the linear system (4.98). Our hybridization preconditioner reduces the residual by a factor of 10^8 on average, which coincides with the specified relative tolerance for the Krylov method on the trace system. In other words, the reduction in the residual for the trace system translates to an overall reduction in the residual for the mixed system by the same factor. We refer the reader to Shipton, Gibson, and Cotter (2018) and Wimmer, Cotter, and Bauer (2019) for further demonstrations of shallow water test cases featuring the use of the hybridization preconditioner described in Section 4.4.3.

4.5.3 Rotating linear gravity wave model

As a final example, we consider the simplified atmospheric model obtained from a linearization of the compressible Boussinesq equations in a rotating domain, as previously detailed in Section 3.4.2:

$$\frac{\partial \boldsymbol{u}}{\partial t} + f\hat{\boldsymbol{k}} \times \boldsymbol{u} = -\nabla p + b\hat{\boldsymbol{k}}, \tag{4.103}$$

$$\frac{\partial p}{\partial t} = -c^2 \nabla \cdot \boldsymbol{u}, \tag{4.104}$$

$$\frac{\partial b}{\partial t} = -N^2 \boldsymbol{u} \cdot \hat{\boldsymbol{k}}, \tag{4.105}$$

where $\boldsymbol{u}$ is the fluid velocity, p the pressure, b is the buoyancy, f the Coriolis parameter, c is the speed of sound ($c \approx 343\text{ms}^{-1}$), and N is the buoyancy frequency ($N \approx 0.01\text{s}^{-1}$). Equations (4.103)–(4.105) permit fast-moving acoustic waves driven by perturbations in b. We solve these equations subject to the rigid-lid condition $\boldsymbol{u} \cdot \boldsymbol{n} = 0$ on all boundaries.

Our domain $\Omega = \mathcal{S}(R) \times [0, H_\Omega]$ is a spherical annulus, with the mesh $\mathcal{T}_h$ constructed from a horizontal "base" mesh of the surface of a sphere with radius R (using a piece-wise cubic approximation of the surface), $\mathcal{S}(R)$, extruded upwards by a height H_Ω. The vertical discretization is a structured one-dimensional grid with uniform height. We consider two kinds of meshes: one obtained by extruding an icosahedral sphere mesh (triangulation), and another from a cubed sphere.

Since our mesh has a natural tensor-product structure, we construct suitable finite element spaces constructed by taking the tensor product of a horizontal space with a vertical space. To ensure our discretization is "compatible," we use the one- and two-dimensional de-Rham complexes: $V_h^0 \xrightarrow{\partial_z} V_h^1$ and $U_h^0 \xrightarrow{\nabla^\perp} U_h^1 \xrightarrow{\nabla\cdot} U_h^2$. We can then construct the three-dimensional complex: $W_h^0 \xrightarrow{\nabla} W_h^1 \xrightarrow{\nabla\times} W_h^2 \xrightarrow{\nabla\cdot} W_h^3$, where the resulting tensor product spaces are defined on prismatic cells:

$$W_h^0 = U_h^0 \otimes V_h^0, \tag{4.106}$$

$$W_h^1 = \texttt{HCurl}(U_h^1 \otimes V_h^0) \oplus \texttt{HCurl}(U_h^0 \otimes V_h^1), \tag{4.107}$$

$$W_h^2 = \texttt{HDiv}(U_h^1 \otimes V_h^1) \oplus \texttt{HDiv}(U_h^2 \otimes V_h^0), \tag{4.108}$$

$$W_h^3 = U_h^2 \otimes V_h^1. \tag{4.109}$$

Each discretization used in this section is constructed from more familiar finite element families, shown in Table 4.5. See Section 2.4.4 for a detailed review on tensor product compatible finite elements.

Implicit solver strategy

We now quickly summarize the solution strategy we outlined in Section 3.4.2 for solving (4.103)–(4.105). The compatible finite element discretization seeks the linear perturbation unknowns in the following spaces:

$$\delta \boldsymbol{u}_h \in \mathring{W}_h^2, \quad \delta p_h \in W_h^3, \quad \delta b_h \in W_h^b := U_h^2 \otimes V_h^0, \tag{4.110}$$

TABLE 4.5: Vertical and horizontal spaces for the three-dimensional compatible finite element discretization of the linear Boussinesq model. The RT_k and BDFM_{k+1} methods are constructed on triangular prism elements, while the RTCF_k method is defined on extruded quadrilateral elements.

	Compatible finite element spaces				
Method	V_h^0	V_h^1	U_h^0	U_h^1	U_h^2
RT_k	P_k	dP_{k-1}	$\mathrm{P}_k(\triangle)$	$\mathrm{RT}_k(\triangle)$	$\mathrm{dP}_{k-1}(\triangle)$
BDFM_{k+1}	P_{k+1}	dP_k	$\mathrm{P}_{k+1}(\triangle)$	$\mathrm{BDFM}_{k+1}(\triangle)$	$\mathrm{dP}_k(\triangle)$
RTCF_k	P_k	dP_{k-1}	$\mathrm{Q}_k(\square)$	$\mathrm{RTC}_k^f(\square)$	$\mathrm{dQ}_{k-1}(\square)$

where $\mathring{W}_h^2$ is the subspace of W_h^2 satisfying the no-slip condition on the boundary of the domain $\partial\Omega$, and the unknowns in (4.110) satisfy

$$\boldsymbol{u}_h^n = \boldsymbol{u}_h^{n-1} + \delta\boldsymbol{u}_h, \quad p_h^n = p_h^{n-1} + \delta p_h, \quad b_h^n = b_h^{n-1} + \delta b_h, \tag{4.111}$$

where the superscript n denotes fields at a given time-step n. Then we arrive at a mixed problem for $\delta\boldsymbol{u}_h$ and δp_h by discretizing (4.103)–(4.105) in time using the implicit midpoint rule, eliminating the buoyancy, followed by discretizing in space: find $(\delta\boldsymbol{u}_h, \delta p_h) \in \mathring{W}_h^2 \times W_h^3$ satisfying

$$\begin{aligned} &\int_{\mathcal{T}_h} \boldsymbol{w}\cdot\delta\boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2}\int_{\mathcal{T}_h} \boldsymbol{w}\cdot f\left(\hat{\boldsymbol{k}}\times\delta\boldsymbol{u}_h\right)\mathrm{d}\boldsymbol{x} \\ &+N^2\frac{\Delta t^2}{4}\int_{\mathcal{T}_h}\hat{\boldsymbol{k}}\cdot\boldsymbol{w}\hat{\boldsymbol{k}}\cdot\delta\boldsymbol{u}_h\,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2}\int_{\mathcal{T}_h}\delta p_h\nabla_h\cdot\boldsymbol{w}\,\mathrm{d}\boldsymbol{x} = \widetilde{R}_{\boldsymbol{u}}[\boldsymbol{w}], \end{aligned} \tag{4.112}$$

$$\int_{\mathcal{T}_h}\phi\delta p_h\,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2}c^2\int_{\mathcal{T}_h}\phi\nabla_h\cdot\delta\boldsymbol{u}_h\,\mathrm{d}\boldsymbol{x} = R_p[\phi], \tag{4.113}$$

for all $(\boldsymbol{w},\phi) \in \mathring{W}_h^2 \times W_h^3$, where $\widetilde{R}_{\boldsymbol{u}}[\boldsymbol{w}]$ and $R_p[\phi]$ are as defined in (3.228)–(3.229). Once (4.112)–(4.113) are solved, δb_h is determined by solving the linear variational problem: find $\delta b_h \in W_h^b$ such that

$$\int_{\mathcal{T}_h}\mu\delta b_h\,\mathrm{d}\boldsymbol{x} = \underbrace{-\Delta t N^2\int_{\mathcal{T}_h}\mu\boldsymbol{u}_h^{n-1}\cdot\hat{\boldsymbol{k}}\,\mathrm{d}\boldsymbol{x}}_{=:R_b[\mu]} - \frac{\Delta t}{2}N^2\int_{\mathcal{T}_h}\mu\delta\boldsymbol{u}_h\cdot\hat{\boldsymbol{k}}\,\mathrm{d}\boldsymbol{x}, \tag{4.114}$$

for all $\mu \in W_h^b$.

We arrive at two matrix equations by expanding the test and trial functions in terms of their finite element bases. The problem in (4.112)–(4.113) becomes the saddle-point system:

$$\boldsymbol{\mathcal{A}}\begin{Bmatrix}\delta\boldsymbol{U}\\ \delta\boldsymbol{P}\end{Bmatrix} = \begin{bmatrix}\boldsymbol{A} & -\frac{\Delta t}{2}\boldsymbol{D}^T\\ \frac{\Delta t}{2}c^2\boldsymbol{D} & \boldsymbol{M}_3\end{bmatrix}\begin{Bmatrix}\delta\boldsymbol{U}\\ \delta\boldsymbol{P}\end{Bmatrix} = \begin{Bmatrix}\widetilde{\boldsymbol{R}}_u\\ \boldsymbol{R}_p\end{Bmatrix}, \tag{4.115}$$

where $\delta\boldsymbol{U}$ and $\delta\boldsymbol{P}$ are coefficient vectors,

$$\boldsymbol{A} = \boldsymbol{M}_2 + \frac{\Delta t}{2}\boldsymbol{C} + N^2\frac{\Delta t^2}{4}\boldsymbol{M}_2^v, \quad (\boldsymbol{M}_2)_{ij} = \int_{\mathcal{T}_h}\boldsymbol{\Psi}_j\cdot\boldsymbol{\Psi}_i\,\mathrm{d}\boldsymbol{x}, \tag{4.116}$$

$$(\boldsymbol{C})_{ij} = \int_{\mathcal{T}_h}\boldsymbol{\Psi}_j\cdot f\left(\hat{\boldsymbol{k}}\times\boldsymbol{\Psi}_i\right)\mathrm{d}\boldsymbol{x}, \quad (\boldsymbol{M}_2^v)_{ij} = \int_{\mathcal{T}_h}\hat{\boldsymbol{k}}\cdot\boldsymbol{\Psi}_j\hat{\boldsymbol{k}}\cdot\boldsymbol{\Psi}_i\,\mathrm{d}\boldsymbol{x}, \tag{4.117}$$

$$(\boldsymbol{D})_{ij} = \int_{\mathcal{T}_h}\Phi_j\nabla_h\cdot\boldsymbol{\Psi}_i\,\mathrm{d}\boldsymbol{x}, \quad (\boldsymbol{M}_3)_{ij} = \int_{\mathcal{T}_h}\Phi_j\,\Phi_i\,\mathrm{d}\boldsymbol{x}, \tag{4.118}$$

and $(\widetilde{\boldsymbol{R}}_u)_j = \widetilde{R}_{\boldsymbol{u}}[\Phi_j]$, $(\boldsymbol{R}_p)_j = R_p[\Phi_j]$, for basis functions $(\boldsymbol{\Psi}_i, \Phi_j) \in \mathring{W}_h^2 \times W_h^3$.

Similarly, we have the linear system for the buoyancy coefficients $\delta\boldsymbol{B}$

$$\boldsymbol{M}_b \delta\boldsymbol{B} = \boldsymbol{R}_b - \frac{\Delta t}{2} N^2 \boldsymbol{Q} \delta\boldsymbol{U}, \tag{4.119}$$

where $(\boldsymbol{M}_b)_{ij} = \int_{\mathcal{T}_h} \eta_j \eta_i \,\mathrm{d}\boldsymbol{x}$, $(\boldsymbol{Q})_{ij} = \int_{\mathcal{T}_h} \eta_j \hat{\boldsymbol{k}} \cdot \boldsymbol{\Psi}_i \,\mathrm{d}\boldsymbol{x}$, and $(\boldsymbol{R}_b)_j = R_b[\eta_j]$ for basis functions $\eta_j \in W_h^b$. The system (4.119) can be solved easily, since $\boldsymbol{M}_b$ is only coupled within columns, with a condition number independent of the mesh resolution and time-step. Therefore $\boldsymbol{M}_b$ can be swiftly inverted with a few ILU-preconditioned conjugate gradient iterations.

Preconditioning the mixed velocity pressure system

The primary difficulty is in finding robust solvers for (4.115). This was studied by Mitchell and Müller (2016) within the context of developing a robust preconditioner to withstand fast-moving acoustic waves. This is critical, as the governing equations support fast waves driven by perturbations to the buoyancy field. However, the implicit treatment of the Coriolis term was not taken into account. We shall consider two preconditioning strategies.

The first strategy follows from Mitchell and Müller (2016). As with the rotating shallow water system in Section 4.5.2, we can construct a preconditioner based on the Schur-complement factorization of $\boldsymbol{\mathcal{A}}$ in (4.115):

$$\boldsymbol{\mathcal{A}}^{-1} = \begin{bmatrix} \boldsymbol{I} & \frac{\Delta t}{2}\boldsymbol{A}^{-1}\boldsymbol{D}^T \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -c^2\frac{\Delta t}{2}\boldsymbol{D}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{bmatrix}, \tag{4.120}$$

where $\boldsymbol{H}$ is the dense pressure Helmholtz operator. Because we have chosen to include the Coriolis term, the operator $\boldsymbol{H}$ is non-symmetric, and we require a sparse approximation to

$$\boldsymbol{H} = \boldsymbol{M}_3 + c^2\frac{\Delta t^2}{4}\boldsymbol{D}\left(\boldsymbol{M}_2 + \frac{\Delta t}{2}\boldsymbol{C} + N^2\frac{\Delta t^2}{4}\boldsymbol{M}_2^v\right)^{-1}\boldsymbol{D}^T. \tag{4.121}$$

As Δt increases, the contribution of $\boldsymbol{C}$ becomes more prominent in $\boldsymbol{H}$, making sparse approximations of $\boldsymbol{H}$ more challenging. We elaborate on this further as we present the results of our second solver.

Our second strategy utilizes the `firedrake.HybridizationPC` preconditioner, which directly forms the hybridizable problem as presented in Section 3.4.2 (equations (3.236)–(3.238)). The space of traces is generated on the faces of triangular-prisms and extruded quadrilaterals, with appropriate polynomial degrees such that every trace function lies in the same polynomial space as the normal components of W_h^2 velocity fields.

We locally eliminate $\delta\boldsymbol{U}$ and $\delta\boldsymbol{P}$ after hybridization, producing the condensed system for the traces:

$$\boldsymbol{H}_\partial \boldsymbol{\Lambda} = \boldsymbol{E}, \quad \boldsymbol{H}_\partial = \boldsymbol{\mathcal{C}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\boldsymbol{\mathcal{C}}^T, \quad \boldsymbol{E} = \boldsymbol{\mathcal{C}}\widehat{\boldsymbol{\mathcal{A}}}^{-1}\begin{Bmatrix} \widehat{\boldsymbol{R}}_{\boldsymbol{u}} \\ \boldsymbol{R}_p \end{Bmatrix}, \tag{4.122}$$

where $\widehat{\mathcal{A}}$ is the result of rendering the mixed operator $\mathcal{A}$ block-diagonal, $\mathcal{C}$ corresponds to the surface terms added by the hybridization preconditioner, and $\boldsymbol{H}_\partial$ is the statically condensed trace operator to be inverted.

Solving the non-symmetric trace system

Our approach combines several ideas from both multigrid for non-symmetric operators and iterative methods employed by operational weather models. We first consider the choice of Krylov subspace method. Since $\boldsymbol{H}_\partial$ is non-symmetric, we employ the generalized conjugate residual (GCR) method (Eisenstat, Elman, and Schultz, 1983), which is also the Krylov method of choice in Thomas et al. (2003). It is also a flexible Krylov method, meaning that GCR allows for variable/non-linear preconditioning (for example, having another Krylov method as part of a preconditioner). We then precondition GCR using smoothed aggregation multigrid (GAMG).

Much of the convergence theory on multigrid relies on the symmetry of the operator. Some analysis has been done for non-symmetric and indefinite systems; see, for example: Mandel (1986), Bramble, Pasciak, and Xu (1988), Bramble, Kwak, and Pasciak (1994), and Elman, Ernst, and O'Leary (2001). In principle, an AMG preconditioner can be applied to a general non-symmetric matrix, but it requires a careful choice of smoother. In particular, a smoother needs to damp out the high-frequency eigenmodes in order for the coarse-grid correction to be successful. For our choice of smoother, we consider a variation of Polynomial-based smoothers, such as Chebyshev methods (Adams et al., 2003), using the GMRES algorithm.

GMRES smoothing: A Krylov method is generally not thought of as a smoother due to the fact that they target the *whole* spectrum of eigenmodes. This will naturally include some low-frequency modes that the coarse-grid correction would have handled otherwise. However, they have been shown to work well for non-symmetric or indefinite operators (Elman, Ernst, and O'Leary, 2001; Lin, Shadid, and Tsuji, 2019).

Given a matrix system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and a residual vector $\boldsymbol{r}^0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0$, where $\boldsymbol{x}^0$ is an initial iterate for the GMRES algorithm, the Krylov subspace of order n is

$$\mathcal{K}_n = \text{Span}\left\{\boldsymbol{r}^0, \boldsymbol{A}\boldsymbol{r}^0, \cdots, \boldsymbol{A}^{n-1}\boldsymbol{r}^0\right\}. \tag{4.123}$$

By construction, we have $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \cdots \subset \mathcal{K}_n$. For a fixed number of GMRES iterations k, the GMRES iterate $\boldsymbol{x}^k$ is the solution of the minimization problem:

$$\min_{\boldsymbol{x}\in\{\boldsymbol{x}^0\}+\mathcal{K}_k} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2, \tag{4.124}$$

where $\|\cdot\|_2$ is the Euclidean 2-norm. Then the GMRES iterate, by definition, has the form

$$\boldsymbol{x} = \boldsymbol{x}^0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{A}^j \boldsymbol{r}^0 = \boldsymbol{x}^0 + \sum_{j=0}^{k-1} \alpha_j \boldsymbol{A}^j \left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0\right) = \sum_{j=0}^{k-1} \beta_j \boldsymbol{A}^j \boldsymbol{b} + \sum_{j=0}^{k} \beta_j \boldsymbol{A}^j \boldsymbol{x}^0, \tag{4.125}$$

for some coefficients α_j, β_j. Then we define the iteration matrix for the smoother as $\boldsymbol{M}_k^{\text{gmres}}[\boldsymbol{A}] = \sum_{j=0}^{k} \beta_j \boldsymbol{A}^j = \mathcal{P}_k^{\text{gmres}}(\boldsymbol{A})$, where $\mathcal{P}_k^{\text{gmres}}(\boldsymbol{A})$ is a matrix-valued polynomial of degree k generated by the GMRES algorithm (see also Saad (2003, §6.5)).

LISTING 4.10: Solver options for the linear gravity wave hybridization preconditioner.

```
-ksp_type preonly
-mat_type matfree
-pmat_type matfree
-pc_type python
-pc_python_type firedrake.HybridizationPC

# solver for the Lagrange multipliers
-hybridization_ksp_type gcr
-hybridization_ksp_max_it 100
-hybridization_ksp_rtol 1e-5
-hybridization_pc_type gamg
-hybridization_pc_gamg_reuse_interpolation True
-hybridization_pc_gamg_sym_graph True
-hybridization_mg_levels_ksp_type gmres
-hybridization_mg_levels_ksp_max_it 5
-hybridization_mg_levels_pc_type bjacobi
-hybridization_mg_levels_sub_pc_type ilu
```

We can extend this further by considering a preconditioned GMRES procedure. Discretizations on thin-domains (small aspect ratio domains) require the use of line relaxation methods (Saad, 2003, §4.1.1) for their efficient solution (Börm and Hiptmair, 2001; Buckeridge and Scheichl, 2010; Müller and Scheichl, 2014; Dedner, Müller, and Scheichl, 2015). In our context, we require line relaxation in the extrusion direction; our problem is defined on a spherical annulus, therefore we require relaxation in the directions of the vertical columns of the mesh. For the finest level, block ILU actually produces a line relaxation method for (4.122) when the trace variable nodes are numbered column-wise (this is known as line-ILU (Shapira, 2008, §11.6)). This is the case in our implementation, since Firedrake employs this numbering approach on extruded meshes (Bercea et al., 2016). Therefore, we shall use block ILU to precondition GMRES in our smoothing procedure. The resulting smoother is summarized compactly as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{M}_{k-1}^{\text{gmres}}[\hat{\boldsymbol{P}}^{-1}\boldsymbol{A}]\boldsymbol{b} + \boldsymbol{M}_{k}^{\text{gmres}}[\hat{\boldsymbol{P}}^{-1}\boldsymbol{A}]\boldsymbol{x}, \quad k \geq 1, \tag{4.126}$$

where $\tilde{\boldsymbol{x}}$ is the smoothed result and $\hat{\boldsymbol{P}}$ is an block ILU factorization. This comes directly from (4.125), with $\hat{\boldsymbol{P}}^{-1}\boldsymbol{A}$ being used to build the Krylov subspace (4.123).

In practice, we only apply a few iterations ($k = 5$) of non-restarted preconditioned GMRES. The outer GCR method is set to terminate when the residual for the trace system is reduced by a factor of 10^5. Complete solver configurations for the preconditioned GCR method and configuration of the AMG precondition are shown in Listing 4.10.

Problem setup

Our setup closely resembles the gravity wave test of Skamarock and Klemp (1994) extended to a spherical annulus. The domain is characterized by the planet radius $R = 6371\text{km}$, a scaling factor X, and the height of the atmospheric lid H_Ω: $\Omega = \mathcal{S}(R/X) \times [0, H_\Omega]$. We initialize the velocity in a simple solid-body rotation: $\boldsymbol{u}(\boldsymbol{x}, 0) = \frac{u_0}{R}\begin{Bmatrix} -y & x & 0 \end{Bmatrix}^T$, where $u_0 = 20\text{ms}^{-1}$. A localized buoyancy anomaly is added to a background buoyancy profile $b(\boldsymbol{x}, 0) = N^2 z + \delta b(\boldsymbol{x}, 0)$ along the equator,

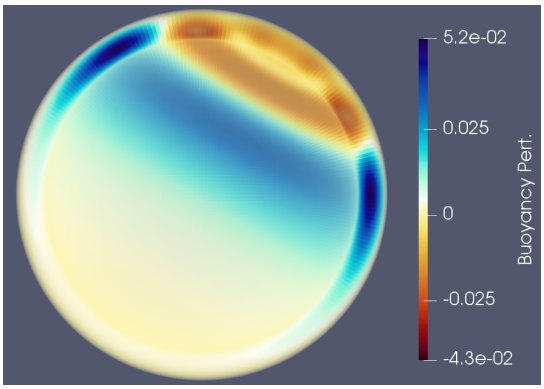


FIGURE 4.5: Buoyancy perturbation ($y-z$ cross-section) after $t = 3600$ seconds from a simple gravity wave test ($\Delta t = 100$ seconds). The equations are discretized using the lowest-order RTCF_1 method, with 24,576 triangular cells in the horizontal and 64 extrusion levels. The velocity-pressure system is solved using hybridization.

where:

$$\delta b(\boldsymbol{x}, 0) = \frac{d^2}{d^2 + r^2} \sin\left(\frac{\pi z}{H_\Omega}\right), \tag{4.127}$$

$d = 5000$m is the width parameter for the perturbation, and H_Ω is the height of the atmospheric lid. The "great circle distance" r is defined via

$$r = \frac{R}{X} \arccos\left(\sin(\phi_c)\sin(\phi) + \cos(\phi_c)\cos(\phi)\cos(\lambda - \lambda_c)\right), \tag{4.128}$$

where $\phi_c = 0$, $\lambda_c = 2\pi/3$ is the latitudal and longitudinal center-point of the perturbation respectively. Note that for convenience, we have chosen to express (4.128) in geographic coordinates (as we have done for certain expressions in Section 3.3.2). The actual Firedrake implementation uses a complete Cartesian formulation. The pressure field is constant $p(\boldsymbol{x}, 0) = 0$ and unperturbed.

A Coriolis term is introduced as a function of the Cartesian coordinate z, and is constant along lines of latitude (f-plane): $f = 2\Omega_r \frac{z}{R}\hat{\boldsymbol{z}}$, with $\hat{\boldsymbol{z}}$ denoting the vertical normal and $\Omega_r = 7.292 \times 10^{-5}\mathrm{s}^{-1}$ is the angular rotation rate. A gravity wave test using our solution strategy and hybridization preconditioner is illustrated in Figure 4.5 for a problem on a condensed Earth (radius scaled down by a factor of $X = 125$) and $H_\Omega = 10$km, as presented by (Ullrich et al., 2012). A complete Firedrake implementation (without Coriolis) is also presented in Gibson et al. (2019a, §5.2.2) using the same initialization conditions and problem domain.

Time-step robustness with implicit Coriolis

A desired property of the implicit linear solver is robustness with respect to the time-step size. In particular, it is desirable for the execution time to remain constant across a wide range of Δt. As Δt increases, the conditioning of the elliptic operator becomes worse. Therefore, iterative solvers need to work much harder to reduce the residual down to a specified tolerance.

We repeat a similar study to that presented in Mitchell and Müller (2016). We fix the resolution of the problem and run the solver over a range of Δt. We measure this by adjusting the horizontal acoustic Courant number $\lambda_C = c\frac{\Delta t}{\Delta x}$, where c is the

TABLE 4.6: Grid set up and discretizations for the acoustic Courant number study. The total unknowns (velocity and pressure) and hybridized unknowns (broken velocity, pressure, and trace) are shown in the last two columns (millions). The vertical resolution is fixed across all discretizations.

Discretizations and grid information						
Method	# horiz. cells	# layers	Δx	Δz	*U-P* dofs	Hybrid. dofs
RT_1	81,920	85	86 km	1 km	24.5 M	59.3 M
RT_2	5,120	85	346 km	1 km	9.6 M	17.4 M
$BDFM_2$	5,120	85	346 km	1 km	10.5 M	18.3 M
$RTCF_1$	98,304	85	78 km	1 km	33.5 M	83.7 M
$RTCF_2$	6,144	85	312 km	1 km	16.7 M	29.3 M

speed of sound and Δx is the horizontal resolution. We remark that the range of Courant numbers used in this paper exceeds what is typical in operational forecast settings, where λ_C is typically between $\mathcal{O}(2)$–$\mathcal{O}(10)$.[13] The grid setup mirrors that of actual global weather models; we extrude a spherical mesh of the Earth upwards to a height $H_\Omega = 85$ km. We also set the scaling factor $X = 1$, which produces a mesh of the entire global atmosphere for the Earth. The set up for the different discretizations, which includes the total number of degrees of freedom for the velocity-pressure and hybridizable systems, are shown in Table 4.6.

Let us now direct our attention to the original system, where we have the Schur-complement factorization given in (4.120). It was shown by Mitchell and Müller (2016) that using a sparse approximation of the pressure Schur-complement of the form:

$$\widetilde{\boldsymbol{H}} = \boldsymbol{M}_3 + c^2 \frac{\Delta t^2}{4} \boldsymbol{D} \mathrm{Diag}(\boldsymbol{A})^{-1} \boldsymbol{D}^T \tag{4.129}$$

served as a good preconditioner, leading to a system that was amenable to multigrid methods and resulted in a solver with λ_C-independent convergence. However, when the Coriolis term is included, this is no longer the case: the diagonal approximation $\mathrm{Diag}(\boldsymbol{A})$ becomes worse with increasing λ_C.

To demonstrate this, we solve the gravity wave system on a low-resolution grid (10km lid, 10 vertical levels, maintaining the same cell aspect ratios as in Table 4.6) using the Schur-complement factorization (4.120). LU factorizations are applied to invert both $\boldsymbol{A}^{-1}$ and $\widetilde{\boldsymbol{H}}^{-1}$. Inverting the Schur-complement $\boldsymbol{H}^{-1}$ is done using preconditioned GMRES iterations, and a flexible GMRES algorithm is used on the full velocity-pressure system. If $\widetilde{\boldsymbol{H}}^{-1}$ is a good approximation to $\boldsymbol{H}^{-1}$, then we should see low iteration counts in the Schur-complement solve. Figure 4.6 shows the results of this study for a range of Courant numbers.

For the lower-order methods, the number of iterations to invert $\boldsymbol{H}$ grow slowly but remain under control. Increasing the approximation degree by one results in degraded performance. As Δt increases, the number of Krylov iterations needed to invert the system to a relative tolerance of 10^{-5} grows rapidly. It is clear that this sparse approximation is not robust against Courant number. This can be explained by the fact that diagonalizing the velocity operator fails to take into account the effects of the Coriolis term (which appear in off-diagonal positions in the operator).

[13] Confirmed in personal conversations with UK Met Office dynamical core developers.

