High Order Hybrid Discontinuous Galerkin Regional Ocean Modelling

by

Mattheus Percy Ueckermann

BASc., University of Waterloo (2007)
S.M., Massachusetts Institute of Technology (2009)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2014

Author .....
Department of Mechanical Engineering
09/20/2013

Certified by .....
Pierre F. J. Lermusiaux
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
David E. Hardt
Chairman, Department Committee on Graduate Theses

High Order Hybrid Discontinuous Galerkin Regional Ocean Modelling

by

Mattheus Percy Ueckermann

Submitted to the Department of Mechanical Engineering
on 09/20/2013, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Accurate modeling of physical and biogeochemical dynamics in coastal ocean regions is required for multiple scientific and societal applications, covering a wide range of time and space scales. However, in light of the strong nonlinearities observed in coastal regions and in biological processes, such modeling is challenging. An important subject that has been largely overlooked is the numerical requirements for regional ocean simulation studies. Major objectives of this thesis are to address such computational questions for non-hydrostatic multiscale flows and for biogeochemical interactions, and to derive and develop numerical schemes that meet these requirements, utilizing the latest advances in computational fluid dynamics.

We are interested in studying nonlinear, transient, and multiscale ocean dynamics over complex geometries with steep bathymetry and intricate coastlines, from sub-mesoscales to basin-scales. These dynamical interests, when combined with our requirements for accurate, efficient and flexible ocean modeling, led us to develop new variable resolution, higher-order and non-hydrostatic ocean modeling schemes. Specifically, we derived, developed and applied new numerical schemes based on the novel hybrid discontinuous Galerkin (HDG) method in combination with projection methods.

The new numerical schemes are first derived for the Navier-Stokes equations. To ensure mass conservation, we define numerical fluxes that are consistent with the discrete divergence equation. To improve stability and accuracy, we derive a consistent HDG stability parameter for the pressure-correction equation. We also apply a new boundary condition for the pressure-corrector, and show the form and origin of the projection method's time-splitting error for a case with implicit diffusion and explicit advection. Our scheme is implemented for arbitrary, mixed-element unstructured grids using a novel quadrature-free integration method for a nodal basis, which is consistent with the HDG method. To prevent numerical oscillations, we design a selective high-order nodal limiter. We demonstrate the correctness of our new schemes using a tracer advection benchmark, a manufactured solution for the steady diffusion and stokes equations, and the 2D lock-exchange problem.

These numerical schemes are then extended for non-hydrostatic, free-surface, variable-density regional ocean dynamics. The time-splitting procedure using projection methods is derived for non-hydrostatic or hydrostatic, and nonlinear free-surface or rigid-lid, versions of the model. We also derive consistent HDG stability parameters for the free-surface and non-hydrostatic pressure-corrector equations to ensure stability and accuracy. New boundary conditions for the free-surface-corrector and pressure-corrector are also introduced. We prove that these conditions lead to consistent boundary conditions for the free-surface and pressure proper. To ensure discrete mass conservation with a moving free-surface, we use an arbitrary Lagrangian-Eulerian (ALE) moving mesh algorithm. These schemes are again verified, this time using a tidal flow problem with analytical solutions and a 3D lock-exchange benchmark.

We apply our new numerical schemes to evaluate the numerical requirements of the coupled biological-physical dynamics. We find that higher-order schemes are more accurate at the same efficiency compared to lower-order (e.g. second-order) accurate schemes when modeling a biological patch. Due to decreased numerical dissipation, the higher-order schemes are capable of modeling biological patchiness over a sustained duration, while the lower-order schemes can lose significant biomass after a few non-dimensional times and can thus solve erroneous nonlinear dynamics.

Finally, inspired by Stellwagen Bank in Massachusetts Bay, we study the effect of non-hydrostatic physics on biological productivity and phytoplankton fields for tidally-driven flows over an idealized bank. We find that the non-hydrostatic pressure and flows are important for biological dynamics, especially when flows are supercritical. That is, when the slope of the topography is larger than the slope of internal wave rays at the tidal frequency. The non-hydrostatic effects increase with increasing nonlinearity, both when the internal Froude number and criticality parameter increase. Even in cases where the instantaneous biological productivity is not largely modified, we find that the total biomass, spatial variability and patchiness of phytoplankton can be significantly altered by non-hydrostatic processes.

Our ultimate dynamics motivation is to allow quantitative simulation studies of fundamental nonlinear biological-physical dynamics in coastal regions with complex bathymetric features such as straits, sills, ridges and shelfbreaks. This thesis develops the necessary numerical schemes that meet the stringent accuracy requirements for these types of flows and dynamics.

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

Acknowledgments

When I first entered MIT, I did not expect to find a community of people who value humility, others above themselves, and collaboration. Yet, now I find myself completely surrounded by good, dear, friends with just those qualities.

Many thanks to my thesis advisor, Pierre, for his guidance throughout the process of this thesis work. Particularly, I thank him for his willingness and desire to help make everything we work on excellent. Pierre, I will forever remember your drive for perfection, your attention to detail, and that "two brains are better than one."

Also thank you for my committee members, Professors Flierl, Marshall, Peraire, and Yue for all the helpful comments and suggestions.

Pat Haley was kind enough to check some of my equations, create model runs for comparison, teach me how to make the best use of the cluster, and to offer many suggestions leading to significant insights. Particularly, Pat suggested that I compare my discrete split equations to the discrete un-split form, and he reproduced the runs with MSEAS to help debug my own code. Pat, thanks so much for all the helpful discussions over the years. Pat, without your suggestions, many of my ideas would still be lost, floating somewhere in the ocean of undiscovered solutions.

Wayne Leslie provided me with the bathymetric data, CTD data, and chlorophyll data for the Stellwagen bank region. And Wayne, thanks also for web–pages, up-and-running nodes, music, and "humour."

Chris Mirabito helped me debug parts of my code. Thanks Chris for all your help, your patience, and for taking up the torch.

Sam Kelly helped with the internal wave theory and results for the idealized bank simulations. Thanks Sam for looking at my simulations, and comparing them with your own. Also, thank you for the useful references, and helping me with the theory.

Jorge Colmenero helped characterized the Stellwagen Bank bathymetry for the idealized bank simulations. Thanks also for all the meshes!

Ngoc–Cuong Nguyen helped me early in this work to get started with hybrid discontinuous Galerkin methods. Thanks for taking time to look at some of my

theoretical issues, and for explaining how the advanced post-processors work.

For the projection method work, I want to thank Dave Shirokoff, for many late-night discussions. There are certainly advantages to having an applied mathematician as a roommate, not to mention July 4^{th} raft adventures.

Thanks to all my lab–mates over the years, Eric, Lisa, Arpit, Themis, Thomas, Tapovan, Konur, Pete, and Akash. Eric for being a friend, a landlord, for always offering a helping hand, and for the occasional hot chocolate. Lisa and Arpit, for the camaraderie during the first few years of hurdles. Themis, for teaching me so much, for your encouragement, guidance, and the wonderful collaborations – I enjoyed writing papers with you and Pierre. Thomas, for all the good times, your love of mathematical rigour, and the epic badminton matches. Tapovan for all your generosity, in food, conversation, and work. I particularly enjoyed working with you on level–sets. Konur, for introducing me to Bertrand Russel, Hamilton-Jacobi-Bellman equations, and all the theological discussions. Pete for using my code! Akash for your patience, your excellent attitude, and fun in the lab.

I also want to thank all my friends outside the lab. Mike Tupek, for so much, from tennis to numerics to weddings, it's been a blast. Beracah for tennis, food, fun, and being there. Matthew Fontana for excellent conversations, food, and friendship. Emily for tennis and being awesome. Then for keeping me fed and social, I also want to thank Rahul, Mike, Tamer, Sian, Aalap, and Will. My community members at Hope Fellowship have truly been a blessing, and a constant source of love and support, thanks so much Tyson, Kelly, Pete, Deb, Dan, Monica, Jordan, Jen, Sean, Nate, Nicole, Chuck, Katie, Matthew, Dane, and Curtis.

To mom and dad, thanks so much for always supporting me in whatever I do. Thanks for pushing when I need a push, and pulling when I am pushing too hard. Thanks to my sister Anabel who helped out so much with her wise, sagely advice. Thank also Richard, Dorrie, and Abby, my new family, for all your love and encouragement.

Last and certainly not least, I want to thank my wonderful wife, Sarah. Thank you for the love, support, patience and grace you have shown me through this time of

uncertainty. Thank you for being the voice of reason and reality to ground my naive optimism. I look forward to facing the trials of this life with you.

I am grateful to the Massachusetts Institute of Technology for awarding me a Pappalardo Fellowship for my first academic year at MIT. I am also very thankful to several agencies for support under the following research grants to MIT: the Office of Naval Research for research support under grants N00014-08-1-109 (ONR6.1) and N00014-07-1-0473 (PhilEx); the National Science Foundation for the grant OCE-1061160 ("Causes and Effects of Shelf-edge Internal Tide Variability"); and the MIT Sea Grant program and National Oceanic and Atmospheric Administration for the grant: "High Productivity on a Coastal Bank: Physical and Biological Interactions." Finally, I am very grateful to the Natural Sciences and Engineering Research Council of Canada for awarding me a scholarship to aid with my financial support.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Analytical Symbols

∇_{xy} The horizontal gradient operator, $\nabla_{xy} = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, 0\right]$.