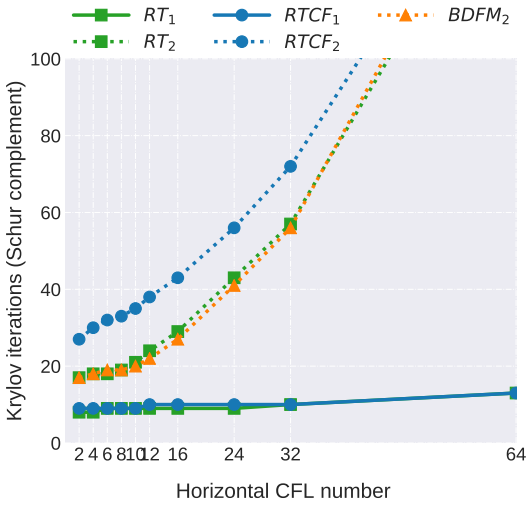
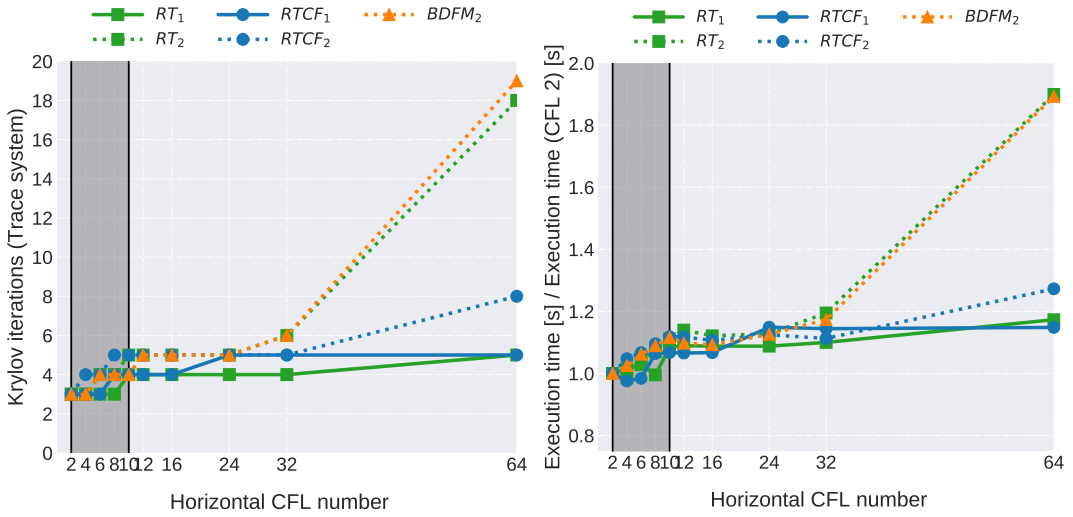


FIGURE 4.6: Number of Krylov iterations to invert the Helmholtz system using an exact application of $\widetilde{\boldsymbol{H}}^{-1}$ as a preconditioner. While the lowest-order methods grow slowly over the Courant number range, the higher-order (by only one approximation order) methods quickly degrade and diverge after the critical range $\lambda_C = \mathcal{O}(2)$–$\mathcal{O}(10)$. At $\lambda_C > 32$, the solvers take over 150 iterations.

Even if one were to use traditional mass-lumping (row-wise sums), the Coriolis effects are effectively cancelled out due to asymmetry.

Hybridization avoids this problem entirely: we always construct an exact Schur complement, and only have to worry about solving the trace system (4.122). We now show that this approach is much more robust against changes in Δt (hence against the horizontal Courant number). We use the same workstation as for the three-dimensional CG/HDG problem in Section 4.5.1 (executed with a total of 40 MPI processes). Figure 4.7 shows the parameter test for all the discretizations described in Table 4.6. We see that, in terms of total number of GCR iterations needed to invert the trace system, hybridization is far more controlled as Courant number increases. They largely remain constant throughout the entire parameter range, only varying by an iteration or two. It is not until after $\lambda_C > 32$ that we begin to see a larger jump in the number of GCR iterations. This is expected, since the Coriolis operator causes the problem to become singular for very large Courant numbers. However, unlike with the approximate Schur-complement solver, iteration counts are still under control. In particular, each method (lowest and higher order) remains constant throughout the critical range (shaded in gray in Figures 4.7A and 4.7B).

In Figure 4.7B, we display the ratio of execution time and the time-to-solution at the lowest Courant number of two. We perform this normalization to better compare the lower and higher order methods (and discretizations on triangular prisms vs extruded quadrilaterals). The calculation of the ratios include the time needed to eliminate and reconstruct the hybridized velocity/pressure variables. The fact that the hybridization solver remains close to one demonstrates that the entire solution procedure is largely λ_C-independent until around $\lambda_C = 32$. The overall trend is largely the same as what is observed in Figure 4.7A. This is due to our hybridization approach being solver dominated, with local operations like forward elimination together with local recovery taking approximately 1/3 of the total execution time for each method (similar to what is already presented in Section 4.5.1).



(A) GCR iterations vs Courant number.

(B) Work vs Courant number.

FIGURE 4.7: Courant number parameter test run on a fully-loaded compute node. Both figures display the hybridized solver for each discretization, described in Table 4.6. The left figure (4.7A) displays total iteration count (preconditioned GCR) to solve the trace system to a relative tolerance of 10^{-5}. The right figure (4.7B) displays the relative work of each solver. Figure 4.7B takes into account not just the time-to-solution of the trace solver, but also the time required to forward eliminate and locally recover the velocity and pressure.

Implicitly treating the Coriolis term has been discussed for semi-implicit discretizations of large-scale geophysical flows (Temperton, 1997; Côté and Staniforth, 1988; Cullen, 2001; Nechaev and Yaremchuk, 2004). Incorporating the Coriolis term in finite element discretizations is difficult, as this makes the challenge of find robust sparse approximation of the resulting elliptic operator even more difficult. Hybridization shows promise here, as we no longer require the inversion of a dense elliptic system. Instead, hybridization allows for the assembly of an operator that both captures the effects of rotation and results in a sparse linear system. In fact, the hybridization process mimics standard staggered finite difference elimination procedures used in many global circulation models. In other words, hybridization is the finite element analogue of point-wise elimination strategies in finite difference codes. More rigorous analysis on the spectral properties of $\boldsymbol{H}_\partial$ in (4.122) is a subject of on-going interest.

4.6 Chapter summary

We have presented Slate, and shown how this language can be used to create concise mathematical representations of localized linear algebra on the tensors corresponding to finite element forms. We have shown how this DSL can be used in tandem with UFL in Firedrake to implement solution approaches making use of automated code generation for static condensation, hybridization, and localized post-processing. Setup and configuration is done at runtime, allowing one to switch in different discretizations at will. In particular, this framework alleviates much of the difficulty in implementing such methods within intricate numerical code, and paves the way for future low-level optimizations. In this way, the framework in this paper

can be used to help enable the rapid development and exploration of new hybridization and static condensation techniques for a wide class of problems. We remark here that the reduction of global matrices via element-wise algebraic static condensation, as described in Guyan (1965) and Irons (1965) is also possible using Slate, including other more general static condensation procedures outside the context of hybridization. A recent example of this was Slate's use for statically condensing a continuous Galerkin discretization (Kirby and Mitchell, 2018a).

Our approach to preconditioner design revolves around its composable nature, in that these Slate-based implementations can be seamlessly incorporated into complicated solution schemes. In particular, there is current research in the design of dynamical cores for numerical weather prediction using implementations of hybridization and static condensation with Slate (Bauer and Cotter, 2018; Shipton, Gibson, and Cotter, 2018; Wimmer, Cotter, and Bauer, 2019; Gibson et al., 2019a). The performance of such methods for geophysical flows are a subject of on-going investigation.

In this chapter, we have provided some examples of hybridization procedures for compatible finite element discretizations of geophysical flows. These approaches avoid the difficulty in constructing sparse approximations of dense elliptic operators. Static condensation arising from hybridizable formulations can best be interpreted as producing an *exact* Schur-complement factorization on the global hybridizable system. This eliminates the need for outer iterations from a suitable Krylov method to solve the full mixed system, and replaces the original global mixed equations with a condensed elliptic system. More extensive performance benchmarks, which requires detailed analysis of the resulting operator systems arising from hybridization, is a necessary next-step to determine whether hybridization provides a scalable solution strategy for compatible finite elements in operational settings.

5 A hybridizable method for the Euler equations

In this chapter, we derive and discuss a new hybridization method for a compatible finite element discretization of the compressible Euler equations. We start by following Staniforth and Wood (2008) and focus our attention on the compressible equations for a dry atmosphere in a rotating frame of reference, as presented in Section 2.1.1. A three-dimensional compatible finite element formulation for this system is discussed by Natale, Shipton, and Cotter (2016). The most substantial contribution, however, is the analysis of approximation properties of some classes of tensor product spaces that are compatible with the finite element exterior calculus (FEEC) framework. As a result, these spaces have been proposed for the development of global atmospheric dynamical cores (Melvin et al., 2019; Adams et al., 2019). Key examples include tensor product spaces derived from classical mixed methods such as the Raviart-Thomas (RT) and Brezzi-Douglas-Marini (BDM) discretizations (Raviart and Thomas, 1977; Brezzi, Douglas, and Marini, 1985).

As we have seen in previous chapters, a Newton-type semi-implicit discretization leads to a saddle-point system that forms the computational bottleneck of atmospheric codes. While the compatible finite element discretizations have several useful properties for geophysical flow applications, the efficient implementation of solver algorithms for the linearized equations poses a significant challenge. For reasons which will become clear, the resulting finite element system presented by Natale, Shipton, and Cotter (2016) is *not* hybridizable as formulated. To address this, we develop a new hybridizable discretization of the linear perturbation equations. This new approach permits the local elimination of prognostic variables to produce a reduced system for the pressure. In many ways, this new hybridizable approach mirrors the finite difference techniques used in the UK Met Office's dynamical core (Wood et al., 2014; Thuburn, 2016).

5.1 A compatible finite element method for the Euler equations

For context, it will be useful to first summarize the compatible finite element method presented by Natale, Shipton, and Cotter (2016). In this section, we shall assume our domain Ω is a spherical annulus, i.e., $\Omega = \mathcal{S}(R) \times [0, H]$, where $\mathcal{S}(R)$ is the surface of a sphere with radius R and H is the height of the domain. Domains of this type for geophysical flows are also characterized by their small aspect ratios, with $R \gg H$. Our starting point is the full three-dimensional system of equations describing a

compressible dry-atmosphere in a rotating domain:

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\,\boldsymbol{u} + 2\boldsymbol{\Omega} \times \boldsymbol{u} = -c_p \theta \nabla \Pi - g\hat{\boldsymbol{k}}, \tag{5.1}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0, \tag{5.2}$$

$$\frac{\partial \theta}{\partial t} + \boldsymbol{u} \cdot \nabla \theta = 0, \tag{5.3}$$

where $\boldsymbol{u} = (u, v, w)$ is the fluid velocity, ρ the density, c_p is the specific heat capacity at constant pressure, θ the potential temperature, $\boldsymbol{\Omega}$ is the Coriolis vector, g is the acceleration due to gravity, $\hat{\boldsymbol{k}}$ is the vertical normal vector, and Π is the Exner pressure defined by:

$$\Pi = \left(\frac{R_d \rho \theta}{p_{\text{ref}}}\right)^{\frac{\kappa}{1-\kappa}}. \tag{5.4}$$

Here, R_d is the gas constant of dry air, p_{ref} is a reference value for the pressure (typically taken to be 100,000 Pa), and $\kappa = R_d/c_p$. Equations (5.1)–(5.3) are also subject to a rigid-lid condition on the upper and lower boundaries of the domain, $\partial\Omega = \partial\Omega_{\text{t}} \cup \partial\Omega_{\text{b}}$:

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega. \tag{5.5}$$

The potential temperature is defined via the expression:

$$\theta = T\left(\frac{p_{\text{ref}}}{p}\right)^{\kappa}, \tag{5.6}$$

where T is the temperature. This should be interpreted as the temperature a parcel of air would obtain if moved adiabatically to a pressure of p_{ref}, with initial conditions for pressure and temperature at p and T. Note that in this formulation, the pressure can be diagnosed from the equation of state $p = \rho R_d T$ together with (5.4) and (5.6). A more complete atmospheric model will include source/sink terms in the momentum equation (5.1) and thermodynamic equation (5.3), and include the effects of moisture. The latter, in addition to necessitating a different continuity equation, would require additional modifications to (5.3). For more details on the incorporation of moisture and other physical parametrizations, see Bendall et al. (2019).

5.1.1 Tensor product finite element complex

Building a compatible finite element discretization of (5.1)–(5.3) requires first constructing a suitable finite element complex of spaces defined on Ω. As we have previously shown in Section 3.4, such meshes are generally unstructured in the horizontal, but vertically structured to facilitate vertical staggering of the potential temperature. The finite element spaces will have a tensor product structure as a result. For context and to keep the material in this chapter relatively self-contained, we will briefly summarize Section 2.4.4 here.

In order to develop a finite element complex on Ω, we require a one-dimensional complex on $[0, H]$:

$$\begin{array}{ccc} H^1([0,H]) & \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}z}} & L^2([0,H]) \\ \downarrow \pi_0^{(1)} & & \downarrow \pi_1^{(1)} \\ V_h^0 & \xrightarrow{\frac{\mathrm{d}}{\mathrm{d}z}} & V_h^1 \end{array} \tag{5.7}$$

where we use the notation for the one-dimensional differential operator $\frac{\mathrm{d}}{\mathrm{d}z}$ to emphasize that (5.7) is associated with the vertical discretization. We also need the two-dimensional complex:

$$\begin{array}{ccccc} H^1(S(R)) & \xrightarrow{\nabla^\perp} & H(\mathrm{div}; S(R)) & \xrightarrow{\nabla\cdot} & L^2(S(R)) \\ \downarrow \pi_0^{(2)} & & \downarrow \pi_1^{(2)} & & \downarrow \pi_2^{(2)} \\ U_h^0 & \xrightarrow{\nabla^\perp} & U_h^1 & \xrightarrow{\nabla\cdot} & U_h^2 \end{array} \tag{5.8}$$

where $\nabla^\perp = (-\partial_y, \partial_x)$ denotes the skew-gradient. A three-dimensional tensor product complex on Ω follows:

$$\begin{array}{ccccccc} H^1(\Omega) & \xrightarrow{\nabla} & H(\mathrm{curl}; \Omega) & \xrightarrow{\nabla\times} & H(\mathrm{div}; \Omega) & \xrightarrow{\nabla\times} & L^2(\Omega) \\ \downarrow \pi_0^{(3)} & & \downarrow \pi_1^{(3)} & & \downarrow \pi_2^{(3)} & & \downarrow \pi_3^{(3)} \\ W_h^0 & \xrightarrow{\nabla} & W_h^1 & \xrightarrow{\nabla\times} & W_h^2 & \xrightarrow{\nabla\cdot} & W_h^3 \end{array} \tag{5.9}$$

where W_h^i, $i = 0, \cdots, 3$, are tensor product finite element spaces defined as:

$$W_h^0 = U_h^0 \otimes V_h^0, \tag{5.10}$$

$$W_h^1 = \mathtt{HCurl}(U_h^0 \otimes V_h^1) \oplus \mathtt{HCurl}(U_h^1 \otimes V_h^0), \tag{5.11}$$

$$W_h^2 = \mathtt{HDiv}(U_h^1 \otimes V_h^1) \oplus \mathtt{HDiv}(U_h^2 \otimes V_h^0), \tag{5.12}$$

$$W_h^3 = U_h^2 \otimes V_h^1. \tag{5.13}$$

Detailed constructions of these spaces are provided in Section 2.4.4, which includes the implementation details necessary to construct each W_h^i. Further information can be found in McRae et al. (2016). The existence of bounded projections $\pi_i^{(d)}$, $d = 1, 2, 3$, such that the diagrams (5.7)–(5.9) commute is discussed in detail by Arnold, Falk, and Winther (2010), Arnold and Awanou (2014), and Arnold, Boffi, and Bonizzoni (2015). This includes both affine and non-affine constructions. Holst and Stern (2012) extended the diagrams for discretizations on embedded manifolds, which is summarized in Section 2.4.6. Natale, Shipton, and Cotter (2016) extended this to the tensor product complex (5.9) and quantified the errors introduced by using polynomial piece-wise approximations of manifold surfaces. We encourage the interested reader to review these references for further details.

For three-dimension systems, the sequence of spaces W_h^i form a discrete L^2 de-Rham complex of finite element spaces. An important property of choosing such spaces is that, unlike mixed methods for Navier-Stokes based on the Taylor-Hood discretization (Taylor and Hood, 1973), discretizations based on compatible finite elements *exactly* preserve the fundamental vector-calculus identities $\nabla \times (\nabla \Psi) = 0$ and $\nabla \cdot (\nabla \times \boldsymbol{w}) = 0$ in the discretized equations.

TABLE 5.1: Finite element spaces defining the vertical and horizontal complexes outlined in this section. Separate degrees may be used in the horizontal vertical spaces. This permits the use of higher-order vertical discretizations, for example.

Family	V_h^0	V_h^1	U_h^0	U_h^1	U_h^2
RT	P_k	dP_{k-1}	P_q	RT_{q-1}	dP_{q-1}
RTCF	P_k	dP_{k-1}	Q_q	RTC^f_{q-1}	dQ_{q-1}
BDM	P_k	dP_{k-1}	P_q	BDM_{q-1}	dP_{q-2}

As we have seen previously in Chapter 3 and 4, the tensor product complex can be obtained from classical mixed finite element spaces summarized in Section 2.2.2. There is flexibility in varying the degree of each complex to facilitate high order vertical staggering, while still maintaining the structure-preserving nature of the tensor product complex (Natale, Shipton, and Cotter, 2016). For $k \geq 1$, there is only one choice for the one-dimensional complex: $(\mathrm{P}_k, \mathrm{dP}_{k-1})$, where P_k and dP_{k-1} denote the usual spaces of continuous and discontinuous Lagrange elements. For the two-dimensional horizontal complex, there are a variety of choices available. Some familiar examples include:

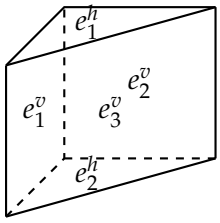
- $(\mathrm{P}_q, \mathrm{RT}_{q-1}, \mathrm{dP}_{q-1})$ for $q \geq 1$, where RT_{q-1} denotes the space of Raviart-Thomas elements of order $q-1$. This complex can be constructed on triangular horizontal mesh. In the FEEC framework, the triangular complex corresponds to the family $\mathcal{P}_q^- \Lambda^r$ (Arnold, Falk, and Winther, 2010).
- $(\mathrm{Q}_q, \mathrm{RTC}^f_{q-1}, \mathrm{dQ}_{q-1})$ is the quadrilateral complex using the Raviart-Thomas family of spaces. This complex corresponds to $\mathcal{Q}_q^- \Lambda^r$.
- $(\mathrm{P}_q, \mathrm{BDM}_{q-1}, \mathrm{dP}_{q-2})$ for triangular meshes with $q \geq 2$, where BDM_{q-1} denotes the Brezzi-Douglas-Marini elements of order $q-1$. This complex corresponds to $\mathcal{P}_q \Lambda^r$.

For simplicity, we consider the cases where the horizontal and vertical degrees are equal ($k = q$). See Table 5.1 for a summary of element choices for the one- and two-dimensional complexes summarized above. The construction of such elements, with appropriate pullbacks (including the necessary Piola transforms for $H(\mathrm{curl})$ and $H(\mathrm{div})$ elements) is discussed by Natale, Shipton, and Cotter (2016). We refer the reader to the work by Rognes, Kirby, and Logg (2009) for the implementation of $H(\mathrm{curl})$ and $H(\mathrm{div})$ elements within a code-generating framework.

The compatible finite element method for solving (5.1)–(5.3) will seek a velocity $\boldsymbol{u}_h \in \mathring{W}_h^2$, where $\mathring{W}_h^2$ is the subspace of W_h^2 such that $\boldsymbol{u}_h \cdot \boldsymbol{n}$ vanishes on the upper and lower boundaries of Ω, $\partial\Omega = \partial\Omega_\mathrm{t} \cup \partial\Omega_\mathrm{b}$. The pressure and density variables are sought in the space W_h^3. There is some flexibility in choosing an approximation space for the potential temperature. However, one in particular is preferable for atmospheric models.

5.1.2 Finite element space for the potential temperature

In Section 2.4.5, we discussed a coupled options for staggering the potential temperature. Based on that discussion, we take $\theta_h \in W_h^\theta$, where $W_h^\theta = U_h^2 \otimes V_h^0$ is the horizontally discontinuous, vertically continuous finite element space. This constructs the compatible finite element extension of the Charney-Phillips staggered



$$K = \triangle \times I$$

$$\partial K = \partial K^{\text{horiz.}} \cup \partial K^{\text{vert.}}$$

$$\partial K^{\text{horiz.}} = e_1^h \cup e_2^h$$

$$\partial K^{\text{vert.}} = e_1^v \cup e_2^v \cup e_3^v$$

FIGURE 5.1: A diagram illustrating the decomposition of the boundary of a product cell into horizontally- and vertically-aligned facets. The cell K is the result of taking the product of a triangle $\triangle$ and an interval I. An identical decomposition of facets can be made for any arbitrary product cell.

grid (Charney and Phillips, 1953), which is does not support the unphysical computational modes introduced by a Lorenz grid (Lorenz, 1960).

It was shown by Natale, Shipton, and Cotter (2016) that this choice of finite element space leads to an injective mapping between the pressure and potential temperature variables in the hydrostatic balance equation (2.232). Therefore, spurious hydrostatic pressure modes are avoided. This was further verified numerically by Melvin et al. (2018). By our choice of approximation space for θ, a transport scheme for partially continuous finite elements is necessary, which we summarize in the next section.

5.1.3 Semi-discrete formulation

In this section, we present the semi-discrete finite element formulation of the three-dimensional compressible Euler system (5.1)–(5.3). We start by reviewing some notation. With $\mathcal{T}_h$ denoting our mesh of Ω consisting of product cells K, we denote the set of facets by

$$\mathcal{E}_h = \{e \subset \partial K : \forall K \in \mathcal{T}_h\}. \tag{5.14}$$

We call $\mathcal{E}_h$ the skeleton of $\mathcal{T}_h$. We can further decompose the skeleton $\mathcal{E}_h$ into sets of horizontally- and vertically-aligned facets $\mathcal{E}_h^{\text{horiz.}}$ and $\mathcal{E}_h^{\text{vert.}}$ respectively, with

$$\mathcal{E}_h^{\text{horiz.}} = \{e \subset \partial K^{\text{horiz.}} : \forall K \in \mathcal{T}_h\}, \tag{5.15}$$

$$\mathcal{E}_h^{\text{vert.}} = \{e \subset \partial K^{\text{vert.}} : \forall K \in \mathcal{T}_h\}, \tag{5.16}$$

$$\mathcal{E}_h = \mathcal{E}_h^{\text{horiz.}} \cup \mathcal{E}_h^{\text{vert.}}. \tag{5.17}$$

In this way, we can distinguish between integrals on particular types of facets of the mesh. An illustration for this decomposition is shown in Figure 5.1.

We now introduce notation for three different types of quantities we will encounter. First, we have the usual *jump* operator defined for any double-valued vector field $\boldsymbol{w}$ on a facet $e \in \mathcal{E}_h$:

$$[\![\boldsymbol{w}]\!]_e = \begin{cases} \boldsymbol{w}|_{e^+} \cdot \boldsymbol{n}|_{e^+} + \boldsymbol{w}|_{e^-} \cdot \boldsymbol{n}|_{e^-}, & e \in \mathcal{E}_h \setminus \partial\Omega, \\ \boldsymbol{w}|_e \cdot \boldsymbol{n}|_e, & e \in \partial\Omega, \end{cases} \tag{5.18}$$

where $+$ and $-$ denotes the positive and negative sides of the facet respectively. Here, $\boldsymbol{n}|_{e^+}$ and $\boldsymbol{n}|_{e^-}$ are the unit normal vectors corresponding to the two sides of an interior facets $e = \partial K^+ \cap \partial K^-$.

Similarly, we can define a jump-like quantity for the tangential-components of a double-valued vector field $\boldsymbol{w}$:

$$\{\boldsymbol{w}\}_e = \begin{cases} \boldsymbol{n}|_{e^+} \times \boldsymbol{w}|_{e^+} + \boldsymbol{n}|_{e^-} \times \boldsymbol{w}|_{e^-} & e \in \mathcal{E}_h \setminus \partial\Omega, \\ \boldsymbol{n}|_e \times \boldsymbol{w}|_e & e \in \partial\Omega. \end{cases} \tag{5.19}$$

Finally, we define the *average* of a scalar field Ψ as:

$$\{\!\{\Psi\}\!\}_e = \begin{cases} \frac{1}{2}\left(\Psi^+ + \Psi^-\right), & e \in \mathcal{E}_h \setminus \partial\Omega \\ \Psi, & e \in \partial\Omega \end{cases} \tag{5.20}$$

where $\Psi^\pm = \Psi|_{e^\pm}$. Whenever the domain is clear by the context, we shall simply write $[\![\cdot]\!]$, $\{\cdot\}$, and $\{\!\{\cdot\}\!\}$.

To develop a finite element formulation of (5.1), we start rewriting the advection term using the vector-invariant form:

$$(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} = (\nabla\times\boldsymbol{u})\times\boldsymbol{u} + \nabla\frac{1}{2}|\boldsymbol{u}|^2, \tag{5.21}$$

and substituting this relation in the momentum equation (5.1). Then we discretize in space by taking our solution $\boldsymbol{u}_h \in \mathring{W}_h^2$, multiplying by a test function $\boldsymbol{w} \in \mathring{W}_2$, and integrating by parts. Then we obtain our first equation:

$$\begin{aligned} &\int_{\mathcal{T}_h} \boldsymbol{w}\cdot\frac{\partial \boldsymbol{u}_h}{\partial t} + \nabla_h\times(\boldsymbol{u}_h\times\boldsymbol{w})\cdot\boldsymbol{u}_h\,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{E}_h}\{\boldsymbol{u}_h\times\boldsymbol{w}\}\cdot\widetilde{\boldsymbol{u}}_h\,\mathrm{d}S \\ -\int_{\mathcal{T}_h}\nabla_h\cdot\boldsymbol{w}\frac{1}{2}|\boldsymbol{u}_h|^2\,\mathrm{d}\boldsymbol{x} &+ \int_{\mathcal{T}_h}\boldsymbol{w}\cdot(2\boldsymbol{\Omega}\times\boldsymbol{u}_h)\,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} c_p\nabla_h\cdot(\boldsymbol{w}\theta_h)\,\Pi_h\,\mathrm{d}\boldsymbol{x} \\ &+ \int_{\mathcal{E}_h^{\text{vert.}}} c_p[\![\boldsymbol{w}\theta_h]\!]\{\!\{\Pi_h\}\!\}\,\mathrm{d}S + \int_{\mathcal{T}_h} g\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\,\mathrm{d}\boldsymbol{x} = 0, \end{aligned} \tag{5.22}$$

for all $\boldsymbol{w} \in \mathring{W}_2$, where ∇_h should be interpreted as the "broken" gradient obtained by evaluating ∇ in each cell:

$$\nabla_h|_K = \nabla|_K, \tag{5.23}$$

and $\widetilde{\boldsymbol{u}}_h$ is the velocity taken from the upwind side of each facet.[1] Note that upon integrating the pressure-gradient term, only vertically-aligned surface integrals appear since $[\![\boldsymbol{w}\theta_h]\!] = 0$ on $\mathcal{E}_h^{\text{horiz.}}$; θ_h is continuous on the horizontally-aligned facets by construction of W_h^θ. Equation (5.22) summarizes our momentum transport equation.

Next, we require suitable transport scheme for ρ_h. Since $\rho_h \in W_h^3$ is fully discontinuous, a standard upwind discontinuous Galerkin method can be used:

$$\int_{\mathcal{T}_h}\phi\frac{\partial\rho_h}{\partial t} - \nabla_h\phi\cdot\boldsymbol{u}_h\rho_h\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h}[\![\boldsymbol{u}_h\phi]\!]\widetilde{\rho}_h\,\mathrm{d}S = 0, \quad \forall\phi\in W_h^3, \tag{5.24}$$

where $\widetilde{\rho}_h$ denotes ρ_h taken in the upwind direction. Both transport equations (5.22) and (5.24) are similar in construction to the nonlinear discretization of the shallow water equations in Section 3.3.

Now we turn our attention to the temperature equation (5.3). Constructing a stable transport scheme for θ_h requires a bit more care. Since θ_h is vertically continuous, there does not exist an upwind stabilization for vertical transport. Following Natale,

[1] That is, the side where $\boldsymbol{u}_h^\pm\cdot\boldsymbol{n}^\pm$ is negative

Shipton, and Cotter (2016), we can instead use a streamline-upwind Petro-Galerkin (SUPG) stabilization scheme (Brooks and Hughes, 1982). First, we multiply (5.3) by a test function $\xi \in W_h^\theta$ and integrate by parts twice to obtain the ultra-weak formulation:

$$\int_{\mathcal{T}_h} \xi \frac{\partial \theta_h}{\partial t} + \xi \boldsymbol{u}_h \cdot \nabla_h \theta_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h^{\text{vert.}}} [\![\boldsymbol{u}_h \xi]\!] \widetilde{\theta}_h - [\![\boldsymbol{u}_h \xi \theta_h]\!] \,\mathrm{d}S = 0, \quad \forall \xi \in W_\theta, \tag{5.25}$$

where $\widetilde{\theta}_h$ denotes θ_h taken from the upwind direction. Now that there are no derivative applied to the test function ξ, we can obtain the full SUPG formulation of (5.25) by substituting in the upwind-stabilized test function $\xi^* := \xi + \eta \boldsymbol{u}_h \cdot \hat{\boldsymbol{k}} \Delta t \hat{\boldsymbol{k}} \cdot \nabla_h \xi$, where η is a positive stabilization constant and Δt is a time-stepping parameter to be determined by the choice of time-integrator.

Now we have a complete semi-discretization as presented by Natale, Shipton, and Cotter (2016). The full nonlinear finite element spatial discretization of (5.1)–(5.3) reads as follows. Find $\boldsymbol{u}_h \in \mathring{W}_h^2$, $\rho \in W_h^3$, and $\theta_h \in W_h^\theta$ such that

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \frac{\partial \boldsymbol{u}_h}{\partial t} + \nabla_h \times (\boldsymbol{u}_h \times \boldsymbol{w}) \cdot \boldsymbol{u}_h \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{E}_h} \{\boldsymbol{u}_h \times \boldsymbol{w}\} \cdot \widetilde{\boldsymbol{u}}_h \,\mathrm{d}S \\
&- \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w} \frac{1}{2} |\boldsymbol{u}_h|^2 \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h} \boldsymbol{w} \cdot (2\boldsymbol{\Omega} \times \boldsymbol{u}_h) \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} c_p \nabla_h \cdot (\boldsymbol{w}\theta_h) \,\Pi_h \,\mathrm{d}\boldsymbol{x} \\
&\qquad + \int_{\mathcal{E}_h^{\text{vert.}}} c_p [\![\boldsymbol{w}\theta_h]\!] \{\!\{ \Pi_h \}\!\} \,\mathrm{d}S + \int_{\mathcal{T}_h} g \boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x} = 0, \quad (5.26) \\
&\int_{\mathcal{T}_h} \phi \frac{\partial \rho_h}{\partial t} - \nabla_h \phi \cdot \boldsymbol{u}_h \rho_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} [\![\boldsymbol{u}_h \phi]\!] \widetilde{\rho}_h \,\mathrm{d}S = 0, \quad (5.27) \\
&\int_{\mathcal{T}_h} \xi^* \frac{\partial \theta_h}{\partial t} + \xi^* \boldsymbol{u}_h \cdot \nabla_h \theta_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h^{\text{vert.}}} [\![\boldsymbol{u}_h \xi^*]\!] \widetilde{\theta}_h - [\![\boldsymbol{u}_h \xi^* \theta_h]\!] \,\mathrm{d}S = 0, \quad (5.28)
\end{aligned}$$

for all $(\boldsymbol{w}, \phi, \xi) \in \mathring{W}_h^2 \times W_h^3 \times W_h^\theta$. It is important to note here that Π_h is a function ρ_h and θ_h via (5.4), which can be diagnosed during the nonlinear solver. This defines our starting point for developing a fully-discrete nonlinear method. Equations (5.26)–(5.28) still requires further investigation. However, it is known that this approach conserves mass, but *not* energy, potential vorticity, or temperature. Initial experiments using this spatial discretization are presented in Natale, Shipton, and Cotter (2016, §7.2), where a nonlinear solver via a Picard linearization is employed. We shall summarize this approach in the next section.

5.2 Fully-discrete nonlinear method

The Picard method for solving (5.26)–(5.28) follows from Melvin et al. (2019) and Adams et al. (2019); it is based on the semi-implicit discretization used by the UK Met Office's atmospheric dynamical core (ENDGame), detailed by Wood et al. (2014) and Thuburn (2016). It is also functionally similar to the nonlinear method for the shallow water equations we described in Section 3.3. We begin by defining intermediate quantities for the time-stepping algorithm.

For given discrete fields which are known (either from initial conditions or a previous time-step) $\boldsymbol{u}_h^{n-1}$, ρ_h^{n-1}, and θ_h^{n-1}, we seek $\Delta \boldsymbol{u}_h^n$, $\Delta \rho_h^n$, and $\Delta \theta_h^n$, defined as the

nonlinear updates evolving the fields forward in time:

$$\boldsymbol{u}_h^n = \boldsymbol{u}_h^{n-1} + \Delta\boldsymbol{u}_h^n, \quad \rho_h^n = \rho_h^{n-1} + \Delta\rho_h^n, \quad \theta_h^n = \theta_h^{n-1} + \Delta\theta_h^n. \tag{5.29}$$

We determine the updates in (5.29) iteratively using a Picard method. This involves successive solutions to a linearized problem.

Let $\Delta\boldsymbol{u}_h^k$, $\Delta\rho_h^k$, and $\Delta\theta_h^k$ denote approximations to the quantities (5.29) at the k-th iteration of the Picard cycle. Before the first Picard iteration, these quantities are initialized to zero: $\Delta\boldsymbol{u}_h^0 = \boldsymbol{0}$, $\Delta\rho_h^0 = 0$, $\Delta\theta_h^0 = 0$. Now we define the residual functions at the k-th Picard iteration ($k \geq 1$) as

$$r_{\boldsymbol{u}}^k = \boldsymbol{u}_h^{n-1} + \Delta\boldsymbol{u}_h^{k-1} - \boldsymbol{u}_h^*, \quad r_\rho^k = \rho_h^{n-1} + \Delta\rho_h^{k-1} - \rho_h^*, \quad r_\theta^k = \theta_h^{n-1} + \Delta\theta_h^{k-1} - \theta_h^*, \tag{5.30}$$

where $\boldsymbol{u}_h^*$, ρ_h^*, and θ_h^* are candidate solutions for the next time-step. These are computed during each Picard iteration by solving transport equations using previously computed fields (or initial conditions if at the first time-step).

To generate the updates $\Delta\boldsymbol{u}_h^k$, $\Delta\rho_h^k$, and $\Delta\theta_h^k$, we solve a coupled linear system of equations of the form:

$$\mathcal{J}\begin{Bmatrix}\delta\boldsymbol{u}_h^k \\ \delta\rho_h^k \\ \delta\theta_h^k\end{Bmatrix} = -\begin{Bmatrix}R_{\boldsymbol{u}}^k[\boldsymbol{w}] \\ R_\rho^k[\phi] \\ R_\theta^k[\xi]\end{Bmatrix}, \tag{5.31}$$

where the δ-quantities are linear functions that update the nonlinear corrections:

$$\Delta\boldsymbol{u}_h^k = \Delta\boldsymbol{u}_h^{k-1} + \delta\boldsymbol{u}_h^k, \quad \Delta\rho_h^k = \Delta\rho_h^{k-1} + \delta\rho_h^k, \quad \Delta\theta_h^k = \Delta\theta_h^{k-1} + \delta\theta_h^k, \tag{5.32}$$

and the right-hand side co-vectors are

$$R_{\boldsymbol{u}}^k[\boldsymbol{w}] = \int_{\mathcal{T}_h} \boldsymbol{w}\cdot r_{\boldsymbol{u}}^k \,\mathrm{d}\boldsymbol{x}, \quad R_\rho^k[\phi] = \int_{\mathcal{T}_h} \phi r_\rho^k \,\mathrm{d}\boldsymbol{x}, \quad R_\theta^k[\xi] = \int_{\mathcal{T}_h} \xi r_\theta^k \,\mathrm{d}\boldsymbol{x}, \tag{5.33}$$

for all $(\boldsymbol{w},\phi,\xi) \in \mathring{W}_h^2 \times W_h^3 \times W_h^\theta$. Here, $\mathcal{J}$ is the Jacobian obtained from the linearization of (5.26)–(5.28). This procedure uses the solutions of (5.31) to determine the next iteration of Δ-quantities. The result is then used in the linear systems for $\boldsymbol{u}_h^*$, ρ_h^*, and θ_h^* and in (5.30) for the next Picard iteration. In practice, between 2-4 Picard iterations is typically good enough to generate sufficiently accurate forecasts (Melvin et al., 2019; Adams et al., 2019), which is experimentally determined. A more rigorous analysis of the Picard method has not yet been performed, but is a topic for future investigations.

Having abstractly defined the procedure, we now detail precisely how the candidates $\boldsymbol{u}_h^*$, ρ_h^*, and θ_h^* are computed. A complete description of the linear system (5.31) will follow in Section 5.2.3.

5.2.1 Obtaining a predictive velocity

To obtain $\boldsymbol{u}_h^*$, we solve the transport equation (5.26) using known field values, which reduces to a linear problem for $\boldsymbol{u}_h^*$. We use a Θ-stepping method for momentum transport, with $\Theta = \frac{1}{2}$. First, we define the midpoint quantities at the k-th Picard

iteration:

$$\boldsymbol{u}_h^{n-\frac{1}{2}} = \boldsymbol{u}_h^{n-1} + \frac{1}{2}\Delta\boldsymbol{u}_h^{k-1}, \quad \rho_h^{n-\frac{1}{2}} = \rho_h^{n-1} + \frac{1}{2}\Delta\rho_h^{k-1}, \quad \theta_h^{n-\frac{1}{2}} = \theta_h^{n-1} + \frac{1}{2}\Delta\theta_h^{k-1}, \tag{5.34}$$

where $\Delta\boldsymbol{u}_h^{k-1}$, $\Delta\rho_h^{k-1}$, and $\Delta\theta_h^{k-1}$ are determined from the previous Picard iteration (or their zero-initialization if $k = 1$). We similarly define a diagnosed midpoint pressure as function of the state variables, using the definition in (5.4):

$$\Pi_h^{n-\frac{1}{2}} = \Pi\left(\rho_h^{n-\frac{1}{2}}, \theta_h^{n-\frac{1}{2}}\right). \tag{5.35}$$

Lastly, we define an implicit midpoint velocity: $\overline{\boldsymbol{u}}_h = \frac{1}{2}\left(\boldsymbol{u}_h^* + \boldsymbol{u}_h^{n-1}\right)$.

Then the transport equation for $\boldsymbol{u}_h^*$ is constructed by discretizing (5.26) in time, using $\boldsymbol{u}_h^{n-\frac{1}{2}}$ as the advecting velocity. This produces:

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \frac{\boldsymbol{u}_h^* - \boldsymbol{u}_h^{n-1}}{\Delta t} + \nabla_h \times \left(\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right) \cdot \overline{\boldsymbol{u}}_h \,\mathrm{d}\boldsymbol{x} \\
&\quad - \int_{\mathcal{E}_h} \left\{\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right\} \cdot \widetilde{\overline{\boldsymbol{u}}}_h \,\mathrm{d}S - \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w} \frac{1}{2}\left|\boldsymbol{u}_h^{n-\frac{1}{2}}\right|^2 \mathrm{d}\boldsymbol{x} \\
&+ \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \left(2\boldsymbol{\Omega} \times \boldsymbol{u}_h^{n-\frac{1}{2}}\right) \mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} c_p \nabla_h \cdot \left(\boldsymbol{w}\theta_h^{n-\frac{1}{2}}\right) \Pi_h^{n-\frac{1}{2}} \,\mathrm{d}\boldsymbol{x} \\
&\quad + \int_{\mathcal{E}_h^{\text{vert.}}} c_p \left[\!\left[\boldsymbol{w}\theta_h^{n-\frac{1}{2}}\right]\!\right] \left\{\!\left\{\Pi_h^{n-\frac{1}{2}}\right\}\!\right\} \mathrm{d}S + \int_{\mathcal{T}_h} g\boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x} = 0,
\end{aligned} \tag{5.36}$$

for all $\boldsymbol{w} \in \mathring{W}_h^2$. Expanding out $\overline{\boldsymbol{u}}_h$ in (5.36), we obtain a linear variational problem for $\boldsymbol{u}_h^*$: find $\boldsymbol{u}_h^* \in \mathring{W}_2$ satisfying for all $\boldsymbol{w} \in \mathring{W}_h^2$:

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{u}_h^* \,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2} \int_{\mathcal{T}_h} \nabla_h \times \left(\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right) \cdot \boldsymbol{u}_h^* \,\mathrm{d}\boldsymbol{x} \\
&\qquad - \frac{\Delta t}{2} \int_{\mathcal{E}_h} \left\{\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right\} \cdot \widetilde{\boldsymbol{u}}_h^* \,\mathrm{d}S = \boldsymbol{R}^{n-\frac{1}{2}}[\boldsymbol{u}_h^{n-1}],
\end{aligned} \tag{5.37}$$

where

$$\begin{aligned}
\boldsymbol{R}^{n-\frac{1}{2}}[\boldsymbol{u}_h^{n-1}] = &\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{u}_h^{n-1} \,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2} \int_{\mathcal{T}_h} \nabla_h \times \left(\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right) \cdot \boldsymbol{u}_h^{n-1} \,\mathrm{d}\boldsymbol{x} \\
&+ \frac{\Delta t}{2} \int_{\mathcal{E}_h} \left\{\boldsymbol{u}_h^{n-\frac{1}{2}} \times \boldsymbol{w}\right\} \cdot \widetilde{\boldsymbol{u}}_h^{n-1} \,\mathrm{d}S \\
&+ \Delta t \int_{\mathcal{T}_h} \nabla_h \cdot \boldsymbol{w} \frac{1}{2}\left|\boldsymbol{u}_h^{n-\frac{1}{2}}\right|^2 \mathrm{d}\boldsymbol{x} + \Delta t c_p \int_{\mathcal{T}_h} \nabla_h \cdot \left(\boldsymbol{w}\theta_h^{n-\frac{1}{2}}\right) \Pi_h^{n-\frac{1}{2}} \,\mathrm{d}S \\
&- \Delta t c_p \int_{\mathcal{E}_h^{\text{vert.}}} \left[\!\left[\boldsymbol{w}\theta_h^{n-\frac{1}{2}}\right]\!\right] \left\{\!\left\{\Pi_h^{n-\frac{1}{2}}\right\}\!\right\} \mathrm{d}S \\
&- \Delta t \int_{\mathcal{T}_h} \boldsymbol{w} \cdot \left(2\boldsymbol{\Omega} \times \boldsymbol{u}_h^{n-\frac{1}{2}}\right) \mathrm{d}\boldsymbol{x} - \Delta t \int_{\mathcal{T}_h} g\boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x}.
\end{aligned} \tag{5.38}$$

5.2.2 Obtaining the predictive density and temperature fields

The density equation (5.27) is simply a continuity equation. Similarly for θ_h, (5.28) is just an advection equation, albeit a more complicated discretization than that of ρ_h. We can use a stabilized time-integrator to obtain the predictive quantities ρ_h^* and θ_h^*. For the time discretization of both fields, we use the stabilized three-stage Runge-Kutta scheme (SSPRK3) of Shu and Osher (Shu and Osher, 1988):

$$\begin{aligned}
\boldsymbol{x}^{(1)} &= \boldsymbol{x}^{n-1} + \mathcal{L}\left(\boldsymbol{x}^{n-1}; \boldsymbol{u}_h^{n-\frac{1}{2}}\right) \\
\boldsymbol{x}^{(2)} &= \frac{3}{4}\boldsymbol{x}^{n-1} + \frac{1}{4}\left(\boldsymbol{x}^{(1)} + \mathcal{L}\left(\boldsymbol{x}^{(1)}; \boldsymbol{u}_h^{n-\frac{1}{2}}\right)\right) \\
\boldsymbol{x}^{*} &= \frac{1}{3}\boldsymbol{x}^{n-1} + \frac{2}{3}\left(\boldsymbol{x}^{(2)} + \mathcal{L}\left(\boldsymbol{x}^{(2)}; \boldsymbol{u}_h^{n-\frac{1}{2}}\right)\right),
\end{aligned} \tag{5.39}$$

where $\mathcal{L}$ denotes the time-evolving advection operator with advecting velocity $\boldsymbol{u}_h^{n-\frac{1}{2}}$, and $\boldsymbol{x}$ denotes the field to be advected.

5.2.3 Picard method for the corrective updates

Once $\boldsymbol{u}_h^*$, ρ_h^*, and θ_h^* are constructed, we again solve a coupled linear system (5.31). We construct $\mathcal{J}$ via a partial linearization of equations (5.1)–(5.3) following Wood et al. (2014). We linearize around a background state at rest, with potential temperature, density, and pressure profiles $\bar{\theta}$, $\bar{\rho}$, and $\bar{\Pi}$ respectively.

Discretizing in time using the Crank-Nicolson method (Crank and Nicolson, 1947) then produces the approximate Jacobian (in strong form):

$$\delta\boldsymbol{u}^k + \frac{\Delta t}{2}\left(2\boldsymbol{\Omega}\times\delta\boldsymbol{u}^k\right) + \frac{\Delta t c_p}{2}\left(\delta\theta^k\nabla\bar{\Pi} + \bar{\theta}\nabla\delta\Pi\right) = -r_{\boldsymbol{u}}^k, \tag{5.40}$$

$$\delta\rho^k + \frac{\Delta t}{2}\nabla\cdot\left(\bar{\rho}\delta\boldsymbol{u}^k\right) = -r_\rho^k \tag{5.41}$$

$$\delta\theta^k + \frac{\Delta t}{2}\delta\boldsymbol{u}^k\cdot\nabla\bar{\theta} = -r_\theta^k, \tag{5.42}$$

where the pressure perturbation $\delta\Pi$ is defined via the identity:

$$\delta\Pi = \frac{\partial\bar{\Pi}}{\partial\bar{\theta}}\delta\theta^k + \frac{\partial\bar{\Pi}}{\partial\bar{\rho}}\delta\rho^k. \tag{5.43}$$

All the "bar" terms are known coefficients, determined from the linearized state. However, this is not our final Jacobian; we make one further modification. Since the effects of θ are dominate in the vertical, we discard all horizontal variations in terms containing $\delta\theta$ in the weak formulation of (5.40)–(5.42). The resulting linear finite element problem defining (5.31) reads as follows. Find $\delta\boldsymbol{u}_h^k \in \mathring{W}_h^2$, $\delta\rho_h^k \in W_h^3$,

and $\delta\theta_h^k \in W_h^\theta$ such that

$$\int_{\mathcal{T}_h} \boldsymbol{w}\cdot\delta\boldsymbol{u}_h^k + \frac{\Delta t}{2}\boldsymbol{w}\cdot\left(2\boldsymbol{\Omega}\times\delta\boldsymbol{u}_h^k\right)\,\mathrm{d}\boldsymbol{x}$$
$$-\frac{\Delta t c_p}{2}\int_{\mathcal{T}_h}\frac{\partial}{\partial z}\left(\delta\theta_h^k\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\bar{\Pi}\,\mathrm{d}\boldsymbol{x} - \frac{\Delta t c_p}{2}\int_{\mathcal{T}_h}\nabla_h\cdot\left(\bar{\theta}\boldsymbol{w}\right)\delta\Pi_h\,\mathrm{d}\boldsymbol{x}$$
$$+\frac{\Delta t c_p}{2}\int_{\mathcal{E}_h^{\text{vert.}}}[\![\bar{\theta}\boldsymbol{w}]\!]\,\{\!\{\delta\Pi_h\}\!\}\,\mathrm{d}S = -R_{\boldsymbol{u}}^k[\boldsymbol{w}], \tag{5.44}$$

$$\int_{\mathcal{T}_h}\phi\delta\rho_h^k - \frac{\Delta t}{2}\nabla_h\phi\cdot\delta\boldsymbol{u}_h^k\bar{\rho}\,\mathrm{d}\boldsymbol{x} + \frac{\Delta t}{2}\int_{\mathcal{E}_h}\left[\!\left[\phi\delta\boldsymbol{u}_h^k\right]\!\right]\{\!\{\bar{\rho}\}\!\}\,\mathrm{d}S = -R_\rho^k[\phi], \tag{5.45}$$

$$\int_{\mathcal{T}_h}\xi\delta\theta_h^k + \frac{\Delta t}{2}\xi\frac{\partial\bar{\theta}}{\partial z}\delta\boldsymbol{u}_h^k\cdot\hat{\boldsymbol{k}}\,\mathrm{d}\boldsymbol{x} = -R_\theta^k[\xi], \tag{5.46}$$

for all $(\boldsymbol{w},\phi,\xi)\in\mathring{W}_h^2\times W_h^3\times W_h^\theta$.

Equations (5.44)–(5.46) defines a tightly-coupled system of each linear perturbation and must be solved several times each time-step. There are several ways to solve this coupled problem. Our current approach is to mimic what is done in staggered finite difference codes, as presented by Wood et al. (2014). We eliminate $\delta\theta_h^k$ at the equation-level, leaving a coupled set of equations exclusively for $\delta\boldsymbol{u}_h^k$ and $\delta\rho_h^k$. However, unlike finite difference methods, the "strong" form of (5.46) no longer holds point-wise. As a result, we are introducing errors near topography.

Performing the elimination via $\delta\theta_h^k = -\frac{\Delta t}{2}\frac{\partial\bar{\theta}}{\partial z}\delta\boldsymbol{u}_h^k\cdot\hat{\boldsymbol{k}} - r_\theta^k$ and substituting in (5.44) produces the following mixed problem: find $\delta\boldsymbol{u}_h^k\in\mathring{W}_h^2$ and $\delta\rho_h^k\in W_h^3$ such that

$$A(\boldsymbol{w};\delta\boldsymbol{u}_h^k) + \frac{\Delta t c_p}{2}G(\boldsymbol{w};\delta\rho_h^k) = -R_{\boldsymbol{u},\theta}^k[\boldsymbol{w}], \tag{5.47}$$

$$\frac{\Delta t}{2}D(\phi;\delta\boldsymbol{u}_h^k) + M_3(\phi;\delta\rho_h^k) = -R_\rho^k[\phi],, \tag{5.48}$$

for all $(\boldsymbol{w},\phi)\in\mathring{W}_h^2\times W_h^3$, where the finite element forms are defined as the expressions:

$$A(\boldsymbol{w};\delta\boldsymbol{u}_h^k) = M_2(\boldsymbol{w};\delta\boldsymbol{u}_h^k) + \frac{\Delta t}{2}C(\boldsymbol{w};\delta\boldsymbol{u}_h^k) + \frac{\Delta t^2 c_p}{4}M_2^v(\boldsymbol{w};\delta\boldsymbol{u}_h^k), \tag{5.49}$$

$$M_2(\boldsymbol{w};\delta\boldsymbol{u}_h^k) = \int_{\mathcal{T}_h}\boldsymbol{w}\cdot\delta\boldsymbol{u}_h^k\,\mathrm{d}\boldsymbol{x}, \tag{5.50}$$

$$C(\boldsymbol{w};\delta\boldsymbol{u}^k) = \int_{\mathcal{T}_h}\boldsymbol{w}\cdot\left(2\boldsymbol{\Omega}\times\delta\boldsymbol{u}_h^k\right)\,\mathrm{d}\boldsymbol{x}, \tag{5.51}$$

$$M_2^v(\boldsymbol{w};\delta\boldsymbol{u}_h^k) = \int_{\mathcal{T}_h}\frac{\partial}{\partial z}\left(\frac{\partial\bar{\theta}}{\partial z}\delta\boldsymbol{u}_h^k\cdot\hat{\boldsymbol{k}}\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\bar{\Pi}\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{T}_h}\frac{\partial\bar{\Pi}}{\partial z}\delta\boldsymbol{u}_h^k\cdot\hat{\boldsymbol{k}}\frac{\partial}{\partial z}\left(\bar{\theta}\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\,\mathrm{d}\boldsymbol{x}$$
$$-\int_{\mathcal{E}_h^{\text{vert.}}}[\![\bar{\theta}\boldsymbol{w}]\!]\left\{\!\!\left\{\frac{\partial\bar{\Pi}}{\partial z}\delta\boldsymbol{u}_h^k\cdot\hat{\boldsymbol{k}}\right\}\!\!\right\}\,\mathrm{d}S, \tag{5.52}$$

$$G(\boldsymbol{w};\delta\rho_h^k) = -\int_{\mathcal{T}_h}\frac{\partial\bar{\Pi}}{\partial\bar{\rho}}\delta\rho_h^k\nabla_h\cdot\left(\bar{\theta}\boldsymbol{w}\right)\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h^{\text{vert.}}}[\![\bar{\theta}\boldsymbol{w}]\!]\left\{\!\!\left\{\frac{\partial\bar{\Pi}}{\partial\bar{\rho}}\delta\rho_h^k\right\}\!\!\right\}\,\mathrm{d}S, \tag{5.53}$$

$$D(\phi;\delta\boldsymbol{u}^k) = -\int_{\mathcal{T}_h}\nabla_h\phi\cdot\delta\boldsymbol{u}_h^k\bar{\rho}\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h}\left[\!\left[\phi\delta\boldsymbol{u}_h^k\right]\!\right]\{\!\{\bar{\rho}\}\!\}\,\mathrm{d}S, \tag{5.54}$$

$$M_3(\phi;\delta\rho_h^k) = \int_{\mathcal{T}_h}\phi\delta\rho_h^k\,\mathrm{d}\boldsymbol{x}, \tag{5.55}$$

and the modified residual is

$$R^k_{\boldsymbol{u},\theta}[\boldsymbol{w}] = R^k_{\boldsymbol{u}}[\boldsymbol{w}] - \frac{\Delta t c_p}{2}\int_{\mathcal{T}_h}\frac{\partial}{\partial z}\left(r^k_\theta \boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\bar{\Pi}\,\mathrm{d}\boldsymbol{x} - \frac{\Delta t}{2}\int_{\mathcal{T}_h}\frac{\partial\bar{\Pi}}{\partial z}r^k_\theta\frac{\partial}{\partial z}\left(\bar{\theta}\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\,\mathrm{d}\boldsymbol{x} + \frac{\Delta t c_p}{2}\int_{\mathcal{E}^{\text{vert.}}_h}[\![\bar{\theta}\boldsymbol{w}]\!]\left\{\!\!\left\{\frac{\partial\bar{\Pi}}{\partial\bar{\theta}}r^k_\theta\right\}\!\!\right\}\mathrm{d}S. \tag{5.56}$$

In terms of linear algebra, this leads to a mixed system for the velocity and density coefficients, $\delta\boldsymbol{U}^k$ and $\delta\boldsymbol{P}^k$, of the form:

$$\begin{bmatrix}\boldsymbol{A} & \frac{\Delta t}{2}c_p\boldsymbol{G}\\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3\end{bmatrix}\begin{Bmatrix}\delta\boldsymbol{U}^k\\ \delta\boldsymbol{P}^k\end{Bmatrix} = \begin{Bmatrix}\boldsymbol{R}_{\boldsymbol{u},\theta}\\ \boldsymbol{R}_\rho\end{Bmatrix}, \tag{5.57}$$

where $\boldsymbol{A} = \boldsymbol{M}_2 + \frac{\Delta t}{2}\boldsymbol{C} + \frac{\Delta t^2}{4}c_p\boldsymbol{M}_2^v$ is a velocity mass matrix augmented with both the Coriolis term $\boldsymbol{C}$ and vertical coupling in $\boldsymbol{M}_2^v$ as a result of eliminating $\delta\theta^k_h$, $\boldsymbol{D}$ and $\boldsymbol{G}$ are off-diagonal "div"/"grad" terms coupling velocity and density degrees of freedom, and $\boldsymbol{M}_3$ is a mass matrix in W^3_h. Once (5.57) is solved, $\delta\theta^k_h$ can be recovered by solving a mass matrix system in W^θ_h. Some initial numerical studies were conducted by Natale, Shipton, and Cotter (2016) using this approach, though the Coriolis term was neglected in $\boldsymbol{A}$. Experiments were restricted to only two vertical slice models.

The significance of these errors introduced by the approximate elimination of $\delta\theta^k_h$ are currently unknown, as initial numerical studies conducted by Natale, Shipton, and Cotter (2016) did not demonstrate any strange anomalies in the reconstructed temperature perturbation field. However, it may not be necessary to exactly treat $\delta\theta^k_h$ since this procedure is nested in a Picard method. In operational settings, the Picard system is *never* solved to convergence. This approach can simply be viewed as a *preconditioner* for the true linearization. We employ a similar elimination procedure in Section 5.3, along with more sensitive experiments which include topography in Section 5.4. These demonstrate that the errors introduced in the linear solver do not largely impact the overall evolution of the temperature field. Further analysis on the effects of treating $\delta\theta^k_h$ in this manner still requires further investigation.

As currently written, this system is *not* hybridizable; this is due to the inter-elemental coupling of *both* velocity test functions and prognostic variables appearing in the surface term:

$$\int_{\mathcal{E}^{\text{vert.}}_h}[\![\bar{\theta}\boldsymbol{w}]\!]\,\{\!\{\delta\Pi_h\}\!\}\,\mathrm{d}S. \tag{5.58}$$

This prevents direct hybridization without modifying the discretization. Instead, (5.57) must be solved iteratively using a suitable Krylov method and preconditioner.

5.2.4 An approximate Schur-complement preconditioner

As before, one could design a preconditioner starting from the inverse of the left-hand side operator in (5.57) using a Schur-complement factorization:

$$\begin{bmatrix}\boldsymbol{A} & \frac{\Delta t}{2}c_p\boldsymbol{G}\\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3\end{bmatrix}^{-1} = \begin{bmatrix}\boldsymbol{I} & -\frac{\Delta t}{2}c_p\boldsymbol{A}^{-1}\boldsymbol{G}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{A}^{-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{S}^{-1}\end{bmatrix}\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ -\frac{\Delta t}{2}\boldsymbol{D}\boldsymbol{A}^{-1} & \boldsymbol{I}\end{bmatrix}, \tag{5.59}$$

where $\boldsymbol{S} = \boldsymbol{M}_3 - \beta^2 c_p \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{G}$, $\beta = \frac{\Delta t}{2}$, is the Schur-complement operator. The operator is globally dense, since it contains the dense inverse $\boldsymbol{A}^{-1}$. A preconditioner $\boldsymbol{\mathcal{P}}$ can be constructed from (5.59) by replacing $\boldsymbol{S}$ with an approximate Schur-complement $\widehat{\boldsymbol{S}} = \boldsymbol{M}_3 - \beta^2 c_p \boldsymbol{D}\widehat{\boldsymbol{A}}^{-1}\boldsymbol{G}$, where $\widehat{\boldsymbol{A}} = \mathrm{Diag}(\boldsymbol{A})$.