∇_z The vertical gradient operator, $\nabla_z = \left[0, 0, \frac{\partial}{\partial z}\right]$.

η The elevation of the free-surface above the mean ocean surface. $\eta = \eta(x, y, t)$.

η_s The surface-pressure at the top of the rigid-lid $\eta_s = \eta_s(x, y, t)$.

κ The turbulent eddy diffusivity. For DNS simulations, this is the molecular diffusivity. In general, $\kappa = \kappa(\mathbf{x}, t)$.

κ_{xy} The turbulent eddy diffusivity in the horizontal directions. For DNS simulations, this is the molecular diffusivity. In general, $\kappa_{xy} = \kappa_{xy}(\mathbf{x}, t)$.

κ_z The turbulent eddy diffusivity in the vertical direction. For DNS simulations, this is the molecular diffusivity. In general, $\kappa_z = \kappa_z(\mathbf{x}, t)$.

μ_{xy} The dynamic turbulent eddy viscosity in the horizontal directions. For DNS simulations, this is the molecular dynamic viscosity of the fluid. In general, $\mu_{xy} = \mu_{xy}(\mathbf{x}, t)$.

μ_z The dynamic turbulent eddy viscosity in the vertical direction. For DNS simulations, this is the molecular dynamic viscosity of the fluid. In general, $\mu_z = \mu_z(\mathbf{x}, t)$.

ν The kinematic turbulent eddy viscosity. For DNS simulations, this is the molecular kinematic viscosity of the fluid. In general, $\nu = \nu(\mathbf{x}, t)$.

ν_{xy} The kinematic turbulent eddy viscosity in the horizontal directions. For DNS simulations, this is the molecular kinematic viscosity of the fluid. In general, $\nu_{xy} = \nu_{xy}(\mathbf{x}, t)$.

ν_z The kinetic turbulent eddy viscosity in the vertical direction. For DNS simulations, this is the molecular kinetic viscosity of the fluid. In general, $\nu_z = \nu_z(\mathbf{x}, t)$.

$\boldsymbol{\phi}$ A collection of tracer fields. For example, we usually reserve $\boldsymbol{\phi} = [\phi_N, \phi_P, \phi_Z, \phi_D]$ for the biological tracers.

ρ The full, 3D density field $\rho = \rho(\mathbf{x}, t)$.

ρ_0 The mean density field $\rho_0 = \frac{1}{\int_\Omega d\Omega} \int_\Omega \rho d\Omega$.

ρ' The perturbation density field $\rho' = \rho - \rho_0$.

$\mathbf{F}$ An external forcing function to the momentum equations. $\mathbf{F} = \mathbf{F}(\mathbf{x}, t)$.

H The absolute distance from the ocean mean surface elevation to the sea floor. $H = H(\mathbf{x})$.

P_{hyd} The depth-integrated hydrostatic, 3D pressure field $P_{hyd} = P_{hyd}(\mathbf{x}, t) = \int_z^\eta p_{hyd} d\zeta$.

Re The Reynolds number, $\mathrm{Re} = \frac{|\mathbf{v}|D}{\nu}$.

S The salinity field $S = S(\mathbf{x}, t)$.

T The temperature field $T = T(\mathbf{x}, t)$.

$\mathbf{U}$ The depth-integrated horizontal components of the velocity field $\mathbf{U} = \int \mathbf{u}(x, y, \zeta, t) d\zeta$.

W The depth-integrated vertical component of the velocity field $W = \int w(x, y, \zeta, t) d\zeta$.

d The number of spatial dimensions.

f The Coriolis parameter. In general, $f = f(x, y)$ on the β-plane.

$\mathbf{g}$ The gravitational constant. $\mathbf{g} = [0, 0, -g]$, $g \approx 9.81 m/s^2$.

$\mathbf{g}_D$ Dirichlet boundary condition value for vector field.

g_D Dirichlet boundary condition value for scalar field.

$\mathbf{g}_N$ Neumann boundary condition value for vector field.

g_N Neumann boundary condition value for scalar field.

p The full, 3D pressure field $p = p(\mathbf{x}, t)$.

p' The non-hydrostatic, 3D pressure field $p' = p - p_{hyd}$.

p_s The rigid-lid surface pressure.

p_{hyd} The hydrostatic, 3D pressure field $p_{hyd} = p_{hyd}(\mathbf{x}, t) = \int_z^\eta \rho(x, y, \zeta, t) g d\zeta$.

$\mathbf{u}$ The 3D divergence-free, horizontal components of the velocity field $\mathbf{u} = [u(\mathbf{x}, t), v(\mathbf{x}, t), 0]$.

$\mathbf{v}$ The 3D divergence-free velocity field $\mathbf{v} = [u(\mathbf{x}, t), v(\mathbf{x}, t), w(\mathbf{x}, t)]$.

w The 3D divergence-free, vertical component of the velocity field $w = [0, 0, w(\mathbf{x}, t)]$.

THIS PAGE INTENTIONALLY LEFT BLANK

List of Discretization Symbols

$\bar{\bullet}$ The first predictor of $\bullet$. For example, the first velocity predictor is $\bar{\mathbf{v}}$.

$\bar{\bar{\bullet}}$ The second predictor of $\bullet$. For example, the second velocity predictor is $\bar{\bar{\mathbf{v}}}$.

$\delta\bullet$ The corrector to $\bullet$. For example, the pressure correction is δp.

$\bullet_\varepsilon$ The test-function restricted to the HDG edge-space, for example, the edge-space equivalent of θ, is $\{\theta_\varepsilon \in L^2(\varepsilon) : \theta_\varepsilon|_e \in \mathcal{P}^{\mathrm{p}}(e) \forall e \in \varepsilon\}$.

$\hat{\bullet}$ The DG edge-flux of $\bullet$. For example, the DG edge-flux for the pressure gradient is $\hat{\mathbf{q}_p} = \mathbf{q}_p + \tau_p(P - \lambda_P)$.

$\langle \bullet_i, \bullet_j \rangle_e$ The integration over a single discontinuous HDG edge-element. $\langle \bullet_i, \bullet_j \rangle_e = \int_e \bullet_i \bullet_j \, d\mathbf{x}$.

$\langle \bullet_i, \bullet_j \rangle_\varepsilon$ The integration over all discontinuous HDG edge-element. $\langle \bullet_i, \bullet_j \rangle_\varepsilon = \sum_k \int_{e_k} \bullet_i \bullet_j \, d\mathbf{x}$.

$(\bullet_i, \bullet_j)_K$ The integration over a single discontinuous element. $(\bullet_i, \bullet_j)_K = \int_K \bullet_i \bullet_j \, d\mathbf{x}$.

$\langle \bullet_i, \bullet_j \rangle_{\partial K}$ The integration over all the edges of a single discontinuous element. $\langle \bullet_i, \bullet_j \rangle_{\partial K} = \sum_k \int_{\partial K_l^k} \bullet_i \bullet_j \, d\mathbf{x}$.

$[[\bullet]]$ The jump of the solution across discontinuous elements. $[[\bullet]] = \bullet^+ + \bullet^-$.

$\{\{\bullet\}\}$ The mean of the solution across discontinuous elements. $\{\{\bullet\}\} = \frac{\bullet^+ + \bullet^-}{2}$.

$a\Delta t$ The numerical time-step size multiplied by constant a which varies depending on the time-integration method used.

Δt The numerical time-step size.

Ω The domain of interest.

$\partial\Omega$ The boundary of the domain of interest.

$\boldsymbol{\Theta}$ The tensor test-function. $\{\boldsymbol{\Theta} \in (L^2(\Omega))^{d\times d} : \boldsymbol{\Theta}|_K \in (\mathcal{P}^p(K))^{d\times d} \forall K \in \mathcal{T}_h\}$.

α The weighting function that selectively applies limiting/filtering.

ε The HDG edge-space. That is, $\varepsilon = \cup\partial K$.

ε° The HDG edge-space on the interior of the domain, excluding the boundaries. That is, $\varepsilon^\circ = \varepsilon \setminus \varepsilon^\partial$.

ε^∂ The HDG edge-space on the boundary of the domain. That is, $\varepsilon^\partial = \varepsilon \cup \partial\Omega$.

$\boldsymbol{\lambda}$ The velocity defined on the HDG edge-space $\boldsymbol{\lambda} \in \varepsilon$.

$\lambda_{\delta p}$ The pressure correction defined on the HDG edge-space $\lambda_{\delta p} \in \varepsilon$.

λ_p The pressure defined on the HDG edge-space $\lambda_p \in \varepsilon$.

$\lambda_{\delta p'}$ The non-hydrostatic pressure correction defined on the HDG edge-space $\lambda_{\delta p'} \in \varepsilon$.

$\boldsymbol{\lambda}_{xy}$ The horizontal velocity defined on the HDG edge-space $\boldsymbol{\lambda}_{xy} \in \varepsilon$.

λ_z The vertical velocity defined on the HDG edge-space $\lambda_z \in \varepsilon$.

τ The HDG stability parameter for velocity diffusion $\tau = \tau(\mathbf{x})$.

τ_p The HDG stability parameter for the pressure equation $\tau_p = \tau_p(\mathbf{x})$.

θ The scalar test-function. $\{\theta \in L^2(\Omega) : \theta|_K \in \mathcal{P}^p(K) \forall K \in \mathcal{T}_h\}$.

$\boldsymbol{\theta}$ The vector test-function. $\{\boldsymbol{\theta} \in (L^2(\Omega))^d : \boldsymbol{\theta}|_K \in (\mathcal{P}^p(K))^d \forall K \in \mathcal{T}_h\}$.

$\mathbf{D}$ The discrete derivative matrix. Defined as $\mathbf{D} = \mathbf{M}^{-1}\mathbf{S}^T$

$\mathbf{I}$ The $d \times d$ identity tensor.

$\mathbf{J}$ The Jacobian, that is, the determinant of the Jacobian matrix, $\mathbf{J} = \det\left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right)_K$

$\mathbf{J}_e$ The edge Jacobian, that is, the determinant of the Jacobian matrix, $\mathbf{J} = \det\left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right)_{\partial K}$

K A discrete element in $\mathcal{T}_h$. That is, $\mathcal{T}_h = \cup K_i$.

∂K The boundary of an element K, composed of a number of edges.

$\mathbf{L}$ The edge lifting matrix. Defined as $\mathbf{L} = \mathbf{M}^{-1}\mathbf{M}_e$

$\mathbf{M}$ The mass-matrix. Defined as $\mathbf{M} = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_{K^{\text{ref}}}$

$\mathbf{M}_e$ The mass-matrices of edges. Defined as $\mathbf{M}_e = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_{\partial K^{\text{ref}}}$

$\mathbf{M}_\varepsilon$ The mass-matrix for an edge. Defined as $\mathbf{M}_\varepsilon = (\boldsymbol{\theta}_{\varepsilon,i}, \boldsymbol{\theta}_{\varepsilon,j})_{\varepsilon^{\text{ref}}}$

P $\mathsf{P}\bullet$ is the L^2 projection of the boundary condition $\bullet$ into the HDG edge–space.

$\mathbf{Q}$ The scaled velocity gradient tensor $\mathbf{Q}_{ij} = \nu\frac{\partial \mathbf{v}_j}{\partial x_i}$.

$\mathbf{S}$ The stiffness matrix. Defined as $\mathbf{S} = (\boldsymbol{\theta}_i, \nabla\boldsymbol{\theta}_j)_{K^{\text{ref}}}$

$\mathcal{T}_h$ The triangulation of the domain Ω into a set of non-overlapping elements K.

$\mathcal{V}$ The modal basis Vandermonde matrix $\mathcal{V}_{ij} = \theta_j^M(\mathbf{x}_i)$, where θ_j^M is the j^{th} modal polynomial, and $\mathbf{x}_i$ is the i^{th} nodal point.

e The unique HDG edge element existing between K^+ and K^-. That is, $e = \partial K^+ \cap \partial K^-$.

e Euler's constant. Usually used as a function, e^x.

$\hat{\mathbf{n}}$ The outward-pointing normal from a face. $\hat{\mathbf{n}} = \hat{\mathbf{n}}(\mathbf{x}, t)$.

$\hat{\mathbf{n}}_{xy}$ The horizontal components of the outward-pointing normal from a face. $\hat{\mathbf{n}}_{xy} = [n_x, n_y, 0]$.

$\hat{\mathbf{n}}_z$ The vertical component of the outward-pointing normal from a face. $\hat{\mathbf{n}}_z = [0, 0, n_z]$.

$\mathbf{p}$ The largest degree of the finite-element polynomial basis. A p^{th} order basis has an order of accuracy $\mathcal{O}(\mathrm{p}+1)$

$\mathbf{q}_{\delta p}$ The gradient of the pressure correction $\mathbf{q}_{\delta p} = \nabla \delta p$.

$\mathbf{q}_{\delta p'}$ The gradient of the non-hydrostatic pressure correction $\mathbf{q}_{\delta p'} = \nabla \delta p'$.

$\mathbf{q}_p$ The pressure gradient $\mathbf{q}_p = \nabla p$.

$\mathbf{q}_z$ The scaled horizontal velocity vertical derivative vector $\mathbf{q}_z = \nu_z \frac{\partial \mathbf{u}}{\partial z}$.

q_z The scaled vertical velocity vertical derivative vector $q_z = \nu_z \frac{\partial w}{\partial z}$.

Chapter 1

Introduction

Numerical models are playing an increasingly important role in engineering and scientific applications. The demand for larger, more complex, and increasingly important problems requires more accurate modeling. While the steady increase in computational resources allow the solution of larger problems, advances in the underlying numerical methods allow the solution of previously intractable problems. Recognizing the importance of application-specific considerations when designing numerical models, this thesis develops new numerical discretization schemes for regional ocean modeling applications.

Ocean modeling presents unique challenges not commonly present in classical mechanical engineering fluid flows. For forecasting applications, ocean modelers are interested in knowing the ocean state at a future time and in understanding its dynamical evolution. In mechanical engineering applications, the mean state or average performance of a device is often desired. As an analogy, forecasting the ocean state would be like predicting the particular flame shape in a combustion chamber. Recently, ocean modelers have also become interested in predicting the statistics of the future ocean state, including the most probable state. Both the future state and its uncertainty are then predicted. A second consideration is that oceanic flows span a wider range of spatial and temporal scales than most engineering flows, from millimeters to planetary length scales, and seconds to geological timescales. Finally, many oceanic flows are in near-equilibrium states. For example, many regional mid-

latitude flows are in are in near-geostrophic balance and near-hydrostatic balance. Near-equilibrium flows are difficult to model since small discretization errors can grow to become relatively large forcing terms. As such, the numerical treatment of ocean flows require unique considerations not commonly present in classical engineering flows.

Notwithstanding the challenges, there has been major progress in simulation for ocean applications. These application span multiple engineering and scientific fields, including scientific studies of tidal to mesoscale interactions, shelf-slope exchanges, and climate dynamics. Additionally, ocean simulation is important for naval operations and national security as well as billion dollar industries such as fisheries, energy, tourism, and shipping. Then, predicting the environmental impacts of pollution as well as extreme weather events are important for human health and safety. These applications are made possible at their present level of sophistication due to the improvement of data from ocean measurements, the improvement of mathematical models, and the steady increase in available computational resources.

We are specifically motivated by quantitative simulation studies of fundamental nonlinear biological-physical dynamics in coastal regions with complex bathymetric features such as straits, sills, ridges, and shelfbreaks. Such features strongly affect flows and, if they are shallow enough, one can expect biological responses in the euphotic zone. Multiple physical scales exist for these flows, from rapid tidal effects to slower water-mass driven overflows, and biological resonances at some of these scales are likely. Additionally, observations of phytoplankton blooms from satellite imagery show the patchiness of biological activity. Due to the complex geometry, the range of scales, and the non-linear dynamics, quantitative simulations studies for this application would represent a major advance in numerical ocean simulation technology.

Major research questions today involve multiple scales and complex processes, and increasingly sophisticated simulations are required to push the limits of scientific knowledge and engineering applications. There are various strategies for reducing errors in present simulations, including: the increase of ocean measurements to reduce

initial/boundary condition error; the increase in computational resources to reduce the grid-size and numerical error of existing models; the refinement of mathematical models to more closely represent the physical system and reduce modeling error; and the improvement of the numerical schemes to allow the simulation of previously intractable problems and reduce numerical error. The first two strategies are costly because both require large capital investments. The next two strategies are relatively inexpensive, but require intellectual and research investments. However, since present operational ocean models are aging, there is a need and opportunity to improve their underlying computational fluid dynamics technology. Specifically, unstructured or variable resolution grids, ubiquitous in modern day mechanical engineering applications, can be incorporated along with high-order accuracy. This strategy may dramatically impact the sophistication of current simulations, enable new applications, and at a relatively low cost. As such, in this thesis we focus on schemes that reduce the numerical errors incurred when solving the mathematical model, but we will also include additional physics.