One can then approximate the inverse of $\boldsymbol{S}$ by iteratively inverting $\widehat{\boldsymbol{S}}$ instead. In contrast to the original operator, $\widehat{\boldsymbol{S}}$ is sparse, but less is understood about this approach, as no formal analysis has been performed on such a system. Note that this approximation suffers from the same problem as in the gravity wave example from Section 4.5.3; it fails to take into account the Coriolis term of $2\boldsymbol{\Omega} \times \delta \boldsymbol{u}_h^k$ due to its contributions being located on off-diagonal positions of the matrix $\boldsymbol{A}$.

Moreover, simplified models such as the linear gravity wave, only capture *some* of the characteristics of (5.57). The equations here are far more difficult to analyze and precondition. Black-box preconditioners using the Schur-complement factorization above have some success in two-dimensional vertical slice models. However, for full three-dimensional experiments, little success has been achieved as iterative methods appear to struggle in reaching convergence. To circumvent this, we develop an alternative discretization which is, in fact, *hybridizable*. The resulting *sparse* hybridized equation appears to be far more numerically amenable to standard iterative techniques for nonsymmetric elliptic equations and negates the need for an outer Krylov method to invert the block system.

5.3 A hybridizable method for the compressible equations

The main difficulty with the approach detailed in Section 5.2.4 is designing a robust solver. This was discussed by Mitchell and Müller (2016) within the context of tensor product multigrid methods (Börm and Hiptmair, 2001; Müller and Scheichl, 2014) for a linear compressible atmospheric model. It is a subject of ongoing research to develop a similar scheme for the equations considered in this section. Here, we present an alternative discretization resulting in a system where the techniques of Chapters 3 and 4 can be applied effectively.

The hybridization procedure we outline in this section reformulates the perturbation system (5.40)–(5.42) in such a way that the system is rendered *block-sparse*. Therefore the prognostic variables can be eliminated via element-wise static condensation to produce a new sparse system for unknowns defined only on the mesh skeleton. The recovery of the prognostic variables can be obtained *locally* by solving a sequence of linear systems in each cell of the mesh. We begin by revisiting the linearized momentum equation.

5.3.1 Discontinuous $H(\mathrm{div})$ velocity fields

Velocity fields $\boldsymbol{w} \in W_h^2$ belong to a subspace of $H(\mathrm{div})$, characterized by the continuity $\boldsymbol{w} \cdot \boldsymbol{n}$ on $\mathcal{E}_h$. Following our discussion in Section 3.1.2, we define the "broken" velocity space as:

$$\widehat{W}_h^2 = \{\boldsymbol{w} \in [L^2(\Omega_h)]^n : \boldsymbol{w}|_K \in W_h^2(K), \forall K \in \mathcal{T}_h\}. \tag{5.60}$$

This space consists of vector fields locally constructed from basis functions in W_h^2 in each cell K, but with normal components not necessarily continuous on $\mathcal{E}_h$. In light of this definition, we note that $W_h^2 = \widehat{W}_h^2 \cap \{\boldsymbol{w} : [\![\boldsymbol{w}]\!] = 0 \text{ on } \mathcal{E}_h\}$ is an $H(\text{div})$-subspace of $\widehat{W}_h^2$.

As with the other models considered in Chapter 3, the hybridization of (5.44)–(5.46) begins with taking the linear perturbation $\delta \boldsymbol{u}_h^k$ and test functions $\boldsymbol{w}$ in $\widehat{W}_h^2$ rather than W_h^2. Following our previous convention, we denote the "broken" velocity perturbation as $\delta \widehat{\boldsymbol{u}}_h^k \in \widehat{W}_h^2$.

Multiplying (5.40) by $\boldsymbol{w} \in \widehat{W}_h^2$ and integrating by parts in a single cell K produces:

$$\begin{aligned}
&\int_K \boldsymbol{w} \cdot \delta \widehat{\boldsymbol{u}}_h^k + \frac{\Delta t}{2} \boldsymbol{w} \cdot \left(2\boldsymbol{\Omega} \times \delta \widehat{\boldsymbol{u}}_h^k\right) \mathrm{d}\boldsymbol{x} \\
-\frac{\Delta t c_p}{2} \int_K \frac{\partial}{\partial z} \left(\delta \theta_h^k \boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right) \bar{\Pi} \, \mathrm{d}\boldsymbol{x} &- \frac{\Delta t c_p}{2} \int_K \nabla \cdot \left(\bar{\theta} \boldsymbol{w}\right) \delta \Pi_h \, \mathrm{d}\boldsymbol{x} \\
&+ \frac{\Delta t c_p}{2} \int_{\partial K} \left(\delta \theta_h^k \hat{\boldsymbol{k}} \left(\boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right)\right) \cdot \boldsymbol{n} \bar{\Pi} \, \mathrm{d}S \\
&+ \frac{\Delta t c_p}{2} \int_{\partial K} \left(\bar{\theta} \boldsymbol{w}\right) \cdot \boldsymbol{n} \delta \Pi_h \, \mathrm{d}S = -\int_K \boldsymbol{w} \cdot r_{\boldsymbol{u}}^k \, \mathrm{d}\boldsymbol{x}, \quad (5.61)
\end{aligned}$$

for all $\boldsymbol{w} \in W_h^2(K)$, where we have dropped the horizontal variations in $\delta \theta_h^k$ as before. Assembling all the local contributions, we obtain the equation on $\mathcal{T}_h$:

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \delta \widehat{\boldsymbol{u}}_h^k + \frac{\Delta t}{2} \boldsymbol{w} \cdot \left(2\boldsymbol{\Omega} \times \delta \widehat{\boldsymbol{u}}_h^k\right) \mathrm{d}\boldsymbol{x} \\
-\frac{\Delta t c_p}{2} \int_{\mathcal{T}_h} \frac{\partial}{\partial z} \left(\delta \theta_h^k \boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right) \bar{\Pi} \, \mathrm{d}\boldsymbol{x} &- \frac{\Delta t c_p}{2} \int_{\mathcal{T}_h} \nabla_h \cdot \left(\bar{\theta} \boldsymbol{w}\right) \delta \Pi_h \, \mathrm{d}\boldsymbol{x} \\
&+ \frac{\Delta t c_p}{2} \int_{\mathcal{E}_h^{\text{horiz.}}} \left[\!\!\left[\delta \theta_h^k \hat{\boldsymbol{k}} \left(\boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right)\right]\!\!\right] \{\!\{\bar{\Pi}\}\!\} \, \mathrm{d}S \\
&+ \frac{\Delta t c_p}{2} \int_{\mathcal{E}_h} [\![\bar{\theta} \boldsymbol{w}]\!] \{\!\{\delta \Pi_h\}\!\} \, \mathrm{d}S = -\widehat{R}_{\boldsymbol{u}}^k[\boldsymbol{w}], \quad (5.62)
\end{aligned}$$

for all $\boldsymbol{w} \in \widehat{W}_h^2$. A couple changes in the weak momentum equation occurs as a result of testing in the space $\widehat{W}_h^2$.

- Surface integrals on $\mathcal{E}_h^{\text{horiz.}}$ reappear since we are no longer guaranteed that $[\![\hat{\boldsymbol{k}} \boldsymbol{w} \cdot \hat{\boldsymbol{k}}]\!] = 0$ on $\mathcal{E}_h^{\text{horiz.}}$. This was not the case previously, as our previous velocities were always continuous in their normal components. However, the surface terms on $\mathcal{E}_h^{\text{vert.}}$ still vanish due to the vertical continuity of $\delta \theta_h^k$.
- Similarly, a surface integral appears on the entire mesh skeleton by the discontinuity of $\boldsymbol{w} \cdot \boldsymbol{n}$ on $\mathcal{E}_h$. We use this term to introduce a new auxiliary variable on the skeleton.

5.3.2 Lagrange multipliers and the pressure trace

Since the original compatible finite element spatial discretization (5.26)–(5.28) constructs a velocity in W_h^2, we require that our new velocity perturbation $\delta \widehat{\boldsymbol{u}}_h^k \in \widehat{W}_h^2$ has continuous normal components. In other words, we want $[\![\delta \widehat{\boldsymbol{u}}_h^k]\!] = 0$ on $\mathcal{E}_h$. We accomplish this by imposing a *jump* condition on $\delta \widehat{\boldsymbol{u}}_h^k \cdot \boldsymbol{n}$. More specifically, we use

functions in an appropriately defined trace space W_h^{tr} to enforce:

$$\int_{\mathcal{E}_h} \gamma \left[\!\left[\delta\widehat{\boldsymbol{u}}_h^k \right]\!\right] \,\mathrm{d}S = \int_{\mathcal{E}_h \setminus \partial\Omega} \gamma \left[\!\left[\delta\widehat{\boldsymbol{u}}_h^k \right]\!\right] \,\mathrm{d}S + \int_{\partial\Omega} \gamma \delta\widehat{\boldsymbol{u}}_h^k \cdot \boldsymbol{n} \,\mathrm{d}S = 0, \quad \forall \gamma \in W_2^{\mathrm{tr}}, \tag{5.63}$$

where W_2^{tr} is defined as the set:

$$W_h^{\mathrm{tr}} = \{\gamma \in L^2(\mathcal{E}_h) : \gamma|_e \in \mathcal{P}_k(e), \forall e \in \mathcal{E}_h\}. \tag{5.64}$$

Here, $\mathcal{P}_k(e)$ is a polynomial space of degree $\leq k$ on $e \in \mathcal{E}_h$. The degree k is determined by the degree of the polynomial space containing $\boldsymbol{w} \cdot \boldsymbol{n}|_e$, for all $\boldsymbol{w} \in W_h^2$. Requiring the trace unknowns to belong in the same polynomial spaces as the normal components of $\boldsymbol{w} \cdot \boldsymbol{n}$ is analogous to the consistency requirement of standard hybridizable methods from Section 3.1.2.

Equation (5.63) ensures we obtain a velocity perturbation whose jump of the normal components vanish on the mesh skeleton. Moreover, this equation also enforces the rigid lid condition on the top and bottom regions of Ω. Meaning that all essential boundary conditions become natural conditions, enforced through the integral forms in the weak formulation. If we construct a velocity satisfying (5.63), then we have $\delta\widehat{\boldsymbol{u}}_h^k \in \mathring{W}_h^2$ by Lemma 2 in Section 3.1.2.

In order to close the system, we introduce an auxiliary unknown which arises from integrating the pressure gradient term. In the original perturbation system (5.44)–(5.46), the pressure gradient term appears as:

$$\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \left(\bar{\theta} \nabla \delta\Pi_h\right) \,\mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}_h} \nabla_h \cdot \left(\bar{\theta}\boldsymbol{w}\right) \delta\Pi_h \,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h} \left[\!\left[\bar{\theta}\boldsymbol{w} \right]\!\right] \left\{\!\left\{ \delta\Pi_h \right\}\!\right\} \mathrm{d}S, \tag{5.65}$$

where $\boldsymbol{w} \in W_h^2$. Because of (5.65), local reconstruction of $\delta\boldsymbol{u}^k$ is impossible due to the dependence on the trial unknowns $\delta\theta_h^k$ and $\delta\rho_h^k$ in neighboring cells (via the facet average $\{\!\{\delta\Pi_h\}\!\}$). To remedy this, we introduce a new independent unknown in W_2^{tr} approximating the average contribution of the pressure perturbation on cell facets. By construction, the trace is a *single-valued* scalar field on $\mathcal{E}_h$.

Integrating in a single cell K, we introduce $\lambda_h \in \widehat{W}_h^{\mathrm{tr}}(\partial K)$ appearing in the surface integral:

$$\int_K \boldsymbol{w} \cdot \left(\bar{\theta} \nabla \delta\Pi_h\right) \,\mathrm{d}\boldsymbol{x} = -\int_K \nabla \cdot \left(\bar{\theta}\boldsymbol{w}\right) \delta\Pi_h \,\mathrm{d}\boldsymbol{x} + \int_{\partial K} \left(\bar{\theta}\boldsymbol{w}\right) \cdot \boldsymbol{n} \lambda_h \,\mathrm{d}S, \tag{5.66}$$

for all $\boldsymbol{w} \in W_h^2(K)$. After collecting all the cell-wise contributions, we have the full momentum equation containing three independent fields:

$$\begin{aligned}
&\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \delta\widehat{\boldsymbol{u}}_h^k + \frac{\Delta t}{2} \boldsymbol{w} \cdot \left(2\boldsymbol{\Omega} \times \delta\widehat{\boldsymbol{u}}_h^k\right) \,\mathrm{d}\boldsymbol{x} \\
-\frac{\Delta t c_p}{2} \int_{\mathcal{T}_h} \frac{\partial}{\partial z} \left(\delta\theta_h^k \boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right) \bar{\Pi} \,\mathrm{d}\boldsymbol{x} &- \frac{\Delta t c_p}{2} \int_{\mathcal{T}_h} \nabla_h \cdot \left(\bar{\theta}\boldsymbol{w}\right) \delta\Pi_h \,\mathrm{d}\boldsymbol{x} \\
&+\frac{\Delta t c_p}{2} \int_{\mathcal{E}_h^{\mathrm{horiz.}}} \left[\!\left[\delta\theta_h^k \hat{\boldsymbol{k}} \left(\boldsymbol{w} \cdot \hat{\boldsymbol{k}}\right) \right]\!\right] \left\{\!\left\{ \bar{\Pi} \right\}\!\right\} \mathrm{d}S \\
&+\frac{\Delta t c_p}{2} \int_{\mathcal{E}_h} \left[\!\left[\bar{\theta}\boldsymbol{w} \right]\!\right] \lambda_h \,\mathrm{d}S = -\widehat{R}_{\boldsymbol{u}}^k[\boldsymbol{w}],
\end{aligned} \tag{5.67}$$

for all $\boldsymbol{w} \in \widehat{W}_h^2$, where $\widehat{R}_{\boldsymbol{u}}^k[\boldsymbol{w}]$ denotes the linear residual tested with discontinuous

functions. We will proceed to eliminated $\delta\theta_h^k$ as in Section 5.2.3. The new unknown λ_h is a global approximation to the average of $\delta\Pi_h$ on $\mathcal{E}_h$. An immediate difference between the hybridizable methods in Chapter 3 and here is that λ_h is approximating a different independent unknown; one may expect the hybridized variable to approximate a density-trace, but here we have reintroduced the Exner pressure into the linear equations.

After performing the elimination of $\delta\theta_h^k$, the hybridization of (5.44)–(5.46) is summarized as the following variational problem: find $\delta\widehat{\boldsymbol{u}}_h^k \in \widehat{W}_2$, $\delta\rho_h^k \in W_3$, and $\lambda_h \in W_h^{\mathrm{tr}}$ such that

$$A_{\text{hybrid.}}(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) + \frac{\Delta t c_p}{2} G_{\text{hybrid.}}(\boldsymbol{w};\delta\rho_h^k) + \frac{\Delta t c_p}{2} K(\boldsymbol{w};\lambda_h) = -\widehat{R}_{\boldsymbol{u},\theta}^k[\boldsymbol{w}], \tag{5.68}$$

$$\frac{\Delta t}{2} D_{\text{hybrid.}}(\phi;\delta\widehat{\boldsymbol{u}}_h^k) + M_3(\phi;\delta\rho_h^k) = -R_\rho^k[\phi],, \tag{5.69}$$

$$K(\delta\widehat{\boldsymbol{u}}_h^k;\gamma) = 0, \tag{5.70}$$

for all $(\boldsymbol{w},\phi,\gamma) \in \widehat{W}_h^2 \times W_h^3 \times W_h^{\mathrm{tr}}$, where the finite element forms are give by

$$A_{\text{hybrid.}}(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) = \widehat{M}_2(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) + \frac{\Delta t}{2}\widehat{C}(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) + \frac{\Delta t^2}{4}\widehat{M}_2^v(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k), \tag{5.71}$$

$$\widehat{M}_2(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) = \int_{\mathcal{T}_h} \boldsymbol{w}\cdot\delta\widehat{\boldsymbol{u}}_h^k \,\mathrm{d}\boldsymbol{x}, \tag{5.72}$$

$$\widehat{C}(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) = \int_{\mathcal{T}_h} \boldsymbol{w}\cdot\left(2\boldsymbol{\Omega}\times\delta\widehat{\boldsymbol{u}}_h^k\right)\mathrm{d}x, \tag{5.73}$$

$$\begin{aligned}\widehat{M}_2^v(\boldsymbol{w};\delta\widehat{\boldsymbol{u}}_h^k) = &\int_{\mathcal{T}_h} \frac{\partial\bar{\Pi}}{\partial z}\delta\widehat{\boldsymbol{u}}_h^k\cdot\hat{\boldsymbol{k}}\frac{\partial}{\partial z}\left(\bar{\theta}\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\mathrm{d}x \\ &- \int_{\mathcal{E}_h^{\text{horiz.}}}\left[\!\!\left[\frac{\partial\bar{\theta}}{\partial z}\delta\widehat{\boldsymbol{u}}_h^k\cdot\hat{\boldsymbol{k}}\left(\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\hat{\boldsymbol{k}}\right]\!\!\right]\{\!\{\bar{\Pi}\}\!\}\mathrm{d}s,\end{aligned} \tag{5.74}$$

$$G_{\text{hybrid.}}(\boldsymbol{w};\delta\rho_h^k) = -\int_{\mathcal{T}_h}\frac{\partial\bar{\Pi}}{\partial\bar{\rho}}\delta\rho_h^k\nabla_h\cdot\left(\bar{\theta}\boldsymbol{w}\right)\,\mathrm{d}\boldsymbol{x} \tag{5.75}$$

$$D_{\text{hybrid.}}(\phi;\delta\widehat{\boldsymbol{u}}_h^k) = -\int_{\mathcal{T}_h}\nabla_h\phi\cdot\delta\widehat{\boldsymbol{u}}_h^k\bar{\rho}\,\mathrm{d}\boldsymbol{x} + \int_{\mathcal{E}_h}\left[\!\!\left[\phi\delta\widehat{\boldsymbol{u}}_h^k\right]\!\!\right]\{\!\{\bar{\rho}\}\!\}\,\mathrm{d}S, \tag{5.76}$$

$$M_3(\phi;\delta\rho_h^k) = \int_{\mathcal{T}_h}\phi\delta\rho_h^k\,\mathrm{d}\boldsymbol{x}, \tag{5.77}$$

$$K(\boldsymbol{w};\lambda_h) = \int_{\mathcal{E}_h}\left[\!\!\left[\bar{\theta}\boldsymbol{w}\right]\!\!\right]\lambda_h\,\mathrm{d}S, \tag{5.78}$$

and the modified residual is given by:

$$\begin{aligned}\widehat{R}_{\boldsymbol{u},\theta}^k[\boldsymbol{w}] = \widehat{R}_{\boldsymbol{u}}^k[\boldsymbol{w}] &- \frac{\Delta t c_p}{2}\int_{\mathcal{T}_h}\frac{\partial}{\partial z}\left(r_\theta^k\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\bar{\Pi}\,\mathrm{d}\boldsymbol{x} \\ &- \frac{\Delta t}{2}\int_{\mathcal{T}_h}\frac{\partial\bar{\Pi}}{\partial z}r_\theta^k\frac{\partial}{\partial z}\left(\bar{\theta}\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\,\mathrm{d}\boldsymbol{x} \\ &+ \frac{\Delta t c_p}{2}\int_{\mathcal{E}_h^{\text{horiz.}}}\left[\!\!\left[r_\theta^k\left(\boldsymbol{w}\cdot\hat{\boldsymbol{k}}\right)\hat{\boldsymbol{k}}\right]\!\!\right]\{\!\{\bar{\Pi}\}\!\}\,\mathrm{d}S.\end{aligned} \tag{5.79}$$

Note that both $\{\!\{\bar{\Pi}\}\!\}$ and $\{\!\{\bar{\rho}\}\!\}$ are averages of *known* fields. This allows us to treat the quantities as functions in W_h^{tr} whose values on the facets are the averages of $\bar{\Pi}$ and $\bar{\rho}$, respectively.

Observe that, while we have dramatically increased the total number of unknowns, we have actually *simplified* the equations just by introducing λ_h. Lower-order surface terms which were originally present in both A and G in (5.52) and (5.53) no longer appear. This is due to the fact that introducing λ_h as an independent unknown decouples surface terms with $\delta\Pi_h$; they are now treated independently in the global system for λ_h.

Since all prognostic variables are only coupled within each cell, we can apply element-wise static condensation to the system (5.68)–(5.70) to assemble a reduced problem for λ_h on the mesh skeleton. Once λ_h is determined, $\delta\widehat{\boldsymbol{u}}_h^k$ and $\delta\rho_h^k$ are reconstructed via solving elemental linear systems. The jump condition:

$$K(\gamma; \delta\widehat{\boldsymbol{u}}_h^k) = 0, \quad \forall \gamma \in W_h^{\mathrm{tr}} \tag{5.80}$$

ensures that our computed velocity perturbation is both $H(\mathrm{div})$-conforming and satisfies the slip-boundary conditions. Note also that scaling (5.80) by any non-zero coefficient does not change the result. Once both fields are determined, $\delta\theta_h^k$ is recovered by solving a simple mass matrix system. We shall elaborate further in our summary of the complete solution procedure for solving (5.1)–(5.3).

Remark 10. *The original spatial discretization in* (5.26)–(5.28)*, along with the perturbation system in* (5.44)–(5.46)*, still requires further analysis. Natale, Shipton, and Cotter (2016) proposed the discretization as an extension of the ideas of Cotter and Shipton (2012) to the full compressible equations* (5.1)–(5.3)*. The scheme does conserve mass, but not energy, potential vorticity, or temperature.*

As a consequence, the hybridizable formulation (5.68)–(5.70) *inherits the same conservation properties for mass, but nothing more. Further analysis of the hybridizable method and the original discretizations are currently work in-progress.*

5.3.3 Full solution procedure with hybridization

The full solution algorithm for solving the compressible Euler equations (2.2)–(2.5) will use the methods outlined in all preceding sections. The implementation of this method is detailed in the open-source project: Gusto (firedrakeproject.org/gusto/), which is a three-dimensional finite element dynamical core toolkit built around the Firedrake Project (Rathgeber et al., 2017). Algorithmically, the procedure is similar to how we described in Section 5.2. However, the nonlinear approach is executed using slightly different mechanics to achieve the same result. The formulation of the perturbation equations and the hybridization method remains unaffected.

Let $\boldsymbol{q}^n$ denote the prognostic state variables $(\boldsymbol{u}_h^n, \rho_h^n, \theta_h^n)$ (Π_h^n is diagnosed from its definition in (5.4)). We use $\mathcal{F}(\boldsymbol{q}^n)$ to describe the forcing operation which is detailed below. The "predicting" advection step evolves the state in time using the midpoint quantities $\boldsymbol{q}^{n-\frac{1}{2}}$, with $\boldsymbol{u}_h^{n-\frac{1}{2}}$ as the advecting velocity. We denote this operation by $\mathcal{A}_{\boldsymbol{u}}(\boldsymbol{q}^{n-\frac{1}{2}}; \boldsymbol{q}^\star)$, where $\boldsymbol{q}^\star$ is an input state to be advected. The "correction" step first finds the residual $\boldsymbol{R}^k$, which are reduced iteratively in the Picard method by successively computing best approximations to the error: $\Delta\boldsymbol{q} = \boldsymbol{q}^n - \boldsymbol{q}^{n-1}$. The errors are approximated by solving the linearized system $\mathcal{J}\delta\boldsymbol{q}^k = \boldsymbol{R}^k$ in the k-th Picard iteration. The linear updates are used to update the field for the next time-step. At the end of a Picard cycle, we have the option to apply a diffusion scheme, which we denote as the operation $\mathcal{D}(\boldsymbol{q}^n)$.

The main contribution of this chapter revolves around solving the linear system, so we refrain from going in-depth into the mechanics of both $\mathcal{F}$ and $\mathcal{D}$. We simply summarize here to establish context. Note that a parameter α is a time-stepping coefficient, which can be varied. In our case, we simply take $\alpha = \frac{1}{2}$. Following Wood et al. (2014), we perform a total of 4 Picard iterations per time-step. This was experimentally determined to be an optimal balance between the total number of required linear solves and the quality of computed solutions. The results we have obtained are in agreement with similar studies performed by Melvin et al. (2019).

The entire simulation process of the Gusto dynamical core can be categorized into two main stages: (1) the initialization stage; and (2) the evolution of the nonlinear equations. We shall quickly summarize the initialization routine first.

Initialization of the dynamical core

For many test cases, the background state of the model will be in hydrostatic balance. In terms of initializing the model, we need to compute an Exner pressure field $\bar{\Pi}$ given an initial temperature field $\bar{\theta}$ and a boundary condition for the pressure (such as a prescribed surface pressure). After determining $\bar{\Pi}$, a background density $\bar{\rho}$ can be determined by solving a variational problem using (5.4). Since $\bar{\rho}$ is determined by $\bar{\Pi}$, we shall only discuss how $\bar{\Pi}$ is obtained. For further details on this initialization stage, we refer the interested reader to Bendall et al. (2019).

Since large-scale atmoshperic flows will always remain near a state of hydrostatic balance (Melvin et al., 2010; Wood et al., 2014), this balanced background state is used in the linearization for the Picard method. Therefore, it is critical that the basic state of hydrostatic balance is accurately represented in the numerical model. In Gusto, this initialization routine uses the hybridizable method for the hydrostatic equation (3.204)–(3.205), which was developed in Section 3.4.1. Since the equations are only coupled in vertical columns, the hybridizable system can be efficiently evaluated by inverted the trace system column-wise.

Semi-implicit procedure

Once the background state $(\bar{\Pi}, \bar{\theta}, \bar{\rho})$ is determined, the Jacobian is obtained via a partial linearization of (5.1)–(5.3) around this state, detailed in Section 5.2.3. We then employ the hybridizable method in Section 5.3 to solve the perturbation equations. The entire semi-implicit procedure is summarized as follows.

For each time-step n, we perform:

1. **Explicit forcing:** Compute the intermediate quantity $\boldsymbol{q}^\star$ via:

$$\boldsymbol{q}^\star \leftarrow \boldsymbol{q}^{n-1} + (1-\alpha)\Delta t \mathcal{F}(\boldsymbol{q}^{n-1}). \tag{5.81}$$

   In the case of a dry atmosphere, evaluating $\mathcal{F}(\boldsymbol{q}^{n-1})$ simply involves computing a velocity $\boldsymbol{u}_h^\star \in W_h^2$ satisfying for all $\boldsymbol{w} \in W_h^2$:

$$\begin{aligned}\int_{\mathcal{T}_h} \boldsymbol{w} \cdot \boldsymbol{u}_h^\star \,\mathrm{d}\boldsymbol{x} = &\int_{\mathcal{T}_h} c_p \nabla_h \cdot \left(\theta_h^{n-1}\boldsymbol{w}\right) - \boldsymbol{w} \cdot 2\boldsymbol{\Omega} \times \boldsymbol{u}_h^{n-1} \,\mathrm{d}\boldsymbol{x} \\ &- \int_{\mathcal{T}_h} g\boldsymbol{w} \cdot \hat{\boldsymbol{k}} \,\mathrm{d}\boldsymbol{x} - c_p \int_{\mathcal{E}_h} [\![\theta_h^{n-1}\boldsymbol{w}]\!] \{\!\{\Pi_h^{n-1}\}\!\} \,\mathrm{d}S.\end{aligned} \tag{5.82}$$

That is, $\boldsymbol{u}_h^\star$ is balanced by all non-advective terms in equation (5.1).

2. **Set**: $\boldsymbol{q}_p^n = \boldsymbol{q}^{n-1}$ and $\Delta\boldsymbol{q} = \boldsymbol{q}_p^n - \boldsymbol{q}^{n-1}$ (zero-initialization of $\Delta\boldsymbol{q}$).
3. **Picard cycle**: For $k = 1, \cdots, K$:
   (a) **Intermediate quantity**: Compute $\boldsymbol{q}^{n-\frac{1}{2}} = \boldsymbol{q}^{n-1} + \alpha\Delta\boldsymbol{q}$.
   (b) **Advect**: Advect the quantities forward in time to get the predictive fields:
   $$\boldsymbol{q}^* \leftarrow \mathcal{A}_{\boldsymbol{u}}(\boldsymbol{q}^{n-\frac{1}{2}}; \boldsymbol{q}^\star), \tag{5.83}$$
   where the application of $\mathcal{A}_{\boldsymbol{u}}(\boldsymbol{q}^{n-\frac{1}{2}}; \cdot)$ is described in Sections 5.2.1 and 5.2.2 as solving the relevant linear variational problems.
   (c) **Residuals:** Evaluate $\boldsymbol{R}^k = \boldsymbol{q}^* + \alpha\Delta t\mathcal{F}(\boldsymbol{q}_p^n) - \boldsymbol{q}_p^n$.
   (d) **Solve**: Solve for the linear perturbations:
   $$\mathcal{J}\delta\boldsymbol{q}^k = \boldsymbol{R}^k. \tag{5.84}$$
   This uses the hybridization method as formulated in (5.68)–(5.70) with reconstruction of $\delta\theta_h^k$ following afterwards.
   (e) **Update**: $\boldsymbol{q}_p^n \leftarrow \boldsymbol{q}_p^n + \delta\boldsymbol{q}^k$.
4. **Diffusion**: $\boldsymbol{q}_p^n \leftarrow \mathcal{D}(\boldsymbol{q}_p^n)$. Here, diffusive terms are treated as physical parameterizations using an interior penalty method (Arnold, 1982). Only one example uses this, which is the test case by Straka et al. (1993).
5. **Advance time-step**: $\boldsymbol{q}^{n-1} = \boldsymbol{q}_p^n$.