Some key questions arise when designing a new ocean simulation model. These include: Are more accurate numerical schemes needed?, How will more accurate numerical schemes impact simulations?, Why utilize unstructured grids?, and why use higher-order schemes? We can address the first two questions in relation to our application of interest. The coupled biological-physical dynamics problem is non-linear, and its solution may exhibit chaotic behavior. As such, we require accurate numerical methods, since small numerical errors can significantly affect quantitative modeling studies. Such methods should be capable of accurate simulations of nonlinear dynamics up to the predictability limit, which can be a relatively long period of time. For the second question, we can illustrate the effect of numerical error using a simplified system. Consider a dynamical system with a steady limit cycle and two components A and B Fig. [1-1]. Small numerical errors integrated over time may cause phase errors (top Fig. [1-1]), but the more insidious problem is the modification of the dynamics (bottom Fig. [1-1]). In this case, the concern is that the numerical errors modify the underlying dynamics described by the mathematical model and its param-

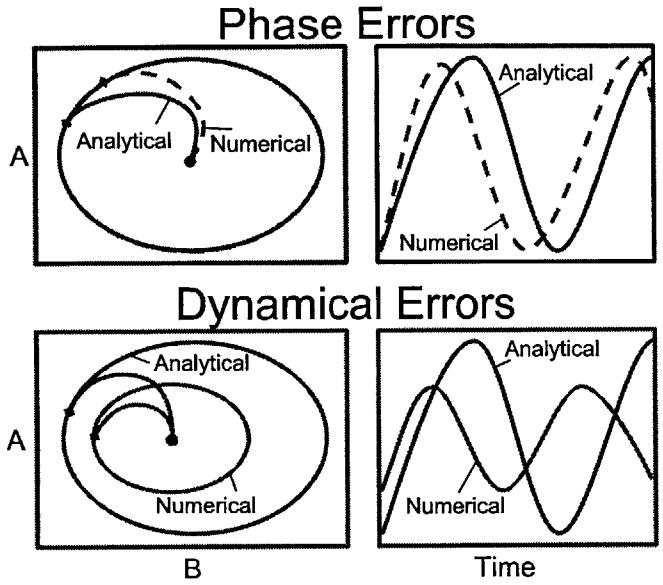


Figure 1-1: Numerical errors types for a two-component (A and B) dynamical systems with steady limit cycles. Phase plots (left column) and time evolution of component A (right column) are sketched for analytical (blue line) and numerical (orange line) solutions, illustrating phase errors (top row) and dynamical errors (bottom row).

eters. We will demonstrate this problem for an idealized system in §6.4.6. Presently, model parameters are tuned by comparing the numerical simulation output to measurements. Without an accurate numerical solution of the mathematical model, these tuned parameters will also need to correct for numerical error, which may have no direct relevance to the physical system. Thus, by using accurate numerical methods, the mathematical model and its parameters can be more directly compared to measurements from the physical system, leading to refinement of the model, and ultimately improving our understanding of the ocean. Therefore, accurate numerical schemes are needed for our application to allow phase-resolved simulations of the true biological-physical dynamics in coastal regions.

To answer the third question, unstructured grids can impact the accuracy of ocean simulation studies in a number of ways. First, unstructured grids can increase the range of resolved scales for multi-scale studies. While structured grid models can achieve the same effect using nesting approaches, unstructured grids are considerably more flexible, needing fewer degrees of freedom for the same resolution. Additionally,

an unstructured grid can more accurately capture complex geometrical features such as coastlines and steep bathymetry. Finally, by using much larger elements near open boundaries, the location of open boundaries can be moved further away from the model domain of interest for a small computational cost. This would serve the purpose of reducing the impact of the open boundary condition on the simulation. The main disadvantage of unstructured grids is the increased difficulty of efficient implementations, and its higher computational cost per degree of freedom. However, since fewer degrees of freedom may be required, the overall simulation cost may be similar.

For the last question, higher-order methods can be much more accurate at the same efficiency compared to lower order methods. To illustrate this point, we can use the 1D Sommerfeld wave equation

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = 0,$$

with initial condition

$$\phi(t = 0, x) = \mathrm{e}^{-10x^2},$$

on the periodic domain [-1,1] with $c = 2$. If we evolve this equation, advecting this initial condition through the domain 20 times, we can see that the high-order scheme is much more accurate compared to the lower-order schemes using the same number of degrees of freedom Fig. [1-2], which is approximately the same wall-clock time for our implementation. A similar observation can be made when comparing the schemes at the same number of flops. Present ocean models (and commercial CFD codes) predominantly use lower-order schemes. These schemes are usually formulated to be second order accurate on structured grids, but often reduce to sub-second-order accuracy on deformed meshes. As such, the best-case scenario for present ocean models is the middle column of Fig. [1-2]. While the advantage of high-order schemes are well documented in literature, their use has not yet become widespread due to their relative difficulty in implementation and stabilization (Vincent and Jameson,

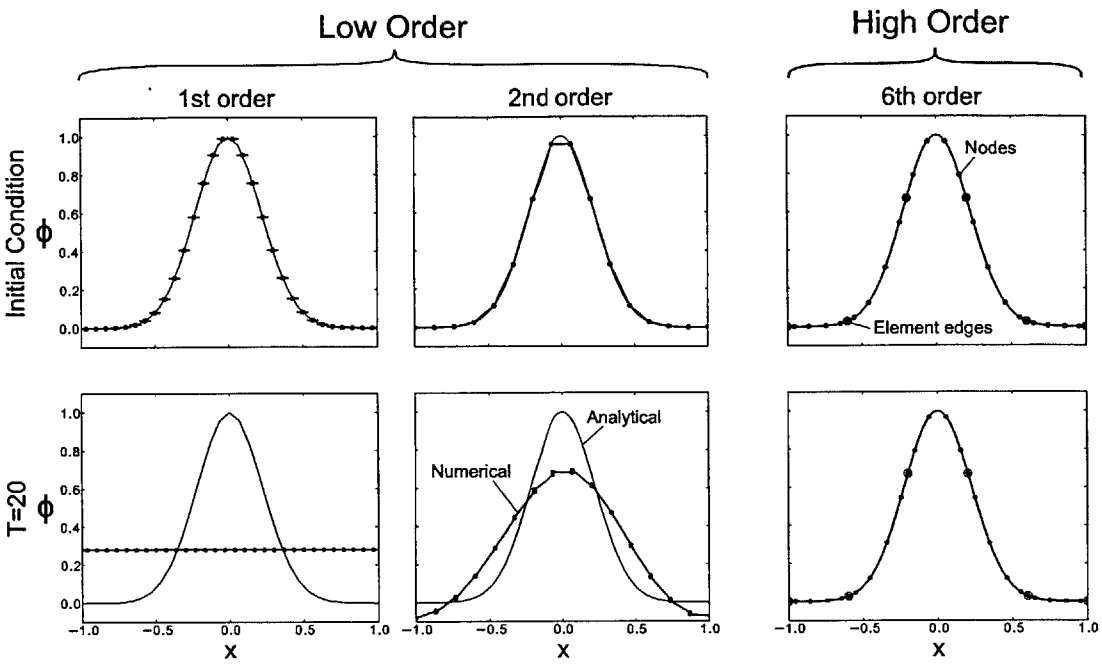


Figure 1-2: The 1D advection of a Gaussian bell over a periodic domain for various discretizations. The advantage of high-order is illustrated; even after advecting the initial conditions 20 time through the domain, the high-order scheme retains the amplitude and shape of the bell, while the lower order solutions have visibly large errors.

2011). Both of those issues are addressed in this thesis.

Now let us briefly describe the state-of-the-art in ocean models (for a more thorough review see §4, Griffies et al. (2000), Ueckermann (2009), Griffies et al. (2010)). In Fig. [1-3] we have categorized the majority of popular ocean models, although the list is not exhaustive. Present operational ocean models solve rather similar geophysical fluid dynamics equations and use similar numerical methods, essentially all based on the schemes proposed by Bryan (1969) and others in the early seventies. These models were developed for various applications, from basin scales to estuaries, and they predominantly use low-order methods on structured grids (top-left box in Fig. [1-3]). In the late nineties to early two thousands, the availability of increased computational resources encouraged the development of new models for applications with multi-scale interactions and in coastal regions with complex geometries (see §4). This newer class of models utilize unstructured grids, but still predominantly use low-order

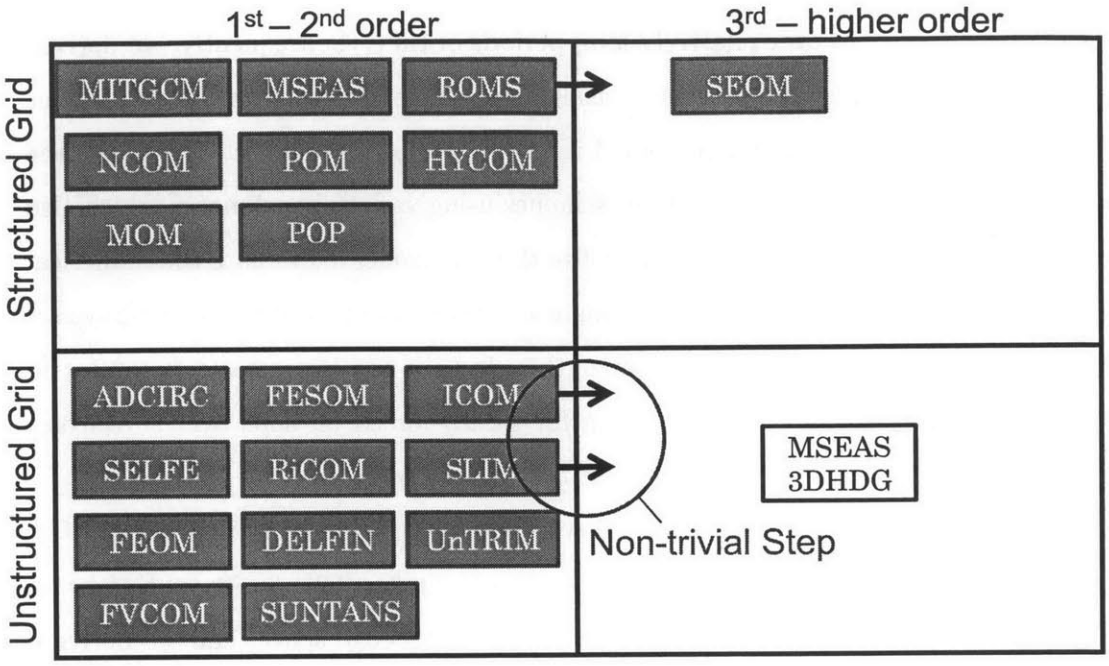


Figure 1-3: Various popular ocean models are categorized by their grid type (structured or unstructured) and order of accuracy. To our knowledge, our MSEAS-3DDG is the only code developed specifically as a high-order, unstructured grid model.