5.3.4 Static condensation procedure and iterative solver

With the "Gusto" algorithm summarized, we now return to our focus on the linear perturbation system. The solution method for computing $\delta\boldsymbol{q}^k$ is as follows. The system (5.68)–(5.70) has the form

$$\begin{bmatrix} \boldsymbol{A} & \frac{\Delta t c_p}{2}\boldsymbol{G} & \boldsymbol{K}^T \\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3 & \boldsymbol{0} \\ \boldsymbol{K} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \delta\widehat{\boldsymbol{U}}^k \\ \delta\boldsymbol{P}^k \\ \boldsymbol{\Lambda} \end{Bmatrix} = \begin{Bmatrix} \widehat{\boldsymbol{R}}_{\boldsymbol{u},\theta} \\ \boldsymbol{R}_\rho \\ \boldsymbol{0} \end{Bmatrix}, \tag{5.85}$$

where $\delta\widehat{\boldsymbol{U}}^k$ and $\delta\boldsymbol{P}^k$ denote the coefficient vectors for $\delta\widehat{\boldsymbol{u}}_h^k$ and $\delta\rho_h^k$, and $\boldsymbol{\Lambda}$ is the vector for the Lagrange multipliers. Due to the elimination of inter-elemental coupling of $\delta\widehat{\boldsymbol{u}}_h^k$, the 2×2 block corresponding to the coupling of velocity and density can be inverted element-wise. This allows for the simultaneous elimination of $\delta\widehat{\boldsymbol{U}}^k$ and $\delta\boldsymbol{P}^k$ in each cell, which produces the following system for $\boldsymbol{\Lambda}$:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \frac{\Delta t c_p}{2}\boldsymbol{G} \\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{K}^T \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \frac{\Delta t c_p}{2}\boldsymbol{G} \\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3 \end{bmatrix}^{-1} \begin{Bmatrix} \widehat{\boldsymbol{R}}_{\boldsymbol{u},\theta} \\ \boldsymbol{R}_\rho \end{Bmatrix}. \tag{5.86}$$

Equation (5.86) can be assembled *locally*, using Firedrake's static condensation interface described in Section 4.4.2.

Once $\boldsymbol{\Lambda}$ is computed, the velocity and density perturbations are recovered by inverting the local systems in each cell:

$$\begin{bmatrix} \boldsymbol{A} & \frac{\Delta t c_p}{2}\boldsymbol{G} \\ \frac{\Delta t}{2}\boldsymbol{D} & \boldsymbol{M}_3 \end{bmatrix} \begin{Bmatrix} \delta\widehat{\boldsymbol{U}}^k \\ \delta\boldsymbol{P}^k \end{Bmatrix} = \begin{Bmatrix} \widehat{\boldsymbol{R}}_{\boldsymbol{u},\theta} \\ \boldsymbol{R}_\rho \end{Bmatrix} - \begin{bmatrix} \boldsymbol{K}^T \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\Lambda}. \tag{5.87}$$

The resulting velocity field is in $\widehat{W}_h^2$, but we require it to be in W_h^2. Since the computed solution $\delta\widehat{\boldsymbol{u}}_h^k$ satisfies the jump condition (5.63), we can use the $H(\mathrm{div})$-averaging operator described in Section 4.4.3; we define the velocity perturbation $\delta\boldsymbol{u}_h^k \in \mathring{W}_h^2$ as the function:

$$\delta\boldsymbol{u}_h^k = \pi_{\mathrm{avg}}(\delta\widehat{\boldsymbol{u}}_h^k). \tag{5.88}$$

In our Firedrake implementation, we evaluate (5.88) by locally averaging the coefficient array: $\delta\boldsymbol{U}^k \leftarrow \boldsymbol{\Pi}_{\mathrm{avg}}\delta\widehat{\boldsymbol{U}}^k$, where $\boldsymbol{\Pi}_{\mathrm{avg}}$ is defined from (4.87).

Once $\delta\boldsymbol{u}_h^k$ is determined, we can reconstruct $\delta\theta_h^k$ by solving the linear finite element problem for $\delta\theta_h^k \in W_h^\theta$:

$$\int_{\mathcal{T}_h} \xi\delta\theta_h^k \,\mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}_h} \xi r_\theta^k \,\mathrm{d}\boldsymbol{x} - \int_{\mathcal{T}_h} \frac{\partial\bar{\theta}}{\partial z}\hat{\boldsymbol{k}}\cdot\delta\boldsymbol{u}_h^k \,\mathrm{d}\boldsymbol{x} \tag{5.89}$$

for all $\xi \in W_h^\theta$. This is just a mass-matrix system and can be efficiently inverted column-wise.

Up until this point, we made no mention on solution approaches for (5.86). At this point, the system for $\boldsymbol{\Lambda}$ requires extensive analysis to construct an effective preconditioner. While we have not yet conducted a rigorous analysis of the spectral properties for the condensed matrix in (5.86), we consider the following details to aid us in devising an effective linear solver:

- λ_h is introduced in surface terms as an approximation to the pressure correction term $\delta\Pi$ on the mesh skeleton $\mathcal{E}_h$.
- Based on the work of Melvin et al. (2010), Wood et al. (2014), and Thomas et al. (2003), we know that in staggered finite difference models, the equation for $\delta\Pi$ is a non-symmetric Helmholtz-like elliptic operator.
- Multigrid methods are increasingly attractive for the elliptic equation due to their ability to resolve multi-scale phenomena and scalable performance on modern supercomputers (Müller and Scheichl, 2014; Mitchell and Müller, 2016; Dedner, Müller, and Scheichl, 2015).
- Due to the small aspect ratio of atmospheric domains, the multigrid procedure will need to use line-relaxation methods to efficiently solve the discrete system (Börm and Hiptmair, 2001; Buckeridge and Scheichl, 2010).

With this in mind, we proceed in a similar manner to our development of a robust solver for the linearized gravity wave system in Section 4.5.3. Following Elman, Ernst, and O'Leary (2001), we employ a Krylov-enhanced multigrid preconditioner for a nonsymmetric Krylov method (flexible GMRES or GCR).

Thus far, we have achieved great success in solving (5.86) using this approach. Furthermore, we show later in a three-dimensional example that standard multigrid methods for non-symmetric elliptic equations achieves Δt-independent convergence for the $\boldsymbol{\Lambda}$ system. This strongly suggests that the operator in (5.86) at least *behaves*

like an elliptic operator. Formal analysis on the properties of the condensed trace system is a subject of on-going investigation. We therefore cannot make any conclusive statements on the spectral properties of left-hand side matrix in (5.86) at this time. To conclude this chapter, we provide results from standard test cases for atmospheric dynamical cores using our hybridization method.

5.4 Numerical results

Here, we present some standard test cases based on the test suite of Melvin et al. (2010) and Melvin et al. (2019). We start with some standard vertical slice examples, which are simulations of the full compressible equations in an (x, z)-slice geometry. We conclude with a three-dimensional example detailed in the Dynamical Core Model Intercomparison Project (DCMIP) suite (Ullrich et al., 2012). The implementation of all vertical slice examples uses a $(\mathrm{RTC}_1^f, \mathrm{dQ}_0)$ discretization and the Firedrake preconditioner "`firedrake.SCPC`," based on the Slate language detailed in Chapter 4. Due to the success of the hybridizable method in both vertical slice and three-dimensional modeling, the approach described in Section 5.3 is now a core component of the Gusto dynamical core toolkit. It has since been extended to handle discretizations of a moist atmosphere by Bendall et al. (2019).

For organization, Table 5.2 displays the model parameters used for all vertical slice models, along with relevant identifiers for context. Options configurations for the hybridization solver used in all vertical slice examples is presented in Listing 5.1. All numerical results are obtained using the new hybridization method.

TABLE 5.2: Model parameters used in Gusto for the vertical slice examples. The identifiers are listed as the following: NHGW - Non-hydrostatic Gravity Wave; NHMW - Non-hydrostatic Mountain Wave; HMW - Hydrostatic Mountain Wave; DC - Density Current. Resolution in x and z directions are provided, along with the time-step size used, computational domain, background state, surface temperature $T_{\text{surf.}}$, and background velocity U. For all test cases, the surface pressure is fixed at $p_{\text{surf.}} = 1000$ hPa.

Model parameters and test cases							
Test case	Δx (km)	Δz (m)	Δt (s)	Domain (km × km)	Background state	$T_{\text{surf.}}$ (K)	U (m/s)
NHGW	1	1000	6	300×10	N=0.01s^{-1}	300	20
NHMW	0.4	250	2.5	144×35	N=0.01s^{-1}	300	10
HMW	2	250	5	240×50	Isothermal $T = T_{\text{surf.}}$	250	20
DC	0.8	800	4	51.2×6.4	Isentropic $\theta = T_{\text{surf.}}$	300	0
	0.4	400	2				
	0.2	200	1				
	0.1	100	0.5				
	0.05	50	0.25				

LISTING 5.1: Solver options for the static condensation of the hybridizable formulation of the Euler perturbation equations. The iterative solver for the Lagrange multipliers uses a flexible Krylov method with an algebraic multigrid preconditioner using Krylov-accelerated ILU smoothers.

```
-ksp_type preonly
-pc_type python
-pc_python_type firedrake.SCPC
-pc_sc_eliminate_fields 0, 1
-condensed_field_ksp_type fgmres
-condensed_field_ksp_rtol 1.0e-8
-condensed_field_ksp_atol 1.0e-8
-condensed_field_ksp_max_it 100
-condensed_field_pc_type gamg
-condensed_field_pc_gamg_sym_graph True
-condensed_field_pc_mg_levels_ksp_type gmres
-condensed_field_pc_mg_levels_ksp_max_it 5
-condensed_field_pc_mg_levels_pc_type bjacobi
-condensed_field_pc_mg_levels_sub_pc_type ilu
```

5.4.1 Non-hydrostatic gravity waves in a periodic channel

The first example was studied by Skamarock and Klemp (1994) to investigate the accuracy of non-hydrostatic dynamics. This test examines the evolution of a potential temperature perturbation $\delta\theta$ in a constant mean flow superimposed on a background atmospheric state with constant buoyancy frequency $N = 0.01\text{s}^{-1}$. This initial potential temperature perturbation radiates symmetrically to the left and right in a 300km $\times$ 10km channel with periodic boundary conditions on the side walls.

The potential temperature perturbation is given by the expression

$$\delta\theta = \delta\theta_0 \frac{\sin\left(\frac{\pi z}{H}\right)}{1 + \left(\frac{x - \frac{L}{2}}{a}\right)^2}, \tag{5.90}$$

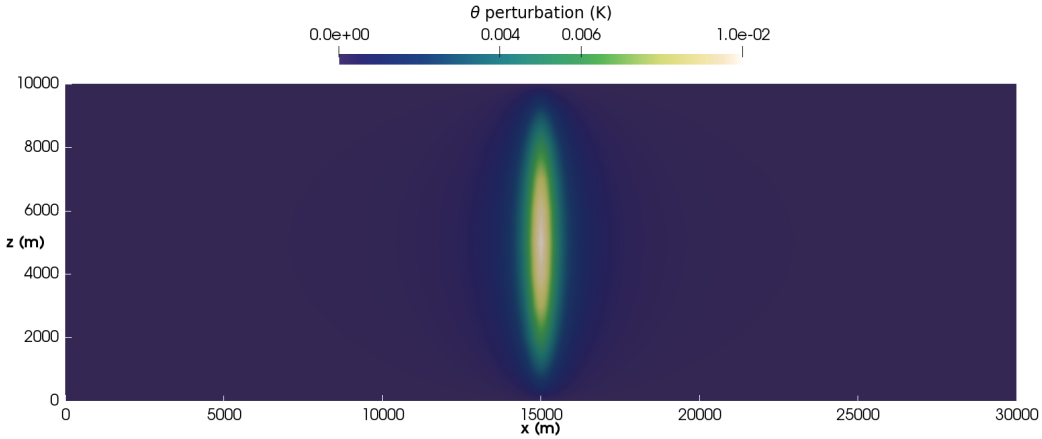
where $\delta\theta_0 = 0.01\text{K}$, $H = 10\text{km}$, $L = 300\text{km}$, and $a = 5\text{km}$. The simulation was run for a total of 3000 seconds. The evolution of potential temperature is illustrated in Figure 5.2. Our profile of $\delta\theta_h$ at $t = 3000$ seconds achieves maximal and minimal values of 2.86×10^{-3} K and -1.51×10^{-3} K respectively, which compares well with previous studies of this same test, c.f. Melvin et al. (2010), Melvin et al. (2019), Choi et al. (2014), Bao, Klöfkorn, and Nair (2015), and Li et al. (2013).

5.4.2 Hydrostatic and non-hydrostatic mountain waves

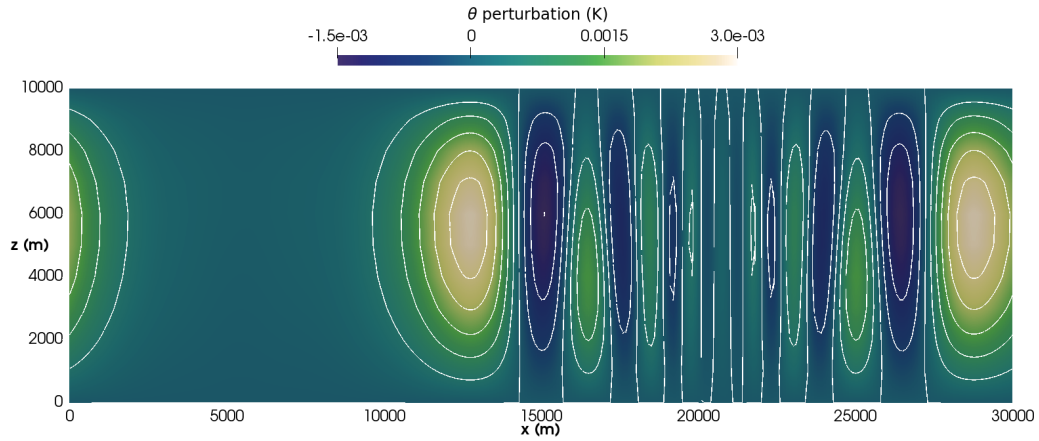
We perform a qualitative comparison of results for hydrostatic and non-hydrostatic mountain waves over a witch-of-Agnesi profile given by

$$z_S(x) = \frac{h_m a^2}{(x - x_c)^2 + a^2}, \tag{5.91}$$

and an imposed mean-flow velocity in the background U to galvanize the wave. For the non-hydrostatic flow regime, we take $a = 10\text{km}$, $x_c = L/2$, where L is the length of the domain, and $h_m = 1\text{m}$. The mean-flow is set as $U = 20\text{ms}^{-1}$.



(A) Initial θ perturbation.



(B) Evolution of $\delta\theta_h$ at $t = 3000$s.

FIGURE 5.2: Evolution of the potential temperature perturbation $\delta\theta_h$ for the non-hydrostatic gravity wave test in a periodic channel. Counters are shown in Figure 5.2B ranging from $[-1.5 \times 10^{-3}, 3.0 \times^{-3}]$ in intervals of 5×10^{-4} K.

The background state is isothermal, with $T_{\text{surf.}} = 250$ K and initialized in a state of hydrostatic balance. Following Klemp, Dudhia, and Hassiotis (2008), we impose an absorbing condition in the upper-atmosphere in the form of a sponge layer. This condition takes the form

$$\mu(z) = \begin{cases} 0 & \text{if } z < z_c, \\ \bar{\mu} \sin^2\left(\frac{\pi}{2} \frac{z - z_c}{H - z_c}\right) & \text{if } z \geq z_c, \end{cases} \tag{5.92}$$

where $z_c = 20$km and $\bar{\mu} = 0.3/\Delta t$. The sponge function $\mu(z)$ is applied implicitly by scaling test functions in the vertical component of the velocity equation during implicit forcing. This prevents spurious waves from reflecting off the boundary lid.

For the non-hydrostatic configuration, a is reduced to 1km and the mean wind speed $U = 10\text{ms}^{-1}$. The background state has constant buoyancy frequency $N = 0.01\text{s}^{-1}$, and therefore has finite depth. As a result, the atmospheric lid is lowered to 35km (compared to 50km in the hydrostatic case) and we set $z_c = 10$km. The sponge layer is adjusted by setting $\bar{\mu} = 0.15/\Delta t$.

For the hydrostatic test case, the model is run in a hydrostatic configuration and run up to $t = 15000$s. The vertical velocity perturbation at the final time is shown in

Figure 5.3A. Qualitatively, the field is comparable to previous tests, for example see Melvin et al. (2019), as well as the linear analytic solutions presented for comparison by Wood et al. (2014, Figure 4(d)). However, in some regions approximately above 4000 m in the vertical, the finite element solution exhibits peaks which are less than 0.0005 m/s in magnitude. It appears the numerical model is slightly underestimating the maximal magnitudes in the vertical velocity, a phenomena also observed by Giraldo and Restelli (2008) and Choi et al. (2014).

Similar phenomena was observed in the non-hydrostatic configuration. The model was run to a final time of 9000s. The resulting vertical velocity perturbation is shown in Figure 5.3B. Again, qualitatively the Gusto model agrees fairly well with the studies by Melvin et al. (2010) and Melvin et al. (2019). We observe that our model slightly underestimates the maximal contour plots for the magnitude of vertical velocity components (approximately $\mathcal{O}(10^{-4})$).

To determine whether the slight underestimations in the vertical components of the fluid velocity is a direct result of hybridization, both test cases were re-run using the discretization of Natale, Shipton, and Cotter (2016) (not shown). The resulting fields were identical to that of the ones produced here. Therefore, we can rule out any potential errors caused via hybridization. This may indicate issues with the nonlinear method, advection schemes, and/or treatment of the potential temperature perturbation when eliminating under the presence of orography. These test cases, along with revisiting the core Gusto-algorithm, is a subject of on-going investigation.

5.4.3 Density current

The last vertical slice test we consider was first presented by Straka et al. (1993). This test consists of a cold bubble suspended in a motionless isentropic atmosphere with $\theta = T_{\text{surf.}} = 300$ K. The profile of the bubble is initialized via

$$\delta\theta = \begin{cases} 0 & \text{if } r > 1, \\ \frac{15}{2}\left(\cos\left(\pi r\right) + 1\right) & \text{if } r \leq 1, \end{cases} \tag{5.93}$$

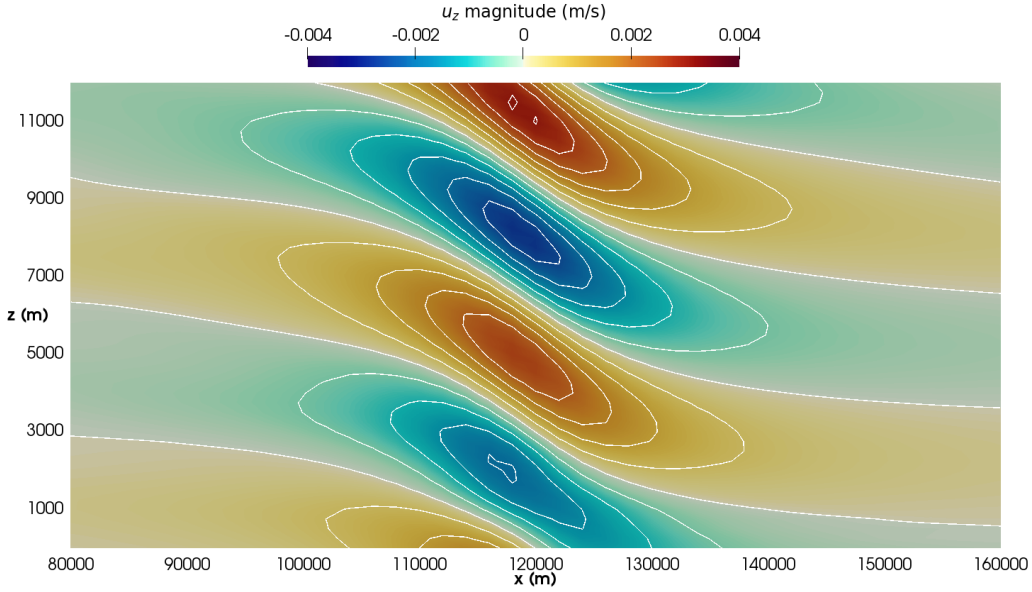
where

$$r = \sqrt{\left(\frac{x - x_c}{x_r}\right)^2 + \left(\frac{z - z_c}{z_r}\right)^2}, \tag{5.94}$$

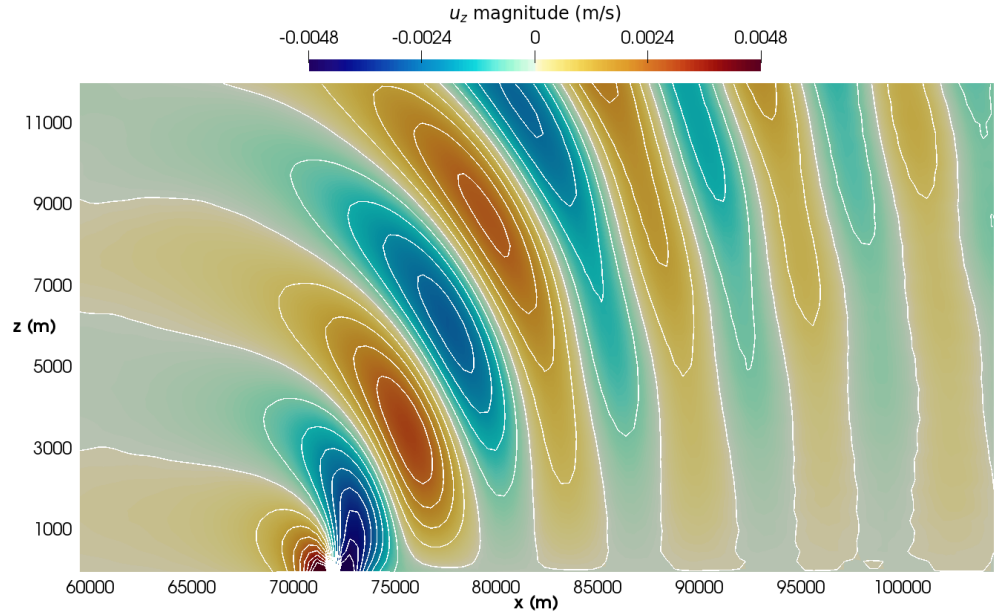
$x_c = \frac{L}{2}$, $x_r = 4000$ m, $z_c = 3000$ m, and $z_r = 2000$ m. The test case also features artificial diffusion using a dynamic viscosity $\nu = 75\text{m}^2\text{s}^{-1}$ to ensure convergence of the numerical solution. Driven by its negative buoyancy, the bubble falls to the bottom of the domain and should spread symmetrically in the horizontal direction, developing Kelvin-Helmholtz rotors.

We run this test using various levels of resolution, listed in Table 5.2. The initial bubble, along with the potential temperature perturbation and velocity fields at the final simulation time of $t = 900$ s are shown in Figures 5.4 and 5.5 at the finest resolution $\Delta x = 50$ m. The solution remains symmetric about the channel origin, indicating little to no phase errors in our numerical method.

Figures 5.6 and 5.7 display the potential temperature perturbation at the final simulation time across various resolutions. The front of the bubble forms at approximately 14637 m from the center, which is comparable to previous studies of this test



(A) The vertical velocity perturbation at $t = 15000$s for the hydrostatic test case.



(B) The vertical velocity perturbation at $t = 9000$s for the non-hydrostatic test case.

FIGURE 5.3: The vertical velocity perturbation for the hydrostatic mountain wave test at $t = 15000$s (Figure 5.3A) and the vertical velocity perturbation for the non-hydrostatic test case at $t = 9000$s (Figure 5.3B). Contours in the hydrostatic vertical velocity range from $[-4 \times 10^{-3}, 4 \times 10^{-3}]$ m/s, with intervals of 5×10^{-4} m/s centered around the 0 contour. For the non-hydrostatic case, contours range from $[-4.8 \times 10^{-3}, 4.8 \times 10^{-3}]$ in intervals of 6×10^{-4} m/s

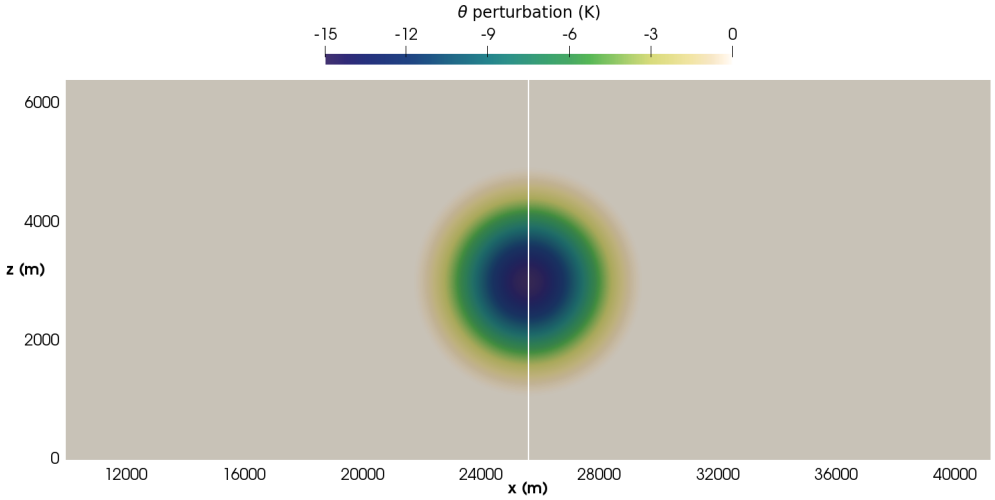
(Melvin et al., 2010; Melvin et al., 2019; Bao, Klöfkorn, and Nair, 2015; Li et al., 2013). Moreover, our numerical converges relatively rapidly in terms of front formation, as well as developing all three Kelvin-Helmholtz rotors at a relatively coarse resolution of $\Delta x = 200$ m.

Table 5.3 shows the minimum and maximum values of the potential temperature perturbation $\delta\theta_h$ after 900 seconds for all resolutions considered here. After $\Delta x = 200$ m, increasing the resolution dramatically reduces the overshoot in $\delta\theta_{\max}$, which

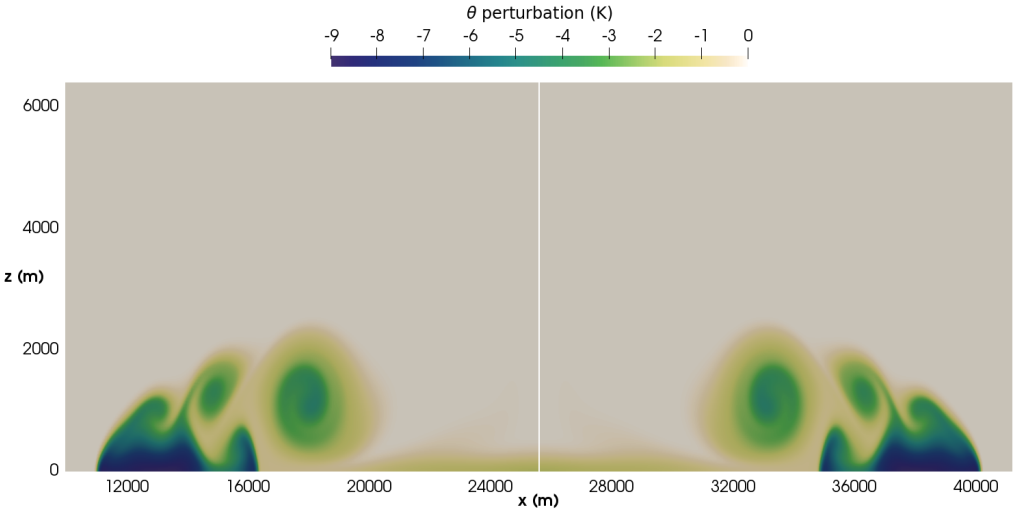
analytically should be 0 K. Increasing the resolution further and comparing with Straka et al. (1993), we observe comparable results with other models considered in the study. All fine-scale structures are fully developed by $\Delta x = 100$ m.

TABLE 5.3: Maximum and minimum values of $\delta\theta_h$ after 900 seconds across various resolutions.

Min and max values of $\delta\theta_h$		
Resolution (m)	$\delta\theta_{\min}$ (K)	$\delta\theta_{\max}$ (K)
800	-8.0464	0.8391
400	-10.1331	1.2621
200	-9.4668	0.6600
100	-8.9137	0.2686
50	-8.7686	0.0237



(A) The initial perturbation in the potential temperature.



(B) The potential temperature field at $t = 900$ s.

FIGURE 5.4: The temperature perturbation at $t = 0$s (5.4A), and $t = 900$ s (5.4B). A solid line is plotted at $x = 25600$m, the mid-point of the domain. The solution remains symmetric about the origin throughout the simulation ($\Delta x = 50$ m).