methods (bottom-left box in Fig. [1-3]). Recently, a number of researchers have begun the pursuit of higher-order accurate schemes. In particular, the Spectral Element Ocean Model (SEOM) is high-order accurate on quadrilateral grids (top-right box in Fig. [1-3]), while the Regional Ocean Modeling System (ROMS) community model is updating its model for higher order accuracy. Additionally, the Imperial College Ocean Model (ICOM) and the Second-generation Louvain-la-Neuve Ice-ocean Model (SLIM) are based on finite element methods and have demonstrated some high-order accurate results. However, in our experience, the leap from a low-order to a high-order model is not a trivial one. As such, our aim for the MSEAS-3DDG code was to develop a stable, truly high-order accurate ocean model on unstructured grids (bottom-right box in Fig. [1-3]).

These considerations led us to develop novel high-order unstructured grid numerical schemes for regional ocean modeling. Our goal was to develop, verify, and utilize a new regional ocean modeling code capable of accurately discretizing a wide

range of spatial scales over relatively long periods of time. Consequently, we derived and developed a discretization scheme using the promising new hybrid discontinuous Galerkin (HDG) method, and combined it with a projection method. We verify the formulation and implementation of our schemes using various benchmarks which also test the numerical stability. Then, we utilize the new model to evaluate the numerical requirements for accurate high-order simulation studies of coupled biological-physical dynamics. We also study and characterize the effects of non-hydrostatic dynamics on plankton productivity and concentration for tidally-forced motions over subsurface ocean banks for various stratification and tidal amplitudes.

The presentation is organized as follows. We begin by deriving the discrete formulation of our HDG projection method for Navier-Stokes (§2). In §2, we also prove the consistency of our velocity correction on the HDG edge space, and we derive a consistent HDG stability parameter for the pressure. Following this, in §3, we describe the implementation of our new quadrature-free integration scheme which is consistent with HDG, we develop our novel selective nodal limiter, and we evaluate our new schemes with benchmarks and convergence studies. In particular, we use a tracer advection benchmark, manufactured solutions for steady diffusion and Stokes equations, and the lock-exchange benchmark. In §4, we extend our HDG projection method scheme to regional ocean flows. This extension includes the non-linear free-surface, Coriolis forcing, and hydrostatic and non-hydrostatic pressures. Then in §5 we describe the extension of our implementation for regional ocean flows. This includes numerical consistency and conservation for the advection and free-surface, and the discretization of the explicit terms for vertical integration, the hydrostatic pressure, the horizontal diffusion, and the Coriolis force. In §6, we evaluate the numerical requirements of the coupled biological-physical dynamics. This is done for an idealized strait. Following this, we study effects of non-hydrostatic flows and pressure on biological productivity and concentrations over coastal banks in §7. Finally, in §8, we present our conclusions and suggest possible avenues of future research.

Chapter 2

Discrete Formulation of 3D Navier-Stokes Equations Using Projection Methods and Hybridized Discontinuous Galerkin Finite Elements

Solving systems of equations that govern fluid flows is required for a vast number of engineering applications, from designing microfluid devices to predicting ocean dynamics, the weather, and climate on Earth. Unfortunately, these equations are difficult to solve accurately and quickly. A common trend in science and engineering applications is to attempt larger, more complex, and increasingly important problems that require more accurate answers. The need for novel numerical schemes that fully utilize new computational architectures is therefore crucial. Our aim is to develop a new class of numerical schemes which combines the recently developed hybrid discontinuous Galerkin method with the well-studied Projection method to obtain a new high-order accurate method with excellent efficiency for ocean applications.

A major challenge with the efficient solution of the incompressible Navier Stokes

equations lies in the coupling of the velocity and pressure components. Considering first the momentum equations, an important consideration is the stiffness of each dynamical term in the equations. This is because the integration of differential equations can often be very efficient if the stiff terms are treated implicitly, while the non-stiff terms are treated explicitly. For the conservation of momentum this means implicit diffusion and explicit advection. These equations can then be efficiently solved (often without a preconditioner) because the discrete matrices obtained for the implicit terms are diagonally dominant, when the Reynolds number is sufficiently large. Unfortunately, even with this efficiency for the momentum conservation equations, the second conservation equation for mass (the continuity or incompressibility constraint) leads to a coupling of the different velocity components with the pressure. The continuity constraint is not as easily invertible as the implicit diffusion terms in the momentum equations. As such, a large system of equations containing all components of velocity and pressure needs to be solved simultaneously.

One approach to address the challenge is to separate the numerical solution of the coupled equations. The common method is to time-split the solution of the momentum and continuity equations. Projection methods, pioneered by Chorin (1968) and Témam (1969) do just that. Using a projection method decouples the equations for solving the velocity components and pressure. As a result, instead of solving one large system, decoupled smaller systems can be solved. Unfortunately, the solution of the pressure equation often dominates the cost of the scheme within a time-step. Consequently, it is important to solve the resulting Poisson equation efficiently. Herein we will consider a class of projection methods where the diffusion terms are treated implicitly, while the remainder of the terms in the equation are treated explicitly, and the pressure will be handled through the particular projection scheme.

Discontinuous Galerkin (Hesthaven and Warburton, 2008) methods are attractive because they can be high-order accurate on arbitrary meshes. A high-order accurate numerical scheme reaches a smaller error tolerance with fewer degrees of freedom than low-order accurate schemes. They also promise to be more efficient on new computational architectures because the computation to memory ratio is higher, and

present computations are often limited by the memory bandwidth. Additionally, the discontinuous Galerkin method is well suited to advection-dominated problems because upwinding can be used to stabilize the scheme. However, continuous Galerkin methods (Strang and Fix, 1973) are deemed to be less expensive for Poisson equations (Kirby et al., 2012), which led us to the hybrid discontinuous Galerkin (Nguyen et al., 2009a, Cockburn et al., 2009a) method, aiming for optimal combination of accuracy with cost, hence efficiency.

The hybrid discontinuous Galerkin (HDG) method overcomes the computational cost concerns associated with discontinuous Galerkin methods (DG). A major drawback to DG is that the degrees of freedom on the edge of an element are duplicated. The cost of inverting a matrix is usually related to the square of the number of degrees of freedom. Thus, compared to their continuous counterparts, DG methods are accused of being much more expensive. While this is true on the same mesh with the same number of elements, the issue of accuracy also needs to be considered for a fair comparison. Notwithstanding, the HDG method does not duplicate all of the degrees of freedom on a edge. Additionally, the HDG solution can be post-processed to obtain a solution that is one order of accuracy higher than the order of basis used (Nguyen et al., 2009a, Cockburn et al., 2009a). Overall, the HDG method promises to be an efficient solution method for the pressure equation, since fewer degrees of freedom will be present in the discrete matrix for the Poisson equation.

As such, we wish to combine Projection methods with HDG Finite Elements. We start by reviewing current literature on HDG, slope limiting, and Projection methods. Then we present the particular form of the Projection method we are using, and review selected theoretical aspects. Following this we derive the discrete finite element formulation. We also prove the consistency of our velocity correction on the HDG edge space, and derive a consistent HDG stability parameter for the pressure correction. Finally, we explain the modifications needed in order to use a Implicit-Explicit Runge Kutta time integrator.

2.1 Literature review

The first papers published on the HDG method were for second order elliptic equations (Cockburn et al., 2009a,b) and convection-diffusion equations (Nguyen et al., 2009a,b). Shortly after Cockburn and Gopalakrishnan (2009) published a derivation for Stokes flow, which was applied by Nguyen et al. (2010a), with an analysis following in Cockburn et al. (2011). The extension to the incompressible Navier–Stokes equations followed, see Nguyen et al. (2010b, 2011). The HDG method has also been applied to compressible flows (Peraire et al., 2010, 2011, Nguyen et al., 2013, Schütz and May, 2013). Additionally, (Huynh et al., 2013) used HDG for elliptic interface problems, and (Ueckermann, 2009, Palma, 2012) applied HDG for curved domains using elliptic and convection-diffusion equations.