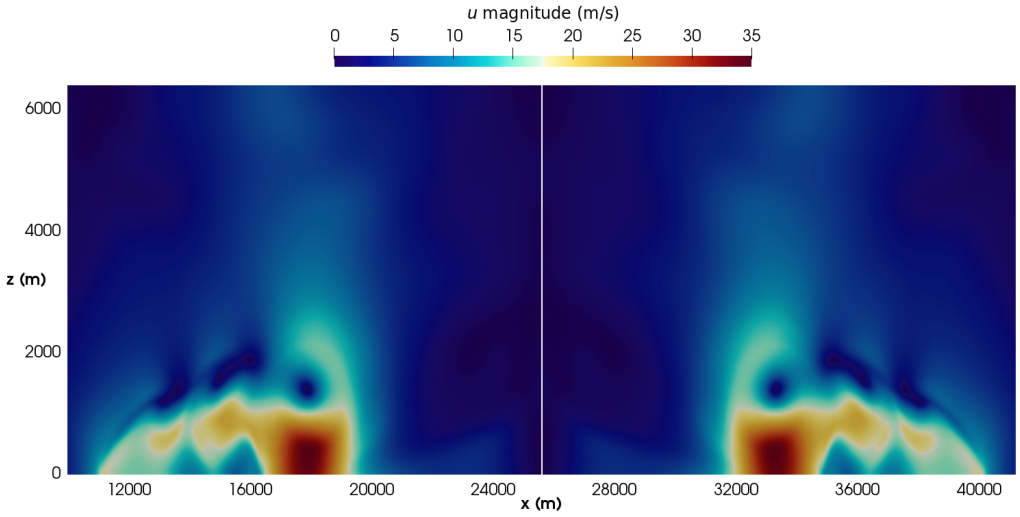
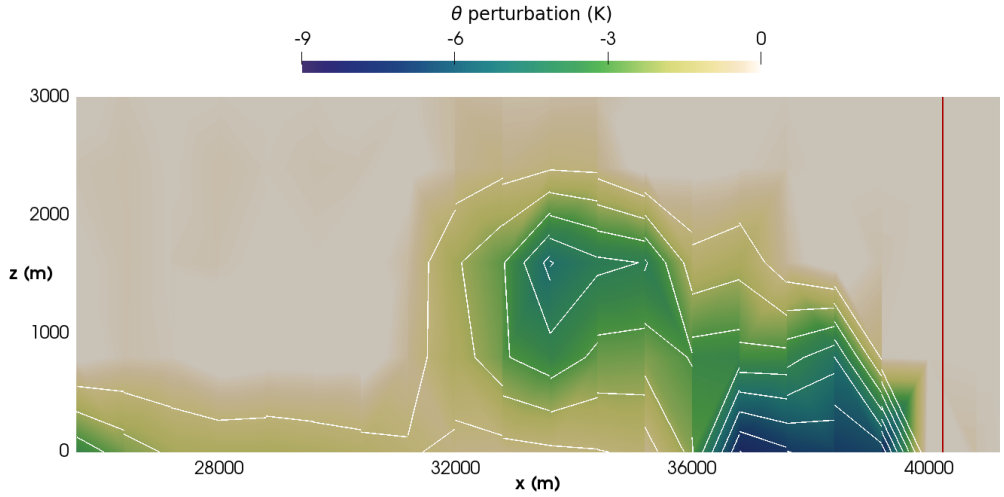
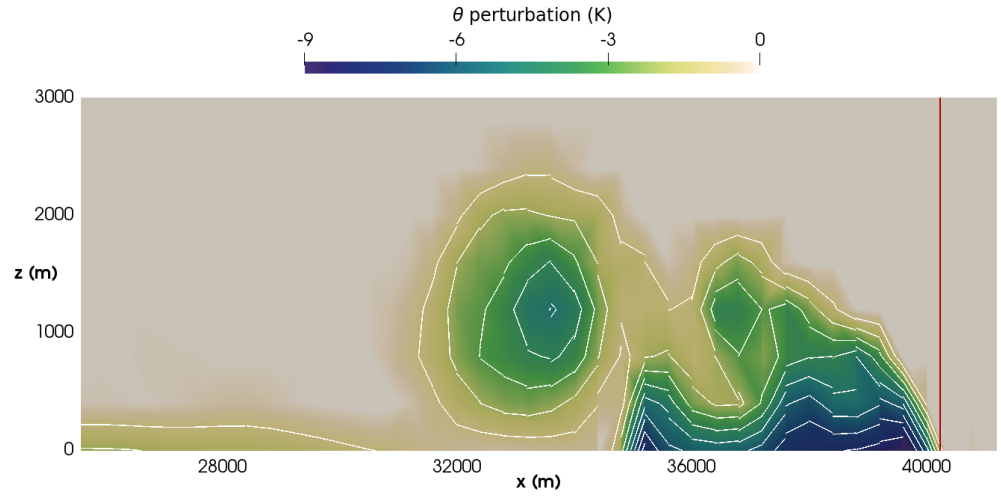


FIGURE 5.5: The velocity field at $t = 900$ s with a solid line plotted at $x = 25600$m ($\Delta x = 50$ m).

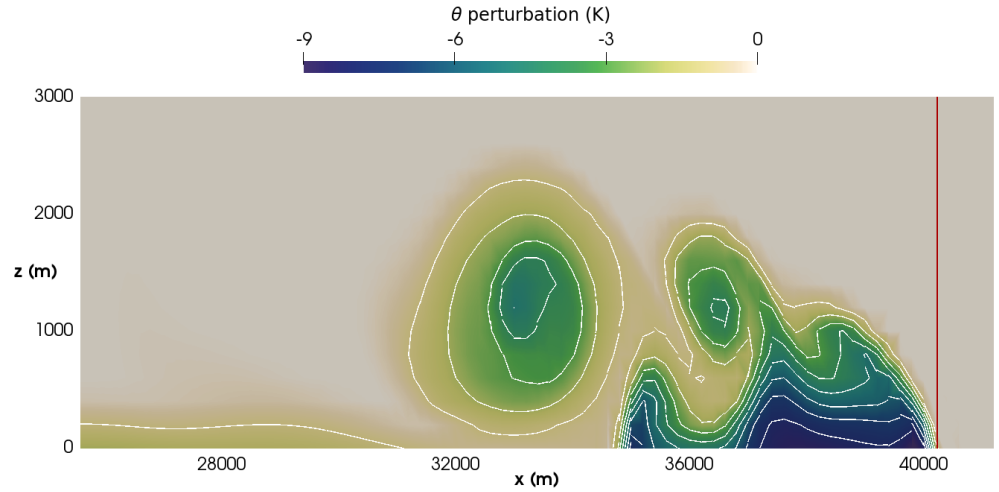


(A) $\Delta x = 800$ m.

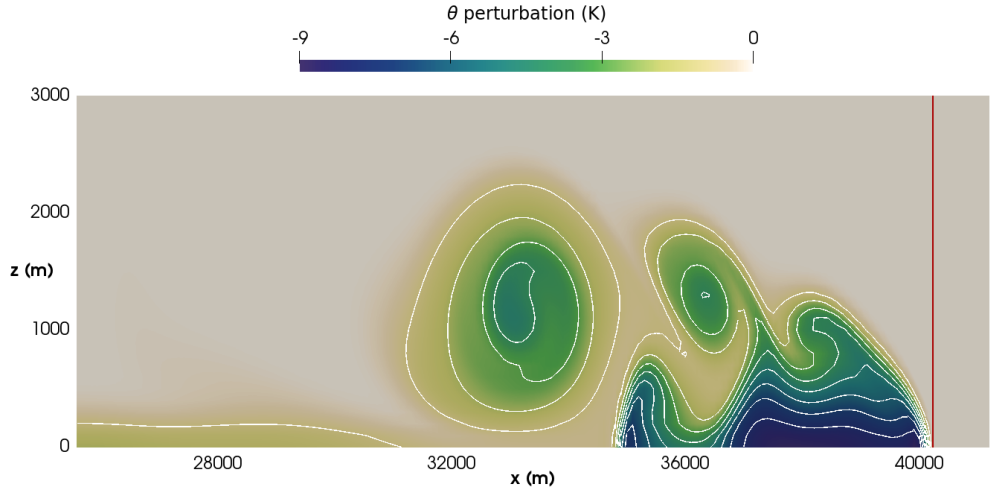


(B) $\Delta x = 400$ m.

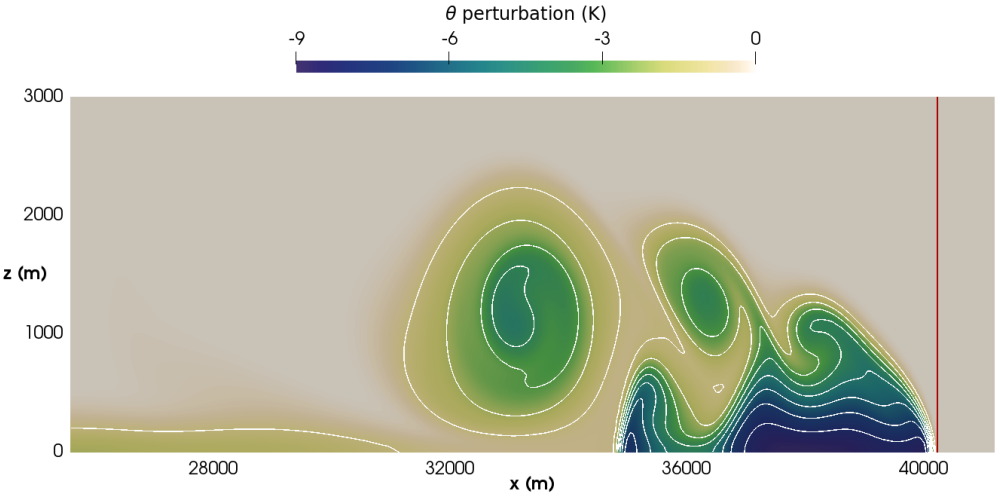
FIGURE 5.6: The potential temperature perturbation ($t = 900$ s) at the coarser resolutions: $\Delta x = 800$m and 400m. The vertical line denotes the front location at the finest resolution: $x = 40237$m (14637 m from the center). Contours range from -9 to -1 K in intervals of 1 K.



(A) $\Delta x = 200$ m.



(B) $\Delta x = 100$ m.



(C) $\Delta x = 50$ m.

FIGURE 5.7: The potential temperature perturbation ($t = 900$ s) at the finest resolutions: $\Delta x = 200$m, 100m, and 50m. The vertical line denotes the front location at the finest resolution solution: $x = 40237$m (14637 m from the center). Contours range from -9 to -1 K in intervals of 1 K.

5.4.4 Non-orographic gravity waves on a small planet

The final test case we consider is a full three-dimensional simulation of a gravity wave on a condensed planet. We run the model in a non-hydrostatic regime following test case 3.1 of the Dynamical Core Model Intercomparison Project (DCMIP) suite (Ullrich et al., 2012). The test consists of a balanced state in a solid body rotation with a mean-flow velocity $U = 20\text{ms}^{-1}$ on the equator. A localized potential temperature perturbation is added along the equator of the form:

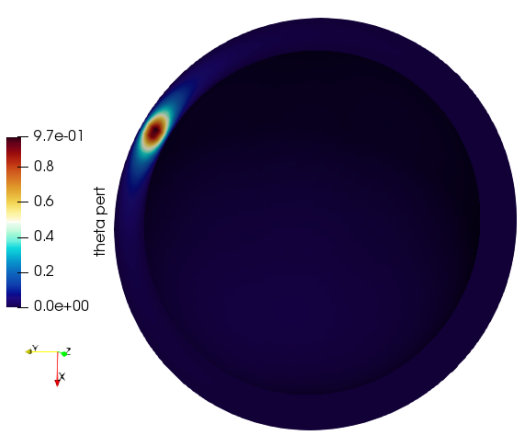
$$\delta\theta = \frac{d^2}{d^2 + r^2} \sin\left(\frac{\pi z}{H}\right), \tag{5.95}$$

where $d = 5000\text{m}$ is the width parameter for the perturbation, $H = 10\text{km}$ is the height of the atmospheric lid. The "great circle distance" r is defined via

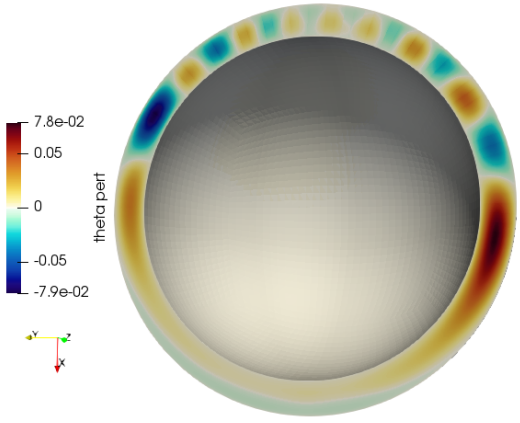
$$r = \frac{R}{X} \arccos\left(\sin(\phi_c)\sin(\phi) + \cos(\phi_c)\cos(\phi)\cos(\lambda - \lambda_c)\right), \tag{5.96}$$

where $\phi_c = 0$, $\lambda_c = 2\pi/3$ is the latitudal and longitudinal center-point of the perturbation respectively, $R = 6371\text{km}$, and $X = 125$ is the planet-radius reduction factor.

The horizontal grid is a cubed sphere consisting of 6,144 quadrilateral cells extruded upwards uniformly with 64 layers, for a total of 393,216 cuboid cells. We construct the finite element complex in three-dimensions using the lowest-order complex presented in Section 2.4.4 for cuboid elements. The initial perturbation field, as well as the field at $t = 3000$ s is presented in Figure 5.8. A time-step size of $\Delta t = 100$ s was used. The numerical results are similar to those provide by other models, which are publicly available as part of workshop hosted by The 2012 Dynamical Core Model Intercomparison Project: https://www.earthsystemcog.org/projects/dcmip-2012/.



(A) Initial temperature perturbation.



(B) Evolution of the potential temperature at $t = 3000$ s.

FIGURE 5.8: Plots of the potential temperature perturbation at $t = 0$ s and $t = 3000$ s for the non-orographic gravity wave test on a condensed planet.

TABLE 5.4: Vertical and horizontal spaces for the semi-implicit hybridizable system for the Euler equations, as well as total degree of freedom count (broken velocity, density, and Lagrange multipliers).

Compatible finite element discretizations						
Method	V_h^0	V_h^1	U_h^0	U_h^1	U_h^2	Hybrid. dofs
RT_1 (LO)	P_1	dP_0	P_1	RT_1	dP_0	2.8 M
RT_2 (NLO)	P_2	dP_1	P_2	RT_2	dP_1	13.1 M
$BDFM_2$ (NLO)	P_2	dP_1	$P_2 \oplus B_3$	$BDFM_2$	dP_1	13.7 M

Robustness against implicit Courant number

We now examine the iterative solver in a three-dimensional setting. Indeed, all the vertical slice examples from Sections 5.4.1–5.4.3 provided insight on our model reproducing relevant dynamics for select dynamical core tests, transitioning the model into three-dimensions requires solvers which are robust against parameters. Of particular interest is time-step size robustness in semi-implicit methods. In this section, we perform a stress-test on the hybridized equation for the Lagrange multipliers by increasing the horizontal implicit Courant number. For this experiment, the implicit Courant number ν is defined as

$$\nu = c_s \frac{\Delta t}{\Delta x}, \quad c_s = \sqrt{\frac{c_p T_0}{\gamma}}, \tag{5.97}$$

where c_s is the speed of sound in an air-parcel and Δx is the horizontal resolution. Here, c_p is the specific heat capacity of a dry atmosphere at constant pressure, $T_0 = 300$ K is the isothermal atmospheric temperature, and $\gamma = (1-\kappa)/\kappa$ with $\kappa = R_d/c_p$ denoting the ratio of the ideal gas constant R_d and c_p.

Computational domain and discretization spaces. The computational domain is a spherical annulus $\Omega = \mathcal{S}(R) \times [0, H]$, where $\mathcal{S}(R)$ is the surface of a sphere with radius $R = 6731$ km and $H = 10$ km is the height of the atmosphere. A mesh $\mathcal{T}_h$ of Ω is constructed by first generating a base mesh of $\mathcal{S}(R)$ consisting of 4 refinements of an icosahedron. This produces a base mesh with 5,120 triangular cells. Then $\mathcal{T}_h$ is obtained by vertically extruding the based mesh into 64 uniform levels, producing a mesh containing 327,680 triangular prism cells.

Using the one- and two-dimensional de-Rham complexes: $V_h^0 \xrightarrow{\partial_z} V_h^1$ and $U_h^0 \xrightarrow{\nabla^\perp\cdot} U_h^1 \xrightarrow{\nabla\cdot} U_h^2$. We construct the three-dimensional complex: $W_h^0 \xrightarrow{\nabla} W_h^1 \xrightarrow{\nabla\times} W_h^2 \xrightarrow{\nabla\cdot} W_h^3$ as before. Here, we consider three discretization methods: one lowest-order (LO) method and two next-to-lowest order (NLO) methods, constructed from the spaces in Table 5.4. Note that the finite element $BDFM_2$, while only mentioned briefly in Chapter 4, was shown to be a part of a compatible sequence by Cotter and Shipton (2012), with $U_h^0 = P_2 \oplus B_3$, where B_3 is a cubic "bubble" function vanishing on cell edges. For our purposes, we only need the latter two spaces in the complex.

Experimental setup. We construct a setup with background fields as in the DCMIP test case 3.1 (scaled to a full sized Earth), but with advection turned off entirely. This is due to the fact that we will run over a parameter range the exceeds the critical

advective CFL limit for our current choice set of advection equations. With no advection, we are simply testing the implicit solver. We invert the hybridizable system (5.85) onto a (seeded) randomly generated right-hand side, with entries sampled from a standard Gaussian distribution. We compute time-step sizes corresponding to increasing ν, as defined in (5.97). Note that typically atmospheric models run with time-steps such that $\nu = \mathcal{O}(2-10)$. In our setup, we consider values of ν up to 20.

Iterative solvers. We consider two solver strategies for inverting the Lagrange multiplier system. Both procedures are outlined here.

1. AMG($\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$): This preconditioning strategy follows exactly as in the linear gravity wave example in Section 4.5.3. The AMG method employed is PETSc's implementation of smoothed aggregation multigrid (GAMG), using a Krylov-accelerated smoother based on line relaxation.

    - Line relaxation is the optimal choice for smoothing in small aspect ration domains. As we have previously mentioned, the trace degrees of freedom are ordered column-wise (Bercea et al., 2016). We can therefore use PETSc's implementation of block ILU as a line-ILU method (Shapira, 2008, §11.6). In other words, block ILU-based smoothing (Wesseling and Oosterlee, 2001; Stevenson, 1994; Wittum, 1989) behaves like a line relaxation method for the trace system.
    - ILU is used in conjunction with non-restarted GMRES due to the non-symmetry of the trace operator. Krylov-accelerated smoothing is more often used for both non-symmetric or indefinite systems, as discussed by Birken, Bull, and Jameson (2016) and Elman, Ernst, and O'Leary (2001).
    - We denote by $\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$ as k iterations of preconditioned GMRES using the block ILU method outlined above.

2. AMG($\boldsymbol{\mathcal{R}}(k)$): Our second strategy also uses a smoothed aggregation multigrid approach. However, instead of Krylov-based smoothing, we consider a more common approach in atmospheric modeling. We construct smoothers based on the Richardson iteration (Richardson, 1911).

    - The Richardson iteration is one of the simplest choices of classical iterative methods. For a matrix system $\boldsymbol{S\Lambda} = \boldsymbol{R}$, the *preconditioned* Richardson iteration takes the form:

    $$\boldsymbol{\Lambda}^{(i+1)} = (\boldsymbol{I} - \omega \boldsymbol{P}^{-1}\boldsymbol{S})\boldsymbol{\Lambda}^{(i)} + \omega \boldsymbol{P}^{-1}\boldsymbol{R}, \quad i \geq 1, \tag{5.98}$$

    where ω is a relaxation parameter and $\boldsymbol{P}$ is a preconditioner. The optimal choice of ω is explored by Smolarkiewicz and Margolin (1994), but produces a parameter $\omega = \omega_i$ that depends on the iterates $\boldsymbol{\Lambda}^{(i)}$.
    - Line relaxation based on Richardson iterations has been demonstrated to be highly effective for accelerating Krylov methods (like GCR) for non-symmetric elliptic operators (particularly for geophysical flows) (Smolarkiewicz and Margolin, 1994; Thomas et al., 2003). We therefore consider the Richardson method preconditioned with block ILU.

- $\boldsymbol{\mathcal{R}}(k)$ denotes k ILU-preconditioned Richardson iterations. In our implementation, we take $\omega = 0.8$ in (5.98). This was heuristically determined based on experimentation.

In this experiment, we use a flexible variant of GMRES on the trace system. Both AMG($\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$) and AMG($\boldsymbol{\mathcal{R}}(k)$) are used as preconditioners to accelerate the convergence of the Krylov method, which is set to terminate when the residual is reduced by a factor of 10^6. We consider two values of k for the AMG smoothers $\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$ and $\boldsymbol{\mathcal{R}}(k)$, $k = 3$ and $k = 5$. The rationale behind this decision is to specifically examine the effectiveness of each preconditioner given different varying levels of smoothing.

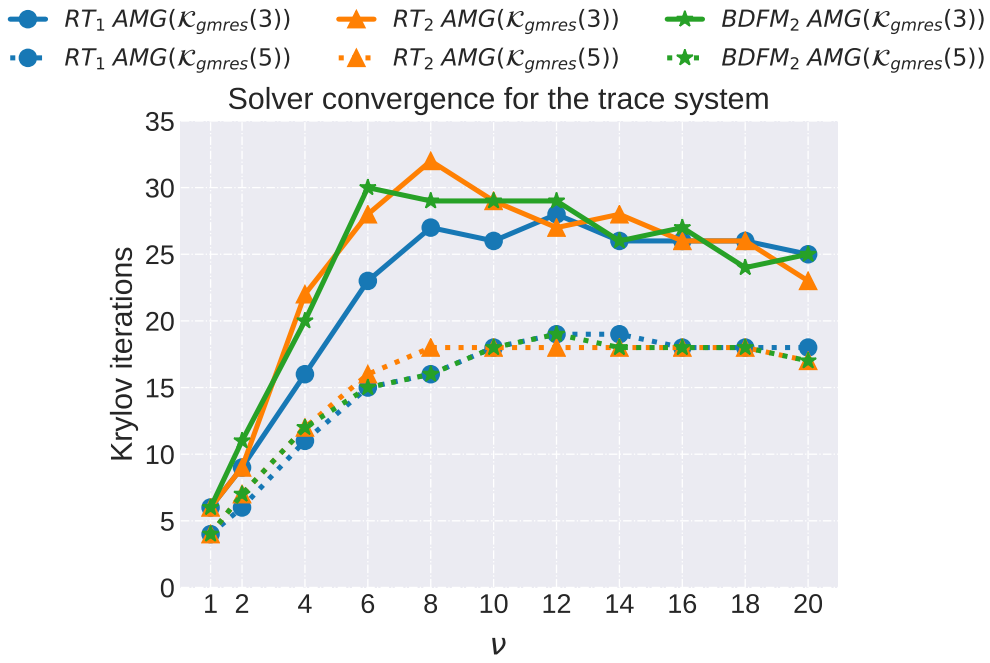
All computations were performed on a fully-loaded compute node of dual-socket Intel E5-2630v4 (Xeon) processors running at 2.2GHz, utilizing a total of 40 MPI processes (2×10 cores with 2 threads per core). Figure 5.9 displays the results of this experiment for increasing ν and all choices of spatial discretizations in Table 5.4. For all cases, a steady increase in iterations throughout the Courant numbers $\nu = \mathcal{O}(1-6)$ is observed. The solver plateaus in iteration count after $\nu = 6$, maintaining convergence behavior up to $\nu = 20$. This demonstrates good solver robustness with respect to Δt. This behavior is further improved by increasing the number of smoothing iterations.

While the spectral properties of the trace operator are not rigorously understood at this time, our results indicate that multigrid is a feasible preconditioning strategy. This warrants further investigation. In particular, the design of optimal smoothing strategies is critical for the success of multigrid. Our findings are promising and have provided some insight on the effectiveness of Krylov-accelerated smoothing and standard line relaxation methods.

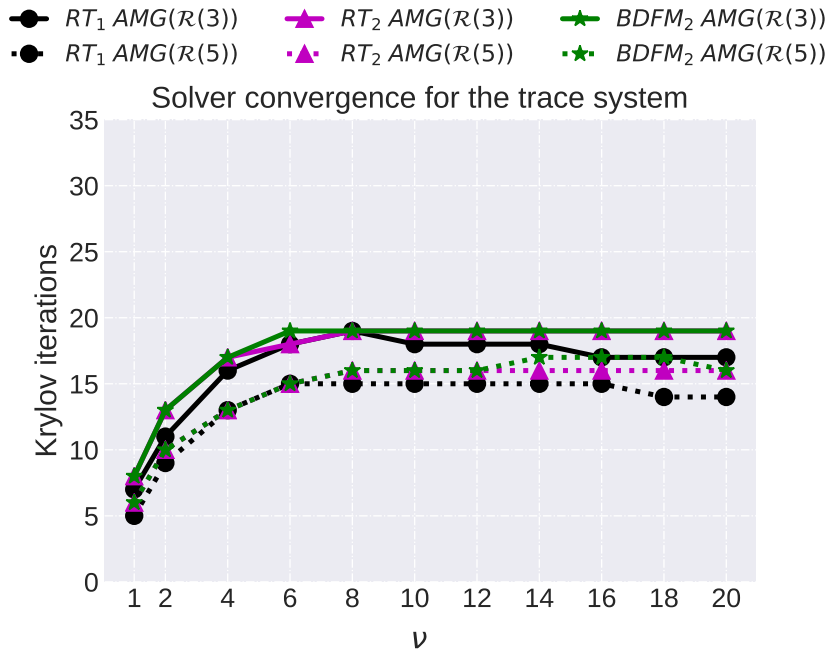
When using Krylov-enhanced smoothers, we observe that the convergence of the outer Krylov method is more sporadic for AMG($\boldsymbol{\mathcal{K}}_{\text{gmres}}(3)$). This is remedied by increasing the number of smoothing iterations ($\boldsymbol{\mathcal{K}}_{\text{gmres}}(5)$), as variations in outer iterations remains far more controlled. This suggests that 3 GMRES-smoothing iterations doesn't sufficiently produce a smooth enough error for the coarse-grid correction to handle effectively. This in turn requires the outer Krylov method to iterate further in order to control the reduction in the problem residual.

In contrast, the convergence behavior using Richardson smoothers in Figure 5.9B is far more controlled (compare with Figure 5.9A). Overall, AMG($\boldsymbol{\mathcal{R}}(k)$) requires less outer iterations to reach convergence. This suggests that the Richardson smoothers are more effective for removing the high-frequency errors. Due to the simplicity of the Richardson method, this is perhaps a far more viable alternative over Krylov-based smoothing. It is worth noting that the results shown in Figure 5.9B depend on the choice of relaxation parameter ω. For $\omega \approxeq 1$ and $\omega < 0.5$, the total number of outer Krylov iterations noticeably increased. It was heuristically determined by experimentation that $\omega \approx 0.8$ produced consistently better results in terms of solver convergence.

There is currently increased interest in *geometric* multigrid approaches (GMG) rather than purely algebraic methods. The approach by Gopalakrishnan and Tan (2009) is particularly interesting, as this involves coarsening the trace problem to a P_1 continuous Lagrange finite element space, where the GMG strategies of Müller and Scheichl (2014) can be applied. A more concentrated study on iterative methods and geometric multigrid techniques for the trace system (5.86) is currently being investigated.



(A) Courant number parameter test for AMG($\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$), $k = 3, 5$



(B) Courant number parameter test for AMG($\boldsymbol{\mathcal{R}}(k)$), $k = 3, 5$

FIGURE 5.9: Number of Krylov iterations to invert the trace system versus horizontal Courant number for AMG($\boldsymbol{\mathcal{K}}_{\text{gmres}}(k)$) (5.9A) and AMG($\boldsymbol{\mathcal{R}}(k)$) (5.9B). Flexible GMRES is used as the outer solver for the trace system and terminates when the residual is reduced by a factor of 10^6.

5.5 Chapter summary

In this chapter, we introduced a new hybridizable method for the linearized Euler equations obtained from a semi-implicit discretization, a common approach in operational dynamical cores. The formulation permits the local elimination of prognostic variables to produce a condensed equation for a new unknown on the mesh skeleton. Unlike the previous hybridizable methods described in Chapters 3 and 4, the formulation in Section 5.3 reintroduces the pressure correction term, which was previously not directly included as an independent unknown. Additionally, the hybridizable formulation actually *simplifies* the mixed system for the linear perturbations due to the introduction of the Lagrange multipliers, eliminating several lower-order surface terms coupling velocity and density unknowns in (5.47)–(5.48).

The software framework in Chapter 4 allows for the rapid implementation of this new hybridizable method. Due to the composable nature of the Slate-based static condensation interfaces, we are able to rapidly prototype various solver configurations to better understand the numerical properties of the condensed equation for the pressure trace. With that in mind, we remark here that this work has opened many different avenues of possible research. The UK Met Office is currently exploring the application of these new hybridization techniques for the equation sets considered in this chapter.

There are currently unanswered performance questions about this new hybridizable method, though much of the mechanical aspects of the algorithm (static condensation, local recovery) can be inferred from the results of Chapter 4. Ultimately, the question about scalable solution methods for the Lagrange multiplier systems must be investigated further. The extension of the hybridizable method in Section 5.3 to a moist atmosphere has been successfully done in Bendall et al. (2019).

6 Summary and outlook

6.1 Summary and conclusions

In this dissertation, we have studied the application of hybridization to the compatible finite element formulations of various equation sets relevant for geophysical flows. We summarize here our findings here.