The computational aspects of HDG, such as its implementation and efficiency compared to existing methods, has been explored by a number of researchers. Ueckermann and Lermusiaux (2010) examined various solver/preconditioner combinations, and found that a stabilized Bi-Conjugate gradient solver combined with a zero-fill-in incomplete-LU preconditioner performed well. Waluga and Egger (2012) implemented HDG in the DUNE framework by constructing a residual operator for an iterative linear solver. In Kirby et al. (2012) the discrete matrices for CG and HDG were constructed, and they found that HDG is computationally competitive on the same mesh for polynomial orders of five and higher when comparing only solution time and neglecting accuracy considerations. In Ahnert and Bärwolff (2013) HDG is compared to a finite-volume solver, and HDG is deemed competitive to finite volume because of higher-order accuracy and better stability properties. While it is still too early to judge whether HDG will be adopted for CFD, the current literature is promising.

Besides the theoretical and computational issues, there are a number of challenges facing the wide-spread adoption of high-order numerical schemes by industry and academia. One major issue is the subject of shock-capturing or slope limiting (Vincent and Jameson, 2011). Without stabilizing a high-order numerical scheme, numerical oscillations can lead to instabilities in the flow. There is an abundance of literature

on slope limiting, and here we do not provide an exhaustive list. For linear elements, there are a number of classical papers dealing with Total Variation Bounded (TVB) limiters (Cockburn and Shu, 1989, Cockburn et al., 1989, 1990, Cockburn and Shu, 1998b, 2001). More recently, higher order limiters have received much attention with WENO-type approaches (Qiu and Shu, 2005, Zhu et al., 2008) and higher-order limiting (Hoteit et al., 2004, Krivodonova, 2007, Michoski et al., 2011, Huerta et al., 2012). Alternatively, stabilization can be added using filtering (Hesthaven and Kirby, 2008, Ueckermann and Lermusiaux, 2010) or artificial dissipation (Persson and Peraire, 2006, Barter and Darmofal, 2007, 2010). Efforts to retain the full order of accuracy of the scheme away from the discontinuities have also been pursued (Blossey and Durran, 2008). However, a standard approach that retains high-order accuracy while capturing sub-cell shocks does not yet exist, and research in this area is active.

We derived a new incompressible Navier-Stokes solver using Projection methods and HDG discretizations, stabilized using a selective nodal limiter of our own design. While many different Projection methods exist (for a recent review see Guermond et al. (2006)), we use the scheme by Timmermans et al. (1996). We only consider a class of Projection methods that treat the diffusion term implicitly, while the remaining terms (such as the non-linear advection) is treated explicitly, and the pressure is handled through the method. In relation to the Projection method literature, we apply a new boundary condition for the pressure, which is related to that proposed by Gresho and Sani (1987) and recently used by Shirokoff and Rosales (2011). We also build on this work by illustrating the form of the splitting error and clearly explaining its origin. The issue of splitting errors was also theoretically examined in Denaro (2003). While incompressible Navier-Stokes solvers using HDG exist, ours is the first to use a Projection method solution approach. In this area we show how to correctly couple the spatial and temporal discretization for a numerically divergence-free, stable, and accurate solution algorithm. Finally, to stabilize our numerical solution while maintaining higher-order accuracy, we derive a selective slope limiting approach. The 3D solver we developed builds on and extends existing methods in literature. It is a step toward a new generation of coastal ocean simulator.

Next, we review selected theory and derivations relevant to the HDG discretization of the incompressible Navier–Stokes equations.

2.2 Projection methods: Selected theory and derivations

In this section we first present the particular form of projection method that we will employ, the rotational incremental form. We then derive the corresponding rotational correction. We also review the Helmholtz-Hodge decomposition while highlighting properties that should be maintained numerically. Finally, we discuss the issue of boundary conditions, and briefly touch on the inf-sup condition.

We consider the non-dimensionalized unsteady incompressible Navier-Stokes equations on a simply connected domain Ω within a finite time interval $[0, T]$.

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v} + \nabla p &= -\nabla \cdot \mathbf{v}\mathbf{v} + \mathbf{f} \quad \text{in } \Omega \times [0,T], \\ \nabla \cdot \mathbf{v} &= 0 \quad \text{in } \Omega \times [0,T], \\ \mathbf{v}|_{\partial\Omega} &= \mathbf{g}_D \quad \text{in } \partial\Omega \times [0,T], \\ \mathbf{v}|_{t=0} &= \mathbf{v}_0 \quad \text{in } \Omega, \end{aligned} \tag{2.1}$$

where $\mathbf{v} = [u, v, w]$ is the velocity, $p = \frac{1}{\rho_0} P$, P is the pressure, $\mathbf{f} = \mathbf{g}\rho$ when there is only density forcing, ρ_0 is the mean density, ρ is the density perturbation, $\mathbf{g}_D$ is some Dirichlet velocity boundary condition, and $\mathbf{v}_0$ is some initial condition for the velocity. For the Boussinesq equations, with density forcing, we need an additional tracer (internal energy) equation for the density:

$$\begin{aligned} \frac{\partial \rho}{\partial t} - \nabla \frac{1}{\mathrm{ReSc}} \cdot \nabla \rho &= -\nabla \cdot \mathbf{v}\rho + \mathbf{f}^\rho \quad \text{in } \Omega \times [0,T], \\ \rho|_{\partial\Omega} &= g_{D_\rho} \quad \text{in } \partial\Omega \times [0,T], \\ \rho|_{t=0} &= \rho_0 \quad \text{in } \Omega, \end{aligned} \tag{2.2}$$

where $\mathrm{Sc} = \frac{\nu}{\kappa}$ is the Schmidt number, which represents the ratio of kinematic viscosity

ν to molecular diffusivity κ. Alternatively, we can solve multiple tracer equations for temperature and salinity, then calculate the density through a state equation (as in §4). However, if the density is a linear combination of salinity and temperature and the Schmidt number is the same for both, we can use (2.2) as above. For the remainder of this chapter, we will focus on the momentum and continuity equations.

Now, since the non-linear term will be treated explicitly, it will not affect the splitting error, and we can group it with the right-hand-side forcing term. As such, we will only consider the Stokes equations henceforth.

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v} + \nabla p &= \mathbf{F}_{\partial t} \quad \text{in } \Omega \times [0,T], \\ \nabla \cdot \mathbf{v} &= 0 \quad \text{in } \Omega \times [0,T], \\ \mathbf{v}|_{\partial\Omega} &= \mathbf{g}_D \quad \text{in } \partial\Omega \times [0,T], \\ \mathbf{v}|_{t=0} &= \mathbf{v}_0 \quad \text{in } \Omega, \end{aligned} \tag{2.3}$$

where $\mathbf{F}_{\partial t} = -\nabla \cdot \mathbf{v}\mathbf{v} + \mathbf{f}$.

We now proceed with the time-discretization of these equations using the rotational incremental pressure correction scheme (Timmermans et al., 1996). There are many different variations of the projection method, and for a thorough review of the different methods see Guermond et al. (2006). We employ this version because it is among the most accurate available (to our knowledge). Considering for now a single step in time, the time-split equations start by solving for the predictor velocity $\bar{\mathbf{v}}^{k+1}$ using an old or guessed value for the pressure.

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \bar{\mathbf{v}}^{k+1} + \nabla p^k = \mathbf{F}^{k,k+1}, \tag{2.4}$$

$$\bar{\mathbf{v}}|_{\partial\Omega}^{k+1} = \mathbf{g}_D, \tag{2.5}$$

$$\mathbf{v}|_{t=0} = \mathbf{v}_0, \tag{2.6}$$

$$p|_{t=0} = p_0, \tag{2.7}$$

where a is some constant associated with the time-integration method, and $\mathbf{F}^{k,k+1}$ contains the explicitly calculated terms (including old values of $\mathbf{v}$) and the right-

hand-side forcing (2.3), see also §2.4). Note, here we only consider Dirichlet velocity boundary conditions, see §4 for other boundary conditions. Next, a Poisson equation is solved for the pressure (note, negative signs are added so that this derivation matches our numerical implementation, see §2.3):

$$-\nabla^2 \delta p^{k+1} = -\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \tag{2.8}$$

$$\left.\frac{\partial \delta p^{k+1}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega} = 0. \tag{2.9}$$

Finally, the velocities and pressure need to be corrected

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \nabla \delta p^{k+1}, \tag{2.10}$$

$$p^{k+1} = p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}. \tag{2.11}$$

We first note that the boundary condition for the pressure–correction (2.9) comes from (2.10) and a no–normal flow boundary condition on both $\mathbf{v}$ and $\bar{\mathbf{v}}$. Hence, the $\mathbf{v}$ velocity satisfies the normal Dirichlet boundary conditions ($\mathbf{v} \cdot \hat{\mathbf{n}}|_{\partial\Omega}^{k+1} = \mathbf{g}_D \cdot \hat{\mathbf{n}}$), and is divergence free ($\nabla \cdot \mathbf{v} = 0$), while $\bar{\mathbf{v}}$ satisfies both the normal and the tangential Dirichlet boundary conditions ($\bar{\mathbf{v}}|_{\partial\Omega}^{k+1} = \mathbf{g}_D$) but is not divergence free ($\nabla \cdot \bar{\mathbf{v}} \neq 0$). We then note that the final term in (2.11) is known as the rotational-correction term. As the origin of this term is important to the HDG spatial discretization, we explain its derivation next.