We started in Chapter 2 by providing standard systems of PDEs which commonly arise in atmospheric (and occasionally ocean) applications. Emphasis was placed on familiarity with common definitions used in context. Next, a formal definition of the finite element was provided. From that definition, various element families corresponding to different classes of Sobolev spaces. From there, we connected the finite element families with the discrete de-Rham complexes in one-, two-, and three-dimensional settings. This led to a formal definition of a *compatible finite element* discretization; by selecting the appropriate sequence of spaces, deriving the finite element formulation produces the desired compatible method. We concluded the chapter with constructions of three-dimensional complexes using tensor product finite elements, a particularly useful construction for atmospheric-shaped domains.

In Chapter 3, we formally introduce the hybridization technique for a linear shallow water model. The purpose of this is to build up for the analysis of the hybridized variable λ_h. In Section 3.2, we analyzed the hybridizable formulation of the linear shallow water equations. The conclusions that we made were the following:

- The hybridization of the shallow water equations defines a well-posed discrete system;
- The local solvers defining the hybridizable method are well-posed;
- The solutions of the hybridizable method are also the solutions to the original compatible finite element discretization;
- The Lagrange multiplier λ_h is the unique solution to a variational problem defined on the mesh skeleton;
- The prognostic variables can be expressed as *local* lifting operators which reconstructs the variables in each cell using data obtained from either λ_h or right-hand side function. Mechanically, this is what the element-wise recovery encapsulates.

The fact that the eliminated variables can be recovered locally is due to the fact that the equations for the local solvers in Section 3.2.1 are well-posed. We then used the hybridizable method within a nonlinear method for the full shallow water equations and provided numerical results from standard test cases on the sphere. We then concluded by constructing new hybridizable formulations of the hydrostatic equation and a linear gravity wave system in Section 3.4.

No mention of the implementation was given in Section 3.3, as the purpose of those numerical results were to provide verification within the context of well-documented test cases. In Chapter 4, we introduce a new domain-specific abstraction which enables all the hybridization procedures in this dissertation. The language, Slate, allows for the concise representation of the local linear algebra operations that is required of solving hybridizable systems. Using Firedrake and its strong composition with the PETSc solver library, we develop Slate-based preconditioner interfaces which allows for on-demand and runtime-configurable static condensation solvers. These precondtioners may be arbitrarily composed with existing precondtioners, and the linear system for the Lagrange multipliers has access to all internal and external solver libraries that PETSc provides. This facilitates rapid proto-typing of solver strategies for the trace system. In Section 4.4.4, we summarize preconditioning strategies for the Lagrange multiplier system using algebraic multigrid methods.

The performance of hybridization was examined in Section 4.5, using Slate-generated kernels for local linear algebra. We demonstrated that hybridizable methods greatly outperforms standard mixed methods in terms of time-to-solution. For the three-dimensional gravity wave experiment, we also showed that hybridization allows for a more parameter-robust solution approach. This is especially apparent when implicitly incorporating the Coriolis term.

Finally, Chapter 5 presents a new hybridizable formulation of the linearized perturbation system for the compressible Euler equations. The main difference compared with all previous hybridization methods is that the hybridizable Euler system reintroduced a separate independent unknown which is not the trace of any prognostic variable in the original mixed system. This already opens up many question regarding the nature of the Lagrange multipliers for that system. We presented standard vertical slice tests using the Gusto dynamical core with the new hybridizable solver. While there were some minor differences in the test case results when compared to ENDGame, the results were comparable to many other existing models using different numerics. A three-dimensional example was provided in Section 5.4.4, which included a parameter-robustness test for the hybridizable solver. Our initial findings suggest that the trace system can be efficiently solved using multigrid techniques.

6.2 Further work

There are many potential areas of future work concerning the work presented in this dissertation.

- **Spectral properties of the trace system**: The hybridization of the linearized shallow water equations in Section 3.2 provides a strong starting point for examining the discrete operator even further. It will therefore be interesting to extend these ideas to the hybridizable methods presented in Section 3.4. For each of these systems, a rigorous proof demonstrating the spectral equivalence of the trace operator with a known elliptic operator would be a powerful result. The idea that such an equivalence can be made is inspired by the findings of Gopalakrishnan (2003).
- **Optmization of dense linear algebra**: Slate as described in Chapter 4 is a continuously evolving piece of software, adapting based on research demands.

Slate-generated kernels can and should be improved. To exploit different architectures, such as accelerators (GPUs), batched or vectorized linear algebra should be incorporated in Slate's compiler. In doing so, we can make good use of the batched linear algebra routines being developed in LAPACK/BLAS (Haidar et al., 2015; Abdelfattah et al., 2016).

- **Analysis of the hybridizable Euler system:** The hybridizable Euler system, while numerically stable, is relatively unknown in terms of the analysis. Currently, a characterization theorem akin to that of Section 3.2 is being developed for the hybridizable Euler system. This will require the following
  - Construction of the local solvers and their well-posedness;
  - A characterization theorem for λ_h that relates the prognostic variables (velocity and density) with the local solvers;
  - Analysis of the variational problem corresponding to λ_h.

  On the latter point, a couple ideas to consider are the following. Since λ_h is an approximation of the pressure-correction, $\delta\Pi$ on the skeleton, then perhaps there is some level of spectral equivalence with the reduced trace operator and the non-symmetric Helmholtz operator for $\delta\Pi$ in staggered finite-difference models (Wood et al., 2014; Thuburn, 2016). Moreover, a proof of the ellipticity of the trace operator would be a strong result for this case. It would further validate the behavior of multigrid preconditioning observed in Section 5.4.4.

- **Non-nested multigrid preconditioning:** It was shown for mixed formulations of elliptic equations that the hybridized variable λ_h could be solved by solving a corresponding P_1 continuous Galerkin system using a non-nest multigrid algorithm (Gopalakrishnan and Tan, 2009). The extension of this multigrid method for a shallow water system using an HDG method is detailed in a paper in preparation based on the work of Cockburn et al. (2013). A natural extension of this multigrid approach for the hybridization of compatible finite element discretizations is a topic of on-going interest.

Bibliography

Abdelfattah, A., A. Haidar, S. Tomov, and J. Dongarra (2016). "Performance, Design, and Autotuning of Batched GEMM for GPUs". In: *Lecture Notes in Computer Science*. Springer International Publishing, pp. 21–38. DOI: 10.1007/978-3-319-41321-1_2.

Adams, M., M. Brezina, J. Hu, and R. Tuminaro (2003). "Parallel multigrid smoothing: polynomial versus Gauss–Seidel". In: *Journal of Computational Physics* 188.2, pp. 593–610. DOI: 10.1016/s0021-9991(03)00194-3.

Adams, S. V., R. W. Ford, M. Hambley, J. M. Hobson, I. Kavčič, C. M. Maynard, T. Melvin, E. H. Müller, S. Mullerworth, A. R. Porter, M. Rezny, B. J. Shipway, and R. Wong (2019). "LFRic: Meeting the challenges of scalability and performance portability in Weather and Climate models". In: *Journal of Parallel and Distributed Computing* 132, pp. 383–396. DOI: 10.1016/j.jpdc.2019.02.007.

Alnæs, M. S., A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells (2014). "Unified form language: A domain-specific language for weak formulations of partial differential equations". In: *ACM Transactions on Mathematical Software* 40.2, pp. 1–37. DOI: 10.1145/2566630.

Amestoy, P. R., I. S. Duff, and J. Y. L'Excellent (2000). "Multifrontal parallel distributed symmetric and unsymmetric solvers". In: *Computer Methods in Applied Mechanics and Engineering* 184.2-4, pp. 501–520. DOI: 10.1016/s0045-7825(99)00242-x.

Arakawa, A. and C. S. Konor (1996). "Vertical Differencing of the Primitive Equations Based on the Charney–Phillips Grid in Hybrid σ–p Vertical Coordinates". In: *Monthly Weather Review* 124.3, pp. 511–528. DOI: 10.1175/1520-0493(1996)124<0511:VDOTPE>2.0.CO;2.

Arakawa, A. and V. R. Lamb (1977). "Computational Design of the Basic Dynamical Processes of the UCLA General Circulation Model". In: *Methods in Computational Physics: Advances in Research and Applications*. Elsevier, pp. 173–265. DOI: 10.1016/b978-0-12-460817-7.50009-4.

Arbogast, T. and Z. Chen (1995). "On the implementation of mixed methods as nonconforming methods for second-order elliptic problems". In: *Mathematics of Computation* 64.211, pp. 943–943. DOI: 10.1090/s0025-5718-1995-1303084-8.

Argyris, J. H. (1955). "Energy theorems and structural analysis: a generalized discourse with applications on energy principles of structural analysis including the effects of temperature and non-linear stress-strain relations part I. General theory". In: *Aircraft Engineering and Aerospace Technology* 27.4, pp. 125–134. DOI: 10.1108/eb032545.

Argyris, J. H., I. Fried, and D. W. Scharpf (1968). "The TUBA Family of Plate Elements for the Matrix Displacement Method". In: *The Aeronautical Journal* 72.692, pp. 701–709. DOI: 10.1017/s000192400008489x.

Arnold, D. N. (1982). "An Interior Penalty Finite Element Method with Discontinuous Elements". In: *SIAM Journal on Numerical Analysis* 19.4, pp. 742–760. DOI: 10.1137/0719052.

– (2013). "Spaces of Finite Element Differential Forms". In: *Analysis and Numerics of Partial Differential Equations*. Springer Milan, pp. 117–140. DOI: 10.1007/978-88-470-2592-9_9.

Arnold, D. N. and G. Awanou (2014). "Finite element differential forms on cubical meshes". In: *Mathematics of Computation* 83.288, pp. 1551–1570. DOI: 10.1090/s0025-5718-2013-02783-4.

Arnold, D. N., D. Boffi, and F. Bonizzoni (2015). "Finite element differential forms on curvilinear cubic meshes and their approximation properties". In: *Numerische Mathematik* 129.1, pp. 1–20. DOI: 10.1007/s00211-014-0631-3.

Arnold, D. N. and F. Brezzi (1985). "Mixed and nonconforming finite element methods : implementation, postprocessing and error estimates". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 19.1, pp. 7–32. DOI: 10.1051/m2an/1985190100071.

Arnold, D. N., R. S. Falk, and R. Winther (2000). "Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$". In: *Numerische Mathematik* 85.2, pp. 197–217. DOI: 10.1007/pl00005386.

– (2006). "Finite element exterior calculus, homological techniques, and applications". In: *Acta Numerica* 15, pp. 1–155. DOI: 10.1017/s0962492906210018.

– (2010). "Finite element exterior calculus: from Hodge theory to numerical stability". In: *Bulletin of the American Mathematical Society* 47.2, pp. 281–354. DOI: 10.1090/s0273-0979-10-01278-4.

Arnold, D. N. and A. Logg (2014). "Periodic table of the finite elements". In: *SIAM News* 47.9, p. 212.

Arnold, D. N., F. Brezzi, B. Cockburn, and L. D. Marini (2002). "Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 39.5, pp. 1749–1779. DOI: 10.1137/s0036142901384162.

Baker, A. H., R. D. Falgout, T. Gamblin, T. V. Kolev, M. Schulz, and U. M. Yang (2011). "Scaling Algebraic Multigrid Solvers: On the Road to Exascale". In: *Competence in High Performance Computing 2010*. Springer Berlin Heidelberg, pp. 215–226. DOI: 10.1007/978-3-642-24025-6_18.

Balay, S., W. D. Gropp, L. C. McInnes, and B. F. Smith (1997). "Efficient Management of Parallelism in Object-Oriented Numerical Software Libraries". In: *Modern Software Tools for Scientific Computing*. Birkhäuser Boston, pp. 163–202. DOI: 10.1007/978-1-4612-1986-6_8.

Balay, S., S. Abhyankar, M. Adams, P. Brune, K. Buschelman, L. Dalcin, W. D. Gropp, B. F. Smith, D. Karpeyev, D. Kaushik, L. C. McInnes, K. Rupp, H. Zhang, and S. Zampini (2016). *PETSc Users Manual Revision 3.7*. Tech. rep. DOI: 10.2172/1255238.

Bao, L., R. Klöfkorn, and R. D. Nair (2015). "Horizontally Explicit and Vertically Implicit (HEVI) Time Discretization Scheme for a Discontinuous Galerkin Nonhydrostatic Model". In: *Monthly Weather Review* 143.3, pp. 972–990. DOI: 10.1175/mwr-d-14-00083.1.

Bauer, P., A. Thorpe, and G. Brunet (2015). "The quiet revolution of numerical weather prediction". In: *Nature* 525.7567, pp. 47–55. DOI: 10.1038/nature14956.

Bauer, W. and C. J. Cotter (2018). "Energy–enstrophy conserving compatible finite element schemes for the rotating shallow water equations with slip boundary conditions". In: *Journal of Computational Physics* 373, pp. 171–187. DOI: 10.1016/j.jcp.2018.06.071.

Bendall, T. M., T. H. Gibson, J. Shipton, C. J. Cotter, and B. J. Shipway (2019). *A Compatible Finite Element Discretisation for the Moist Compressible Euler Equations*. arXiv: 1910.01857 [math.NA].

Benzi, M., G. H. Golub, and J. Liesen (2005). "Numerical solution of saddle point problems". In: *Acta Numerica* 14, pp. 1–137. DOI: 10.1017/s0962492904000212.

Bercea, G.-T., A. T. T. McRae, D. A. Ham, L. Mitchell, F. Rathgeber, L. Nardi, F. Luporini, and P. H. J. Kelly (2016). "A structure-exploiting numbering algorithm for finite elements on extruded meshes, and its performance evaluation in Firedrake". In: *Geoscientific Model Development* 9.10, pp. 3803–3815. DOI: 10.5194/gmd-9-3803-2016.

Birken, P., J. Bull, and A. Jameson (2016). "A study of multigrid smoothers used in compressible CFD based on the convection diffusion equation". In: *Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016)*. Institute of Structural Analysis and Antiseismic Research School of Civil Engineering National Technical University of Athens (NTUA) Greece. DOI: 10.7712/100016.1987.7289.

Boffi, D., F. Brezzi, and M. Fortin (2013). *Mixed finite element methods and applications*. Springer International Publishing. ISBN: 978-3-642-36519-5. DOI: 10.1007/978-3-642-36519-5.

Boffi, D., F. Brezzi, L. F. Demkowicz, R. G. Durán, R. S. Falk, and M. Fortin (2008). *Mixed finite elements, compatibility conditions, and applications*. Springer International Publishing. ISBN: 978-3-540-78319-0. DOI: 10.1007/978-3-540-78319-0.

Börm, S. and R. Hiptmair (2001). "Analysis of tensor product multigrid". In: *Numerical Algorithms* 26.3, pp. 219–234. DOI: 10.1023/A:1016686408271.

Bramble, J. H., D. Y. Kwak, and J. E. Pasciak (1994). "Uniform Convergence of Multigrid V-Cycle Iterations for Indefinite and Nonsymmetric Problems". In: *SIAM Journal on Numerical Analysis* 31.6, pp. 1746–1763. DOI: 10.1137/0731089.

Bramble, J. H., J. E. Pasciak, and J. Xu (1988). "The analysis of multigrid algorithms for nonsymmetric and indefinite elliptic problems". In: *Mathematics of Computation* 51.184, pp. 389–389. DOI: 10.1090/s0025-5718-1988-0930228-6.

Bramble, J. H. and J. Xu (1989). "A Local Post-Processing Technique for Improving the Accuracy in Mixed Finite-Element Approximations". In: *SIAM Journal on Numerical Analysis* 26.6, pp. 1267–1275. DOI: 10.1137/0726073.

Brenner, S. C. and R. Scott (2008). *The Mathematical Theory of Finite Element Methods*. Springer International Publishing. ISBN: 978-0-387-75934-0. DOI: 10.1007/978-0-387-75934-0.

Brezzi, F., J. Douglas, and L. D. Marini (1985). "Two families of mixed finite elements for second order elliptic problems". In: *Numerische Mathematik* 47.2, pp. 217–235. DOI: 10.1007/bf01389710.

Brezzi, F. and M. Fortin (1991). *Mixed and hybrid finite element methods*. Springer International Publishing. ISBN: 978-1-4612-3172-1. DOI: 10.1007/978-1-4612-3172-1.

Brezzi, F., J. Douglas, M. Fortin, and L. D. Marini (1987a). "Efficient rectangular mixed finite elements in two and three space variables". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 21.4, pp. 581–604. DOI: 10.1051/m2an/1987210405811.

Brezzi, F., J. Douglas, R. Durán, and M. Fortin (1987b). "Mixed finite elements for second order elliptic problems in three variables". In: *Numerische Mathematik* 51.2, pp. 237–250. DOI: 10.1007/bf01396752.

Brooks, A. N. and T. J.R. Hughes (1982). "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations". In: *Computer Methods in Applied Mechanics and Engineering* 32.1-3, pp. 199–259. DOI: 10.1016/0045-7825(82)90071-8.

Brown, J., M. G. Knepley, D. A. May, L. C. McInnes, and B. F. Smith (2012). "Composable Linear Solvers for Multiphysics". In: *2012 11th International Symposium on Parallel and Distributed Computing*. IEEE. DOI: 10.1109/ispdc.2012.16.

Brunner, T. A. and T. V. Kolev (2011). "Algebraic Multigrid for Linear Systems Obtained by Explicit Element Reduction". In: *SIAM Journal on Scientific Computing* 33.5, pp. 2706–2731. DOI: 10.1137/100801640.

Bryan, G. H. and J. M. Fritsch (2002). "A Benchmark Simulation for Moist Nonhydrostatic Numerical Models". In: *Monthly Weather Review* 130.12, pp. 2917–2928. DOI: 10.1175/1520-0493(2002)130<2917:ABSFMN>2.0.CO;2.

Buckeridge, S. and R. Scheichl (2010). "Parallel geometric multigrid for global weather prediction". In: *Numerical Linear Algebra with Applications* 17.2-3, pp. 325–342. DOI: 10.1002/nla.699.

Charney, J. G. and N. A. Phillips (1953). "Numerical Integration of The Quasi-Geostrophic Equations For Barotropic and Simple Baroclinic Flows". In: *Journal of Meteorology* 10.2, pp. 71–99. DOI: 10.1175/1520-0469(1953)010<0071:NIOTQG>2.0.CO;2.

Choi, S. J., F. X. Giraldo, J. Kim, and S. Shin (2014). "Verification of a non-hydrostatic dynamical core using the horizontal spectral element method and vertical finite difference method: 2-D aspects". In: *Geoscientific Model Development* 7.6, pp. 2717–2731. DOI: 10.5194/gmd-7-2717-2014.

Ciarlet, P. G. (2002). "Introduction to the Finite Element Method". In: *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, pp. 36–109. DOI: 10.1137/1.9780898719208.ch2.

Clough, R. W. (1960). "The Finite Element Method in Plane Stress Analysis". In: *Proceedings of 2nd ASCE Conference on Electronic Computation*.

Cockburn, B. (2016). "Static Condensation, Hybridization, and the Devising of the HDG Methods". In: *Lecture Notes in Computational Science and Engineering*. Springer International Publishing, pp. 129–177. DOI: 10.1007/978-3-319-41640-3_5.

Cockburn, B. and J. Gopalakrishnan (2004). "A Characterization of Hybridized Mixed Methods for Second Order Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 42.1, pp. 283–301. DOI: 10.1137/s0036142902417893.

– (2005). "Error analysis of variable degree mixed methods for elliptic problems via hybridization". In: *Mathematics of Computation* 74.252, pp. 1653–1678. DOI: 10.1090/s0025-5718-05-01741-2.

Cockburn, B., J. Gopalakrishnan, and R. Lazarov (2009). "Unified Hybridization of Discontinuous Galerkin, Mixed, and Continuous Galerkin Methods for Second Order Elliptic Problems". In: *SIAM Journal on Numerical Analysis* 47.2, pp. 1319–1365. DOI: 10.1137/070706616.

Cockburn, B., J. Gopalakrishnan, and F.-J. Sayas (2010). "A projection-based error analysis of HDG methods". In: *Mathematics of Computation* 79.271, pp. 1351–1367. DOI: 10.1090/s0025-5718-10-02334-3.

Cockburn, B., J. Guzmán, and H. Wang (2009). "Superconvergent discontinuous Galerkin methods for second-order elliptic problems". In: *Mathematics of Computation* 78.265, pp. 1–1. DOI: 10.1090/s0025-5718-08-02146-7.

Cockburn, B., J. Gopalakrishnan, F. Li, N.-C. Nguyen, and J. Peraire (2010). "Hybridization and Postprocessing Techniques for Mixed Eigenfunctions". In: *SIAM Journal on Numerical Analysis* 48.3, pp. 857–881. DOI: 10.1137/090765894.

Cockburn, B., O. Dubois, J. Gopalakrishnan, and S. Tan (2013). "Multigrid for an HDG method". In: *IMA Journal of Numerical Analysis* 34.4, pp. 1386–1425. DOI: 10.1093/imanum/drt024.

Côté, J. and A. Staniforth (1988). "A Two-Time-Level Semi-Lagrangian Semi-implicit Scheme for Spectral Models". In: *Monthly Weather Review* 116.10, pp. 2003–2012. DOI: 10.1175/1520-0493(1988)116<2003:ATTLSL>2.0.CO;2.

Cotter, C. J. and A. T. T. McRae (2014). *Compatible finite element methods for numerical weather prediction*. arXiv: 1401.0616 [math.NA].

Cotter, C. J. and J. Shipton (2012). "Mixed finite elements for numerical weather prediction". In: *Journal of Computational Physics* 231.21, pp. 7076–7091. DOI: 10.1016/j.jcp.2012.05.020.

Cotter, C. J. and J. Thuburn (2014). "A finite element exterior calculus framework for the rotating shallow-water equations". In: *Journal of Computational Physics* 257, pp. 1506–1526. DOI: 10.1016/j.jcp.2013.10.008.

Courant, R. (1943). "Variational methods for the solution of problems of equilibrium and vibrations". In: *Bulletin of the American Mathematical Society* 49.1, pp. 1–24. DOI: 10.1090/s0002-9904-1943-07818-4.

Courant, R., K. Friedrichs, and H. Lewy (1928). "Über die partiellen Differenzengleiehungen der mathematischen Physik". In: *Mathematische Annalen* 100.1, pp. 32–74. DOI: 10.1007/bf01448839.

Crank, J. and P. Nicolson (1947). "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 43.1, pp. 50–67. DOI: 10.1017/s0305004100023197.

Cullen, M. J. P. (2001). "Alternative implementations of the semi-Lagrangian semi-implicit schemes in the ECMWF model". In: *Quarterly Journal of the Royal Meteorological Society* 127.578, pp. 2787–2802. DOI: 10.1256/smsqj.57813.

Dalcin, L. D., R. R. Paz, P. A. Kler, and A. Cosimo (2011). "Parallel distributed computing using Python". In: *Advances in Water Resources* 34.9, pp. 1124–1139. DOI: 10.1016/j.advwatres.2011.04.013.

Danilov, S. (2010). "On utility of triangular C-grid type discretization for numerical modeling of large-scale ocean flows". In: *Ocean Dynamics* 60.6, pp. 1361–1369. DOI: 10.1007/s10236-010-0339-6.

Davies, T., A. Staniforth, N. Wood, and J. Thuburn (2003). "Validity of anelastic and other equation sets as inferred from normal-mode analysis". In: *Quarterly Journal of the Royal Meteorological Society* 129.593, pp. 2761–2775. DOI: 10.1256/qj.02.195.

Davis, P. J. and P. Rabinowitz (1984). *Methods of Numerical Integration*. 2nd. Academic Press. ISBN: 978-0-12-206360-2. DOI: 10.1016/C2013-0-10566-1.

Davis, T. A. (2004). "Algorithm 832: UMFPACK V4.3–an unsymmetric-pattern multifrontal method". In: *ACM Transactions on Mathematical Software* 30.2, pp. 196–199. DOI: 10.1145/992200.992206.

De Veubeke, B. M. F. (1965). "Displacement and Equilibrium Models in the Finite Element Method". In: *B.M. Fraeijs De Veubeke Memorial Volume of Selected Papers*. Springer Netherlands, pp. 101–168. DOI: 10.1007/978-94-009-9147-7_3.

Dedner, A., E. H. Müller, and R. Scheichl (2015). "Efficient multigrid preconditioners for atmospheric flow simulations at high aspect ratio". In: *International Journal for Numerical Methods in Fluids* 80.1, pp. 76–102. DOI: 10.1002/fld.4072.

Dennis, J. M., J. Edwards, K. J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, A. St-Cyr, M. A. Taylor, and P. H. Worley (2011). "CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model". In: *The International Journal of High Performance Computing Applications* 26.1, pp. 74–89. DOI: 10.1177/1094342011428142.

Devloo, P. R. B., C. O. Faria, A. M. Farias, S. M. Gomes, A. F. D. Loula, and S. M. C. Malta (2018). "On continuous, discontinuous, mixed, and primal hybrid finite element methods for second-order elliptic problems". In: *International Journal for Numerical Methods in Engineering* 115.9, pp. 1083–1107. DOI: 10.1002/nme.5836.

Dobrev, V., T. Kolev, C. S. Lee, V. Tomov, and P. S. Vassilevski (2019). "Algebraic Hybridization and Static Condensation with Application to Scalable H(div) Preconditioning". In: *SIAM Journal on Scientific Computing* 41.3, B425–B447. DOI: 10.1137/17m1132562.

Donea, J. (1984). "A Taylor-Galerkin method for convective transport problems". In: *International Journal for Numerical Methods in Engineering* 20.1, pp. 101–119. DOI: 10.1002/nme.1620200108.

Durran, D. R. (1989). "Improving the Anelastic Approximation". In: *Journal of the Atmospheric Sciences* 46.11, pp. 1453–1461. DOI: 10.1175/1520-0469(1989)046<1453:ITAA>2.0.CO;2.

– (2008). "A physically motivated approach for filtering acoustic waves from the equations governing compressible stratified flow". In: *Journal of Fluid Mechanics* 601, pp. 365–379. DOI: 10.1017/s0022112008000608.

Eisenstat, S. C., H. C. Elman, and M. H. Schultz (1983). "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations". In: *SIAM Journal on Numerical Analysis* 20.2, pp. 345–357. DOI: 10.1137/0720023.

Elman, H. C., O. G. Ernst, and D. P. O'Leary (2001). "A Multigrid Method Enhanced by Krylov Subspace Iteration for Discrete Helmholtz Equations". In: *SIAM Journal on Scientific Computing* 23.4, pp. 1291–1315. DOI: 10.1137/s1064827501357190.

Fabien, M. S., M. G. Knepley, R. T. Mills, and B. M. Rivière (2019). "Manycore Parallel Computing for a Hybridizable Discontinuous Galerkin Nested Multigrid Method". In: *SIAM Journal on Scientific Computing* 41.2, pp. C73–C96. DOI: 10.1137/17m1128903.

Falgout, R. D. (2006). "An introduction to algebraic multigrid". In: *Computing in Science & Engineering* 8.6, pp. 24–33. DOI: 10.1109/mcse.2006.105.

Falgout, R. D., J. E. Jones, and U. M. Yang (2006). "The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners". In: *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, pp. 267–294. DOI: 10.1007/3-540-31619-1_8.

Fournier, A., M. A. Taylor, and J. J. Tribbia (2004). "The Spectral Element Atmosphere Model (SEAM): High-Resolution Parallel Computation and Localized Resolution of Regional Dynamics". In: *Monthly Weather Review* 132.3, pp. 726–748. DOI: 10.1175/1520-0493(2004)132<0726:TSEAMS>2.0.CO;2.

Fulton, S. R., P. E. Ciesielski, and W. H. Schubert (1986). "Multigrid Methods for Elliptic Problems: A Review". In: *Monthly Weather Review* 114.5, pp. 943–959. DOI: 10.1175/1520-0493(1986)114<0943:MMFEPA>2.0.CO;2.

Gassmann, A. (2011). "Inspection of hexagonal and triangular C-grid discretizations of the shallow water equations". In: *Journal of Computational Physics* 230.7, pp. 2706–2721. DOI: 10.1016/j.jcp.2011.01.014.

Gee, M. W., C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala (2006). *ML 5.0 smoothed aggregation user's guide*. Tech. rep. Technical Report SAND2006-2649, Sandia National Laboratories.

Geuzaine, C. and J. F. Remacle (2009). "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering* 79.11, pp. 1309–1331. DOI: 10.1002/nme.2579.

Gibson, T. H., A. T.T. McRae, C. J. Cotter, L. Mitchell, and D. A. Ham (2019a). *Compatible Finite Element Methods for Geophysical Flows*. Springer International Publishing. ISBN: 978-3-030-23957-2. DOI: 10.1007/978-3-030-23957-2.

Gibson, T. H., L. Mitchell, D. A. Ham, and C. J. Cotter (2019b). *Slate: extending Firedrake's domain-specific abstraction to hybridized solvers for geoscience and beyond*. arXiv: 1802.00303 [cs.MS].

Giraldo, F. X., J. F. Kelly, and E. M. Constantinescu (2013). "Implicit-Explicit Formulations of a Three-Dimensional Nonhydrostatic Unified Model of the Atmosphere (NUMA)". In: *SIAM Journal on Scientific Computing* 35.5, B1162–B1194. DOI: 10.1137/120876034.