2.2.1 Origin of rotational-correction term

The $\frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}$ term in (2.11) arises because the diffusion term is treated implicitly, as a function of $\bar{\mathbf{v}}^{k+1}$. This means that the subsequent update to p (from continuity only) has to be modified based on this implicit momentum diffusion. As a corollary, if the advection terms in $\mathbf{F}^{k,k+1}$ had also been treated implicitly, the implicit advection of $\bar{\mathbf{v}}^{k+1}$ would then affect the updated pressure. This holds in general for any terms (in $\mathbf{F}^{k,k+1}$ or elsewhere) that depend on velocity if they are not divergence free and if they are treated implicitly instead of explicitly. This can be seen by taking the

difference between the final discretized equation (sum of all steps in the projection method) and the predictor step (to obtain $\bar{\mathbf{v}}^{k+1}$). These comments will be expanded upon later.

To explain the above mathematically, let's begin by writing the un-split equations, that is, the discrete equations that we want to solve

$$\frac{\mathbf{v}^{k+1}}{a\Delta t} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla\mathbf{v}^{k+1} + \nabla p^{k+1} = \mathbf{F}^{k,k+1}. \tag{2.12}$$

Then subtract out (2.4)

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla\bar{\mathbf{v}}^{k+1} + \nabla p^{k} = \mathbf{F}^{k,k+1},$$

which gives

$$\frac{\mathbf{v}^{k+1} - \bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla(\mathbf{v}^{k+1} - \bar{\mathbf{v}}^{k+1}) + \nabla p^{k+1} - \nabla p^{k} = 0,$$

noting that the forcing terms $\mathbf{F}^{k,k+1}$ are the same for the split and un-split equations. Now substituting for $\bar{\mathbf{v}}^{k+1}$ using (2.10) and canceling terms, we have

$$\nabla\delta p^{k+1} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla(a\Delta t\nabla\delta p^{k+1}) + \nabla p^{k} - \nabla p^{k+1} = 0.$$

Now, solve for ∇p^{k+1}

$$\begin{aligned}
\nabla p^{k+1} &= \nabla p^{k} + \nabla\delta p^{k+1} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla(a\Delta t\nabla\delta p^{k+1}) \\
&= \nabla p^{k} + \nabla\delta p^{k+1} - \nabla\left(\nabla\frac{1}{\mathrm{Re}} \cdot \nabla(a\Delta t\delta p^{k+1})\right) \\
&= \nabla\left(p^{k} + \delta p^{k+1} - \nabla\frac{1}{\mathrm{Re}} \cdot \nabla(a\Delta t\delta p^{k+1})\right),
\end{aligned} \tag{2.13}$$

where we have switched the order of the Laplacian and the gradient in the second equality, which is correct if we assume Re is a constant (since these operators apply

to the scalar δp). Next, we can substitute (2.8) to obtain

$$\nabla p^{k+1} = \nabla \left(p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1} \right). \tag{2.14}$$

To recover (2.11), we drop the gradient,

$$p^{k+1} = p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}. \tag{2.15}$$

Note, this is true up to a constant, since the gradient of a sum of constants is zero. Also, (2.14) can be projected onto the normal of the boundary $\partial\Omega$, giving:

$$\nabla p^{k+1} \cdot \hat{\mathbf{n}} = \nabla p^k \cdot \hat{\mathbf{n}} + \nabla \delta p^{k+1} \cdot \hat{\mathbf{n}} - \frac{1}{\mathrm{Re}} \nabla (\nabla \cdot \bar{\mathbf{v}}^{k+1}) \cdot \hat{\mathbf{n}}. \tag{2.16}$$

and with a no–normal Neumann boundary condition on δp^{k+1} we have:

$$\nabla p^{k+1} \cdot \hat{\mathbf{n}} = \nabla p^k \cdot \hat{\mathbf{n}} - \frac{1}{\mathrm{Re}} \nabla (\nabla \cdot \bar{\mathbf{v}}^{k+1}) \cdot \hat{\mathbf{n}}. \tag{2.17}$$

This equation (2.17) shows that any change in the normal pressure derivative at the boundary comes through the rotational correction term. If a non–zero Neumann condition was used for the pressure correction (2.9), then it would enter (2.16), but it would not be consistent with the zero normal–velocity boundary condition.

While the above derivation shows mathematically where the correction comes from, some intuition is useful, especially for later. When solving for the velocity predictor, (2.4), the right-hand-side forcing terms are not divergence free because at this stage we are not using the final pressure. Now, because the predictor velocity is divergent, the implicit diffusion term will operate on this divergent component of the velocity. The projection equation (2.8) sets the gradient of δp^{k+1} equal to the divergent part of the predictor velocity. So, with $\nabla \delta p^{k+1}$ we are removing both the divergent part of the right-hand-side forcing and the diffusion of the divergent part of the right-hand-side forcing. Hence, when correcting the pressure, we have to remember that the gradient of δp^{k+1} was operated on by the implicit diffusion term,

and this should not be a part of the pressure gradient; the pressure gradient should only correct for the divergent part of the right-hand-side and not also its diffusion. In other words, if we had used the final pressure gradient to start with, the whole right-hand-side would have been divergence free and the implicit diffusion would not operate on any divergent component. This suggests that evaluating the pressure by projecting out the divergence of the right-hand-side before taking the implicit time-step might be a better strategy. However, as we will show in §3.8, the standard projection does more than just remove the divergence present in the right-hand-side.

In this section we mathematically showed the origin of the rotational correction term, and gave an intuitive explanation of its function. Next we discuss some important theoretical properties.

2.2.2 Helmholtz-Hodge decomposition

First we will discuss the underlying decomposition that makes projection methods viable. The Helmholtz-Hodge decomposition states that any vector field on a simply connected domain can be uniquely decomposed into a divergence-free (solenoidal) part, and a curl-free (irrotational) part, see for example Girault and Raviart (1986), Denaro (2003). In our case, using the notation from above we have

$$\bar{\mathbf{v}}^{k+1} = \mathbf{v}^{k+1} + a\Delta t \nabla \delta p^{k+1}, \tag{2.18}$$

$$\Rightarrow \mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \nabla \delta p^{k+1}, \tag{2.19}$$

with boundary condition

$$\mathbf{v}^{k+1} \cdot \hat{\mathbf{n}} = 0. \tag{2.20}$$

where the vector field $\bar{\mathbf{v}}^{k+1}$ is composed of solenoidal $\mathbf{v}^{k+1}$ and irrotational $a\Delta t \nabla \delta p^{k+1}$ parts, and the re-arranged form (2.19) matches the velocity correction equation (2.10). The gradient of δp is irrotational because of the vector identity $\nabla \times \nabla \phi = 0$ for any field ϕ. The idea behind the projection method, then, is that the velocity solved from

the momentum equations using the incorrect pressure is divergent, and we wish to remove or project out this part.

In light of this decomposition we can better understand the mathematical role of the pressure in the incompressible Navier-Stokes equations. Consider, now, the divergence of the momentum equation in (2.3):

$$\begin{aligned} \nabla \cdot \frac{\partial \mathbf{v}}{\partial t} - \nabla \cdot (\nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v}) + \nabla^2 p &= \nabla \cdot \mathbf{F}_{\partial t} \\ \Rightarrow \frac{\partial \nabla \cdot \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla (\nabla \cdot \mathbf{v}) + \nabla^2 p &= \nabla \cdot \mathbf{F}_{\partial t} \end{aligned}$$

Applying the continuity constraint we obtain the pressure Poisson equation

$$\nabla^2 p = \nabla \cdot \mathbf{F}_{\partial t}.$$

This shows that the role of the pressure is to balance divergent terms in the right-hand side forcing such that the final velocity is divergence free. This leads to two important properties that need to be maintained in the numerical scheme.

1. If $\nabla \cdot \mathbf{F}_{\partial t} = 0$ then p is a linear function in space, or a constant if $\nabla p \cdot \hat{\mathbf{n}} = 0$.
2. If $\mathbf{F} = \nabla p$, then $\mathbf{v} = 0$, if the velocity is initially zero, and have zero Dirichlet boundary conditions.

We will test these properties in §3.8.

2.2.3 Boundary conditions for pressure

Classical pressure boundary condition form: Another issue that often arises is the question of boundary conditions for the pressure. The original equations did not require a pressure equation, but once we took an additional derivative of the momentum equations, we require additional boundary conditions. What, then, are the correct boundary conditions for the pressure? One way to find a pressure boundary condition is as follows. If we project the momentum equations, (2.3), unto the normals of the boundaries we obtain a boundary condition for the pressure that is "consistent" with

the interior equations:

$$\frac{\partial \mathbf{v}\cdot\hat{\mathbf{n}}}{\partial t} - \hat{\mathbf{n}}\cdot\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\mathbf{v} + \hat{\mathbf{n}}\cdot\nabla p = \hat{\mathbf{n}}\cdot\mathbf{F}_{\partial t},$$

In particular, if $\mathbf{v}\cdot\hat{\mathbf{n}}$ is given and constant (i.e. time invariant) at the boundary (e.g. no–slip boundary condition), we obtain

$$\Rightarrow \hat{\mathbf{n}}\cdot\nabla p = \hat{\mathbf{n}}\cdot\mathbf{F}_{\partial t} + \hat{\mathbf{n}}\cdot\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\mathbf{v}, \tag{2.21}$$

Similarly in the discrete sense, for the un–split (2.12) which is the time-discrete of (2.3), we obtain

$$\Rightarrow \hat{\mathbf{n}}\cdot\nabla p^{k+1} = \hat{\mathbf{n}}\cdot\mathbf{F}^{k,k+1} + \hat{\mathbf{n}}\cdot\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\mathbf{v}^{k+1}, \tag{2.22}$$

where we used the correct p^{k+1} instead of p^k as in (2.4). We note that (2.22) is thus the pressure boundary condition that the un–split equations will obey, and is therefore the boundary conditions we wish to satisfy.