Giraldo, F. X. and M. Restelli (2008). "A study of spectral element and discontinuous Galerkin methods for the Navier–Stokes equations in nonhydrostatic mesoscale atmospheric modeling: Equation sets and test cases". In: *Journal of Computational Physics* 227.8, pp. 3849–3877. DOI: 10.1016/j.jcp.2007.12.009.

Gopalakrishnan, J. (2003). "A Schwarz Preconditioner for a Hybridized Mixed Method". In: *Computational Methods in Applied Mathematics* 3.1. DOI: 10.2478/cmam-2003-0009.

Gopalakrishnan, J. and S. Tan (2009). "A convergent multigrid cycle for the hybridized mixed method". In: *Numerical Linear Algebra with Applications* 16.9, pp. 689–714. DOI: 10.1002/nla.636.

Guennebaud, G., B. Jacob, et al. (2010). *Eigen v3*. URL: http://eigen.tuxfamily.org.

Guyan, R. J. (1965). "Reduction of stiffness and mass matrices". In: *AIAA Journal* 3.2, pp. 380–380. DOI: 10.2514/3.2874.

Haidar, A., T. Dong, P. Luszczek, S. Tomov, and J. Dongarra (2015). "Batched matrix computations on hardware accelerators based on GPUs". In: *The International Journal of High Performance Computing Applications* 29.2, pp. 193–208. DOI: 10.1177/1094342014567546.

Hairer, E. and G. Wanner (1996). *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-05221-7.

Hecht, F. (2012). "New development in freefem++". In: *Journal of Numerical Mathematics* 20.3-4. DOI: 10.1515/jnum-2012-0013.

Hiptmair, R. and J. Xu (2007). "Nodal Auxiliary Space Preconditioning in $H(\mathrm{curl})$ and $H(\mathrm{div})$ Spaces". In: *SIAM Journal on Numerical Analysis* 45.6, pp. 2483–2509. DOI: 10.1137/060660588.

Holdaway, D., J. Thuburn, and N. Wood (2012a). "Comparison of Lorenz and Charney-Phillips vertical discretisations for dynamics-boundary layer coupling. Part I: Steady states". In: *Quarterly Journal of the Royal Meteorological Society* 139.673, pp. 1073–1086. DOI: 10.1002/qj.2016.

– (2012b). "Comparison of Lorenz and Charney-Phillips vertical discretisations for dynamics-boundary layer coupling. Part II: Transients". In: *Quarterly Journal of the Royal Meteorological Society* 139.673, pp. 1087–1098. DOI: 10.1002/qj.2017.

Holst, M. and A. Stern (2012). "Geometric Variational Crimes: Hilbert Complexes, Finite Element Exterior Calculus, and Problems on Hypersurfaces". In: *Foundations of Computational Mathematics* 12.3, pp. 263–293. DOI: 10.1007/s10208-012-9119-7.

Homolya, M., R. C. Kirby, and D. A. Ham (2017). *Exposing and exploiting structure: optimal code generation for high-order finite element methods*. arXiv: 1711.02473 [cs.MS].

Homolya, M., L. Mitchell, F. Luporini, and D. A. Ham (2018). "TSFC: A Structure-Preserving Form Compiler". In: *SIAM Journal on Scientific Computing* 40.3, pp. C401–C428. DOI: 10.1137/17m1130642.

Horn, R. A. and C. R. Johnson (2012). *Matrix Analysis*. 2nd. Cambridge University Press. ISBN: 9781139020411. DOI: 10.1017/9781139020411.

Hrennikoff, A. (1941). "Solution of problems of elasticity by the framework method". In: *Journal of Applied Mechanics* 8.4, pp. 169–175.

Irons, B. (1965). "Structural eigenvalue problems - elimination of unwanted variables". In: *AIAA Journal* 3.5, pp. 961–962. DOI: 10.2514/3.3027.

Jablonowski, C. and D. L. Williamson (2006). "A baroclinic instability test case for atmospheric model dynamical cores". In: *Quarterly Journal of the Royal Meteorological Society* 132.621C, pp. 2943–2975. DOI: 10.1256/qj.06.12.

Kalchev, D. Z., C. S. Lee, U. Villa, Y. Efendiev, and P. S. Vassilevski (2016). "Upscaling of Mixed Finite Element Discretization Problems by the Spectral AMGe Method". In: *SIAM Journal on Scientific Computing* 38.5, A2912–A2933. DOI: 10.1137/15m1036683.

Kalnay, E. (2002). "Atmospheric predictability and ensemble forecasting". In: *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, pp. 205–260. DOI: 10.1017/cbo9780511802270.007.

Kang, S., F. X. Giraldo, and T. Bui-Thanh (2019). "IMEX HDG-DG: A coupled implicit hybridized discontinuous Galerkin and explicit discontinuous Galerkin approach for shallow water systems". In: *Journal of Computational Physics*, p. 109010. DOI: 10.1016/j.jcp.2019.109010.

Kelly, J. F. and F. X. Giraldo (2012). "Continuous and discontinuous Galerkin methods for a scalable three-dimensional nonhydrostatic atmospheric model: Limited-area mode". In: *Journal of Computational Physics* 231.24, pp. 7988–8008. DOI: 10.1016/j.jcp.2012.04.042.

Kirby, R. C. (2004). "Algorithm 839: FIAT, a new paradigm for computing finite element basis functions". In: *ACM Transactions on Mathematical Software* 30.4, pp. 502–516. DOI: 10.1145/1039813.1039820.

– (2018). "A general approach to transforming finite elements". In: *SMAI Journal of Computational Mathematics* 4, pp. 197–224. DOI: 10.5802/smai-jcm.33.

Kirby, R. C. and A. Logg (2006). "A compiler for variational forms". In: *ACM Transactions on Mathematical Software* 32.3, pp. 417–444. DOI: 10.1145/1163641.1163644.

Kirby, R. C. and L. Mitchell (2018a). *Code generation for generally mapped finite elements*. arXiv: 1808.05513 [cs.MS].

– (2018b). "Solver Composition Across the PDE/Linear Algebra Barrier". In: *SIAM Journal on Scientific Computing* 40.1, pp. C76–C98. DOI: 10.1137/17m1133208.

Kirby, R. C., A. Logg, M. E. Rognes, and A. R. Terrel (2012). "Common and unusual finite elements". In: *Automated Solution of Differential Equations by the Finite Element Method*. Springer International Publishing, pp. 95–119. DOI: 10.1007/978-3-642-23099-8_3.

Kirby, R. M., S. J. Sherwin, and B. Cockburn (2011). "To CG or to HDG: A Comparative Study". In: *Journal of Scientific Computing* 51.1, pp. 183–212. DOI: 10.1007/s10915-011-9501-7.

Klemp, J. B., J. Dudhia, and A. D. Hassiotis (2008). "An Upper Gravity-Wave Absorbing Layer for NWP Applications". In: *Monthly Weather Review* 136.10, pp. 3987–4004. DOI: 10.1175/2008mwr2596.1.

Knepley, M. G. and D. A. Karpeev (2009). "Mesh Algorithms for PDE with Sieve I: Mesh Distribution". In: *Scientific Programming* 17.3, pp. 215–230. DOI: 10.1155/2009/948613.

Kolev, T. V. and P. S. Vassilevski (2009). "Parallel Auxiliary Space AMG for H(Curl) Problems". In: *Journal of Computational Mathematics* 27.5, pp. 604–623. DOI: 10.4208/jcm.2009.27.5.013.

Kronbichler, M. and W. A. Wall (2018). "A Performance Comparison of Continuous and Discontinuous Galerkin Methods with Fast Multigrid Solvers". In: *SIAM Journal on Scientific Computing* 40.5, A3423–A3448. DOI: 10.1137/16m110455x.

Lange, M., L. Mitchell, M. G. Knepley, and G. J. Gorman (2016). "Efficient Mesh Management in Firedrake Using PETSc DMPlex". In: *SIAM Journal on Scientific Computing* 38.5, S143–S155. DOI: 10.1137/15m1026092.

Lax, P. D. and A. N. Milgram (1955). "Parabolic Equations". In: *Contributions to the Theory of Partial Differential Equations*. Princeton University Press, pp. 167–190. DOI: 10.1515/9781400882182-010.

Li, X., C. Chen, X. Shen, and F. Xiao (2013). "A Multimoment Constrained Finite-Volume Model for Nonhydrostatic Atmospheric Dynamics". In: *Monthly Weather Review* 141.4, pp. 1216–1240. DOI: 10.1175/mwr-d-12-00144.1.

Lin, P. T., J. N. Shadid, and P. H. Tsuji (2019). *On the performance of Krylov smoothing for fully coupled AMG preconditioners for VMS resistive MHD*. Tech. rep. 12, pp. 1297–1309. DOI: 10.1002/nme.6178.

Logg, A., K.-A. Mardal, and G. N. Wells (2012). *Automated Solution of Differential Equations by the Finite Element Method*. Springer International Publishing. ISBN: 978-3-642-23098-1. DOI: 10.1007/978-3-642-23099-8.

Logg, A. and G. N. Wells (2010). "DOLFIN: Automated finite element computing". In: *ACM Transactions on Mathematical Software* 37.2, pp. 1–28. DOI: 10.1145/1731022.1731030.

Logg, A., K. B. Ølgaard, M. E. Rognes, and G. N. Wells (2012). "FFC: the FEniCS form compiler". In: *Automated Solution of Differential Equations by the Finite Element Method*. Springer Berlin Heidelberg, pp. 227–238. DOI: 10.1007/978-3-642-23099-8_11.

Long, K., R. C. Kirby, and B. G. van Bloemen Waanders (2010). "Unified Embedded Parallel Finite Element Computations via Software-Based Fréchet Differentiation". In: *SIAM Journal on Scientific Computing* 32.6, pp. 3323–3351. DOI: 10.1137/09076920x.

Lorenz, E. N. (1960). "Energy and Numerical Weather Prediction". In: *Tellus* 12.4, pp. 364–373. DOI: 10.1111/j.2153-3490.1960.tb01323.x.

Luporini, F., A. L. Varbanescu, F. Rathgeber, G.-T. Bercea, J. Ramanujam, D. A. Ham, and P. H. J. Kelly (2015). "Cross-Loop Optimization of Arithmetic Intensity for Finite Element Local Assembly". In: *ACM Transactions on Architecture and Code Optimization* 11.4, pp. 1–25. DOI: 10.1145/2687415.

Lynch, P. (2008). "The ENIAC Forecasts: A Re-creation". In: *Bulletin of the American Meteorological Society* 89.1, pp. 45–56. DOI: 10.1175/bams-89-1-45.

Mandel, J. (1986). "Multigrid convergence for nonsymmetric, indefinite variational problems and one smoothing step". In: *Applied Mathematics and Computation* 19.1-4, pp. 201–216. DOI: 10.1016/0096-3003(86)90104-9.

Marini, L. D. (1985). "An Inexpensive Method for the Evaluation of the Solution of the Lowest Order Raviart–Thomas Mixed Method". In: *SIAM Journal on Numerical Analysis* 22.3, pp. 493–496. DOI: 10.1137/0722029.

Markall, G. R., A. Slemmer, D. A. Ham, P. H. J. Kelly, C. D. Cantwell, and S. J. Sherwin (2012). "Finite element assembly strategies on multi-core and many-core architectures". In: *International Journal for Numerical Methods in Fluids* 71.1, pp. 80–97. DOI: 10.1002/fld.3648.

Marras, S., J. F. Kelly, M. Moragues, A. Müller, M. A. Kopera, M. Vázquez, F. X. Giraldo, G. Houzeaux, and O. Jorba (2015). "A Review of Element-Based Galerkin Methods for Numerical Weather Prediction: Finite Elements, Spectral Elements, and Discontinuous Galerkin". In: *Archives of Computational Methods in Engineering* 23.4, pp. 673–722. DOI: 10.1007/s11831-015-9152-1.

McRae, A. T. T. and C. J. Cotter (2014). "Energy- and enstrophy-conserving schemes for the shallow-water equations, based on mimetic finite elements". In: *Quarterly Journal of the Royal Meteorological Society* 140.684, pp. 2223–2234. DOI: 10.1002/qj.2291.

McRae, A. T. T., G.-T. Bercea, L. Mitchell, D. A. Ham, and C. J. Cotter (2016). "Automated Generation and Symbolic Manipulation of Tensor Product Finite Elements". In: *SIAM Journal on Scientific Computing* 38.5, S25–S47. DOI: 10.1137/15m1021167.

Melvin, T., M. Dubal, N. Wood, A. Staniforth, and M. Zerroukat (2010). "An inherently mass-conserving iterative semi-implicit semi-Lagrangian discretization of the non-hydrostatic vertical-slice equations". In: *Quarterly Journal of the Royal Meteorological Society* 136.648, pp. 799–814. DOI: 10.1002/qj.603.

Melvin, T., T. Benacchio, J. Thuburn, and C. J. Cotter (2018). "Choice of function spaces for thermodynamic variables in mixed finite-element methods". In: *Quarterly Journal of the Royal Meteorological Society* 144.712, pp. 900–916. DOI: 10.1002/qj.3268.

Melvin, T., T. Benacchio, B. J. Shipway, N. Wood, J. Thuburn, and C. J. Cotter (2019). "A mixed finite-element, finite-volume, semi-implicit discretization for atmospheric dynamics: Cartesian geometry". In: *Quarterly Journal of the Royal Meteorological Society* 145.724, pp. 2835–2853. DOI: 10.1002/qj.3501.

Met Office, UK (2019). *Next generation atmospheric model development*. Accessed: 2019-09-13. URL: https://www.metoffice.gov.uk/research/news/2019/gungho-and-lfric.

Mitchell, L. and E. H. Müller (2016). "High level implementation of geometric multigrid solvers for finite element problems: Applications in atmospheric modelling". In: *Journal of Computational Physics* 327, pp. 1–18. DOI: 10.1016/j.jcp.2016.09.037.

Morley, L. S. D. (1971). "The constant-moment plate-bending element". In: *Journal of Strain Analysis* 6.1, pp. 20–24. DOI: 10.1243/03093247v061020.

Müller, E. H. and R. Scheichl (2014). "Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction". In: *Quarterly Journal of the Royal Meteorological Society* 140.685, pp. 2608–2624. DOI: 10.1002/qj.2327.

Nair, R. D., S. J. Thomas, and R. D. Loft (2005). "A Discontinuous Galerkin Global Shallow Water Model". In: *Monthly Weather Review* 133.4, pp. 876–888. DOI: 10.1175/mwr2903.1.

Natale, A. and C. J. Cotter (2017). "A variational $H(\mathrm{div})$ finite-element discretization approach for perfect incompressible fluids". In: *IMA Journal of Numerical Analysis* 38.3, pp. 1388–1419. DOI: 10.1093/imanum/drx033.

Natale, A., J. Shipton, and C. J. Cotter (2016). "Compatible finite element spaces for geophysical fluid dynamics". In: *Dynamics and Statistics of the Climate System* 1.1. DOI: 10.1093/climsys/dzw005.

Nechaev, D. and M. Yaremchuk (2004). "On the Approximation of the Coriolis Terms in C-Grid Models". In: *Monthly Weather Review* 132.9, pp. 2283–2289. DOI: 10.1175/1520-0493(2004)132<2283:OTAOTC>2.0.CO;2.

Nédélec, J.-C. (1980). "Mixed finite elements in $\mathbb{R}^3$". In: *Numerische Mathematik* 35.3, pp. 315–341. DOI: 10.1007/bf01396415.

– (1986). "A new family of mixed finite elements in $\mathbb{R}^3$". In: *Numerische Mathematik* 50.1, pp. 57–81. DOI: 10.1007/bf01389668.

Ogura, Y. and N. A. Phillips (1962). "Scale Analysis of Deep and Shallow Convection in the Atmosphere". In: *Journal of the Atmospheric Sciences* 19.2, pp. 173–179. DOI: 10.1175/1520-0469(1962)019<0173:SAODAS>2.0.CO;2.

Ølgaard, K. B. and G. N. Wells (2010). "Optimizations for quadrature representations of finite element tensors through automated code generation". In: *ACM Transactions on Mathematical Software* 37.1, pp. 1–23. DOI: 10.1145/1644001.1644009.

Penrose, R. (1955). "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 51.3, pp. 406–413. DOI: 10.1017/s0305004100030401.

Poirier, D., S. Allmaras, D. McCarthy, M. Smith, and F. Enomoto (1998). "The CGNS system". In: *29th AIAA, Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.1998-3007.

Prud'homme, C., V. Chabannes, V. Doyeux, M. Ismail, A. Samake, and G. Pena (2012). "Feel++: A computational framework for Galerkin Methods and Advanced Numerical Methods". In: *ESAIM: Proceedings* 38, pp. 429–455. DOI: 10.1051/proc/201238024.

Putman, W. M. and S. J. Lin (2007). "Finite-volume transport on various cubed-sphere grids". In: *Journal of Computational Physics* 227.1, pp. 55–78. DOI: 10.1016/j.jcp.2007.07.022.

Rathgeber, F., G. R. Markall, L. Mitchell, N. Loriant, D. A. Ham, C. Bertolli, and P. H.J. Kelly (2012). "PyOP2: A High-Level Framework for Performance-Portable Simulations on Unstructured Meshes". In: *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE. DOI: 10.1109/sc.companion.2012.134.

Rathgeber, F., D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. Mcrae, G.-T. Bercea, G. R. Markall, and P. H. J. Kelly (2017). "Firedrake: automating the finite element method by composing abstractions". In: *ACM Transactions on Mathematical Software* 43.3, pp. 1–27. DOI: 10.1145/2998441.

Raviart, P. A. and J. M. Thomas (1977). "A mixed finite element method for 2-nd order elliptic problems". In: *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, pp. 292–315. DOI: 10.1007/bfb0064470.

Richardson, L. F. (1911). "The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 210.459-470, pp. 307–357. DOI: 10.1098/rsta.1911.0009.

Ringler, T. D., J. Thuburn, J. B. Klemp, and W. C. Skamarock (2010). "A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured C-grids". In: *Journal of Computational Physics* 229.9, pp. 3065–3090. DOI: 10.1016/j.jcp.2009.12.007.

Rognes, M. E., R. C. Kirby, and A. Logg (2009). "Efficient Assembly of H(div) and H(curl) Conforming Finite Elements". In: *SIAM Journal on Scientific Computing* 31.6, pp. 4130–4151. DOI: 10.1137/08073901x.

Rognes, M. E., D. A. Ham, C. J. Cotter, and A. T. T. McRae (2013). "Automating the solution of PDEs on the sphere and other manifolds in FEniCS 1.2". In: *Geoscientific Model Development* 6.6, pp. 2099–2119. DOI: 10.5194/gmd-6-2099-2013.

Rothe, E. (1930). "Zweidimensionale parabolische Randwertaufgaben als Grenzfall eindimensionaler Randwertaufgaben". In: *Mathematische Annalen* 102.1, pp. 650–670. DOI: 10.1007/bf01782368.

Ruge, J. W. and K. Stüben (1987). "Algebraic Multigrid". In: *Multigrid Methods*. Society for Industrial and Applied Mathematics, pp. 73–130. DOI: 10.1137/1.9781611971057.ch4.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718003.

Schneider, E. K. (1987). "An Inconsistency in Vertical Discretization in Some Atmospheric Models". In: *Monthly Weather Review* 115.9, pp. 2166–2169. DOI: 10.1175/1520-0493(1987)115<2166:AIIVDI>2.0.CO;2.

Schoof, L. A. and V. R. Yarberry (1994). *EXODUS II: A finite element data model*. Tech. rep. DOI: 10.2172/10102115.

Shapira, Y. (2008). *Matrix-Based Multigrid*. Springer US. DOI: 10.1007/978-0-387-49765-5.

Shewchuk, J. R. (1996). "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator". In: *Applied Computational Geometry Towards Geometric Engineering*. Springer Berlin Heidelberg, pp. 203–222. DOI: 10.1007/bfb0014497.

Shipton, J., T. H. Gibson, and C. J. Cotter (2018). "Higher-order compatible finite element schemes for the nonlinear rotating shallow water equations on the sphere". In: *Journal of Computational Physics* 375, pp. 1121–1137. DOI: 10.1016/j.jcp.2018.08.027.

Shu, C. W. and S. Osher (1988). "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2, pp. 439–471. DOI: 10.1016/0021-9991(88)90177-5.

Siefert, C. and E. De Sturler (2006). "Preconditioners for Generalized Saddle-Point Problems". In: *SIAM Journal on Numerical Analysis* 44.3, pp. 1275–1296. DOI: 10.1137/040610908.

Skamarock, W. C. and J. B. Klemp (1994). "Efficiency and Accuracy of the Klemp-Wilhelmson Time-Splitting Technique". In: *Monthly Weather Review* 122.11, pp. 2623–2630. DOI: 10.1175/1520-0493(1994)122<2623:EAAOTK>2.0.CO;2.

Smolarkiewicz, P. K. and L. G. Margolin (1994). *Variational elliptic solver for atmospheric applications*. Tech. rep. DOI: 10.2172/10130964.

Staniforth, A. and J. Côté (1991). "Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review". In: *Monthly Weather Review* 119.9, pp. 2206–2223. DOI: 10.1175/1520-0493(1991)119<2206:SLISFA>2.0.CO;2.

Staniforth, A., T. Melvin, and C. J. Cotter (2013). "Analysis of a mixed finite-element pair proposed for an atmospheric dynamical core". In: *Quarterly Journal of the Royal Meteorological Society* 139.674, pp. 1239–1254. DOI: 10.1002/qj.2028.

Staniforth, A. and J. Thuburn (2011). "Horizontal grids for global weather and climate prediction models: a review". In: *Quarterly Journal of the Royal Meteorological Society* 138.662, pp. 1–26. DOI: 10.1002/qj.958.

Staniforth, A. and N. Wood (2008). "Aspects of the dynamical core of a nonhydrostatic, deep-atmosphere, unified weather and climate-prediction model". In: *Journal of Computational Physics* 227.7, pp. 3445–3464. DOI: 10.1016/j.jcp.2006.11.009.

Stenberg, R. (1991). "Postprocessing schemes for some mixed finite elements". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 25.1, pp. 151–167. DOI: 10.1051/m2an/1991250101511.

Stevenson, R. (1994). "Robust Multi-grid with 7-point ILU Smoothing". In: *Multigrid Methods IV*. Birkhäuser Basel, pp. 295–307. DOI: 10.1007/978-3-0348-8524-9_22.

Straka, J. M., R. B. Wilhelmson, L. J. Wicker, J. R. Anderson, and K. K. Droegemeier (1993). "Numerical solutions of a non-linear density current: A benchmark solution and comparisons". In: *International Journal for Numerical Methods in Fluids* 17.1, pp. 1–22. DOI: 10.1002/fld.1650170103.

Strang, G. (1973). "Piecewise polynomials and the finite element method". In: *Bulletin of the American Mathematical Society* 79.6, pp. 1128–1138. DOI: 10.1090/s0002-9904-1973-13351-8.

Stüben, K. (2001). "A review of algebraic multigrid". In: *Numerical Analysis: Historical Developments in the 20th Century*. Elsevier, pp. 331–359. DOI: 10.1016/b978-0-444-50617-7.50015-x.

Süli, E. and D. F. Mayers (2003). *An Introduction to Numerical Analysis*. Cambridge University Press. ISBN: 9780511801181. DOI: 10.1017/CBO9780511801181.

Taylor, C. and P. Hood (1973). "A numerical solution of the Navier-Stokes equations using the finite element technique". In: *Computers & Fluids* 1.1, pp. 73–100. DOI: 10.1016/0045-7930(73)90027-3.

Temperton, C. (1997). "Treatment of the Coriolis Terms in Semi-Lagrangian Spectral Models". In: *Atmosphere-Ocean* 35.sup1, pp. 293–302. DOI: 10.1080/07055900.1997.9687353.

Thomas, J. M. (1976). "Méthode des éléments finis hybrides duaux pour les problèmes elliptiques du second ordre". In: *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique* 10.R3, pp. 51–79. DOI: 10.1051/m2an/197610r300511.

Thomas, S. J. and R. D. Loft (2005). "The NCAR spectral element climate dynamical core: Semi-implicit eulerian formulation". In: *Journal of Scientific Computing* 25.1-2, pp. 307–322. DOI: 10.1007/bf02728993.

Thomas, S. J., J. P. Hacker, P. K. Smolarkiewicz, and R. B. Stull (2003). "Spectral Preconditioners for Nonhydrostatic Atmospheric Models". In: *Monthly Weather Review* 131.10, pp. 2464–2478. DOI: 10.1175/1520-0493(2003)131<2464:SPFNAM>2.0.CO;2.

Thuburn, J. (2016). "ENDGame: The New Dynamical Core of the Met Office Weather and Climate Prediction Model". In: *UK Success Stories in Industrial Mathematics*. Springer International Publishing, pp. 27–33. DOI: 10.1007/978-3-319-25454-8_4.

Thuburn, J. and C. J. Cotter (2012). "A Framework for Mimetic Discretization of the Rotating Shallow-Water Equations on Arbitrary Polygonal Grids". In: *SIAM Journal on Scientific Computing* 34.3, B203–B225. DOI: 10.1137/110850293.

Thuburn, J., C. J. Cotter, and T. Dubos (2014). "A mimetic, semi-implicit, forward-in-time, finite volume shallow water model: comparison of hexagonal–icosahedral and cubed-sphere grids". In: *Geoscientific Model Development* 7.3, pp. 909–929. DOI: 10.5194/gmd-7-909-2014.

Ullrich, P. A., C. Jablonowski, and B. Van Leer (2010). "High-order finite-volume methods for the shallow-water equations on the sphere". In: *Journal of Computational Physics* 229.17, pp. 6104–6134. DOI: 10.1016/j.jcp.2010.04.044.

Ullrich, P. A., K. A. Reed, and C. Jablonowski (2015). "Analytical initial conditions and an analysis of baroclinic instability waves in f- and β-plane 3D channel models". In: *Quarterly Journal of the Royal Meteorological Society* 141.693, pp. 2972–2988. DOI: 10.1002/qj.2583.

Ullrich, P. A., C. Jablonowski, J. Kent, P. H. Lauritzen, R. D. Nair, and M. A. Taylor (2012). "Dynamical core model intercomparison project (DCMIP) test case document". In: *DCMIP Summer School* 83.

Ullrich, P. A., T. Melvin, C. Jablonowski, and A. Staniforth (2013). "A proposed baroclinic wave test case for deep- and shallow-atmosphere dynamical cores". In: *Quarterly Journal of the Royal Meteorological Society* 140.682, pp. 1590–1602. DOI: 10.1002/qj.2241. URL: https://doi.org/10.1002%2Fqj.2241.

Vallis, G. K. (2017). *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-Scale Circulation*. 2nd. Cambridge University Press. ISBN: 9780511790447. DOI: 10.1017/CBO9780511790447.

Vaněk, P., M. Brezina, and J. Mandel (2001). "Convergence of algebraic multigrid based on smoothed aggregation". In: *Numerische Mathematik* 88.3, pp. 559–579. DOI: 10.1007/s211-001-8015-y.

Vaněk, P., J. Mandel, and M. Brezina (1996). "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems". In: *Computing* 56.3, pp. 179–196. DOI: 10.1007/bf02238511.

Wesseling, P. and C. W. Oosterlee (2001). "Geometric multigrid with applications to computational fluid dynamics". In: *Partial Differential Equations*. Elsevier, pp. 311–334. DOI: 10.1016/b978-0-444-50616-0.50013-0.

Williamson, D. L. (2007). "The Evolution of Dynamical Cores for Global Atmospheric Models". In: *Journal of the Meteorological Society of Japan* 85B, pp. 241–269. DOI: 10.2151/jmsj.85b.241.

Williamson, D. L., J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber (1992). "A standard test set for numerical approximations to the shallow water equations in spherical geometry". In: *Journal of Computational Physics* 101.1, pp. 227–228. DOI: 10.1016/0021-9991(92)90060-c.

Wimmer, G. A., C. J. Cotter, and W. Bauer (2019). "Energy conserving upwinded compatible finite element schemes for the rotating shallow water equations". In: *Journal of Computational Physics*, p. 109016. DOI: 10.1016/j.jcp.2019.109016.

Wittum, G. (1989). "On the Robustness of ILU Smoothing". In: *SIAM Journal on Scientific and Statistical Computing* 10.4, pp. 699–717. DOI: 10.1137/0910043.

Wood, N., A. Staniforth, A. White, T. Allen, M. Diamantakis, M. Gross, T. Melvin, C. Smith, S. Vosper, M. Zerroukat, and J. Thuburn (2014). "An inherently mass-conserving semi-implicit semi-Lagrangian discretization of the deep-atmosphere global non-hydrostatic equations". In: *Quarterly Journal of the Royal Meteorological Society* 140.682, pp. 1505–1520. DOI: 10.1002/qj.2235.

Yakovlev, S., D. Moxey, R. M. Kirby, and S. J. Sherwin (2015). "To CG or to HDG: A Comparative Study in 3D". In: *Journal of Scientific Computing* 67.1, pp. 192–220. DOI: 10.1007/s10915-015-0076-6.

Zängl, G., D. Reinert, P. Rípodas, and M. Baldauf (2014). "The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core". In: *Quarterly Journal of the Royal Meteorological Society* 141.687, pp. 563–579. DOI: 10.1002/qj.2378.

Zenodo/Tabula-Rasa (2019). *Tabula Rasa: experimentation framework for hybridization and static condensation*. DOI: 10.5281/zenodo.2616031.

Zienkiewicz, O.C. and Y. K. Cheung (1965). "Finite elements in the solution of field problems". In: *The Engineer* 220.5722, pp. 507–510.