In the classical presentation of the rotational incremental pressure correction method, no such "project the interior governing equations to the boundary" boundary conditions is usually explicitly mentioned. Instead, a boundary condition for the pressure correction (2.9) is specified instead, leaving the question of what boundary condition is implicitly imposed on the pressure. However, we note that in fact, the rotational increment gives a pressure boundary condition which is consistent with this projection equation (2.22). To prove this, we start from (2.4) and solve for the rotational correction term projected onto the normal of the boundary by expanding the diffusion term using the identity $\nabla^2 = -\nabla\times\nabla + \nabla(\nabla\cdot)$, and again assuming $\frac{1}{\mathrm{Re}}$

is a constant:

$$\mathbf{F}^{k,k+1} = \frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla\frac{1}{\mathrm{Re}}\cdot\nabla\bar{\mathbf{v}}^{k+1} + \nabla p^k, \tag{2.23}$$

$$= \frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} + \frac{1}{\mathrm{Re}}\nabla\times\nabla\times\bar{\mathbf{v}}^{k+1} - \frac{1}{\mathrm{Re}}\nabla\nabla\cdot\bar{\mathbf{v}}^{k+1} + \nabla p^k, \tag{2.24}$$

$$\Rightarrow \frac{1}{\mathrm{Re}}\nabla\nabla\cdot\bar{\mathbf{v}}^{k+1} = \frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} + \frac{1}{\mathrm{Re}}\nabla\times\nabla\times\bar{\mathbf{v}}^{k+1} + \nabla p^k - \mathbf{F}^{k,k+1}, \tag{2.25}$$

$$\Rightarrow \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\nabla\cdot\bar{\mathbf{v}}^{k+1} = \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\times\nabla\times\bar{\mathbf{v}}^{k+1} + \hat{\mathbf{n}}\cdot\nabla p^k - \hat{\mathbf{n}}\cdot\mathbf{F}^{k,k+1}. \tag{2.26}$$

Substituting (2.26) into (2.17) we recover (2.22), completing the proof

$$\hat{\mathbf{n}}\cdot\nabla p^{k+1} = \hat{\mathbf{n}}\cdot\nabla p^k - \frac{1}{\mathrm{Re}}\nabla(\nabla\cdot\bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}, \tag{2.27}$$

$$= \hat{\mathbf{n}}\cdot\nabla p^k - \left(\frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\times\nabla\times\bar{\mathbf{v}}^{k+1} + \hat{\mathbf{n}}\cdot\nabla p^k - \hat{\mathbf{n}}\cdot\mathbf{F}^{k,k+1}\right), \tag{2.28}$$

$$= -\frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\times\nabla\times\bar{\mathbf{v}}^{k+1} + \hat{\mathbf{n}}\cdot\mathbf{F}^{k,k+1}, \tag{2.29}$$

$$= \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\cdot\nabla\mathbf{v}^{k+1} + \hat{\mathbf{n}}\cdot\mathbf{F}^{k,k+1}. \tag{2.30}$$

where the final line follows from $\nabla\cdot\nabla\mathbf{v}^{k+1} = -\nabla\times\nabla\mathbf{v}^{k+1} = -\nabla\times\nabla\bar{\mathbf{v}}^{k+1}$. This proof also shows that without the rotational correction term, the pressure would contain an inconsistent numerical boundary layer $\hat{\mathbf{n}}\cdot\nabla p^{k+1} = \hat{\mathbf{n}}\cdot\nabla p^k$ (see also Guermond et al. (2006)).

New pressure boundary condition form with divergence-free predictor velocity: Using the boundary conditions from the rotational incremental pressure-correction algorithm, the remaining splitting error manifests itself through the tangential boundary velocity (Guermond et al., 2006). To explain this, we first note that the equation for the predictor velocity (2.4) is solved using the correct boundary conditions for velocity (2.5), but at this stage the velocity is divergent. When the predictor velocity is corrected using the pressure-correction-gradient $\mathbf{v}^{k+1} = \mathbf{v}^{k+1} - a\Delta t\nabla\delta p^{k+1}$ (2.10), the tangential boundary conditions are modified. The normal boundary conditions for the partial update of the pressure are not modified since we set $\nabla\delta p^{k+1}\cdot\hat{\mathbf{n}} = 0$ on the boundary (2.9), but the tangential boundary conditions are modified, since

$\nabla \delta p^{k+1} \cdot \hat{\mathbf{t}} \neq 0$. An example of this can be seen in Fig. [2-5], where no-slip boundary conditions are used, but after one time-step the final velocity does not vanish at the boundary. We cannot constrain the tangential pressure-correction gradient at the domain boundary since doing so will over-determine the system of equations, and as a result we would modify the normal pressure-correction gradient. Finally, we also see that this remaining splitting error in the tangential boundary condition is of the order $\mathcal{O}(\Delta t)$. This splitting error is inherent in the scheme, and as such an order of accuracy of greater than $\mathcal{O}(\Delta t^2)$ has not been achieved with standard projection methods (Guermond et al., 2006).

Motivated by this remaining time-splitting error on the tangential velocity, some researchers have argued that the boundary conditions should be split differently than (2.5) and (2.9). Specifically, Gresho and Sani (1987) proposed using $\nabla \cdot \mathbf{v}$ as an additional boundary condition, and Shirokoff and Rosales (2011) used this boundary condition to solve the split equations. In Shirokoff and Rosales (2011), they show that the split system of equations are equivalent to the original Navier-Stokes equations, and they claim that when using their boundary conditions, the scheme is not limited to second order accuracy. It is important to note that Shirokoff and Rosales (2011) used explicit diffusion for their method, and supported their claims with theory and numerical convergence studies. Since we are interested in implicit diffusion, we modified their approach, which gives the system of equations:

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \bar{\mathbf{v}}^{k+1} + \nabla p^k = \mathbf{F}, \tag{2.31}$$

$$\begin{aligned} \hat{\mathbf{n}} \times \bar{\mathbf{v}}|_{\partial\Omega}^{k+1} &= \hat{\mathbf{n}} \times \mathbf{g}_D, \\ \nabla \cdot \bar{\mathbf{v}}|_{\partial\Omega}^{k+1} &= 0 \end{aligned} \tag{2.32}$$

$$-\nabla^2 \delta p^{k+1} = -\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \tag{2.33}$$

$$\left.\frac{\partial \delta p^{k+1}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega} = \left.\frac{1}{a\Delta t} \hat{\mathbf{n}} \cdot (\mathbf{v}^{k+1} - \bar{\mathbf{v}}^{k+1})\right|_{\partial\Omega}, \tag{2.34}$$

with velocity and pressure corrector equations:

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \nabla \delta p^{k+1}, \tag{2.35}$$

$$p^{k+1} = p^k + \delta p^{k+1} - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}^{k+1}. \tag{2.36}$$

We note that while the governing equations and corrector equations remain unchanged ((2.31) = (2.4), (2.36) = (2.11), (2.35) = (2.10), and (2.36) = (2.11)), the boundary conditions have been modified ((2.32) $\neq$ (2.5), (2.34) $\neq$ (2.9)). Also, note that (2.34) comes from projecting the velocity corrector equation (2.35) onto the normal of the boundary, and this quantity $\frac{1}{a\Delta t}\hat{\mathbf{n}} \cdot (\mathbf{v}^{k+1} - \bar{\mathbf{v}}^{k+1})\big|_{\partial\Omega}$ is never expected to be zero, but would be small for fine discretizations. Additionally, we could take the divergence of (2.35) at the boundary, we obtain an alternate boundary conditions for the pressure correction, that is, $\nabla^2 \delta p^{k+1} = 0$. However, (2.34) is simpler to implement in a finite–element framework, and it assures that the final boundary condition for velocity is numerically satisfied.

The implementation of the velocity-divergence boundary condition is simplified for rectangular domains where the components of velocities are aligned with the walls. In this case, no-slip boundary conditions are enforced as follows. For the first step, set zero Dirichlet conditions for the tangential component of velocity and zero Neumann for the normal component of velocity. For the second step, set the pressure-correction boundary condition as the Neumann condition given in (2.34). Since this is easily tested for simple rectangular domains, we examine the effect of these boundary conditions in §3.8.

2.2.4 The inf-sup condition

Finally, we have some brief comments about the well-known inf-sup condition. This condition is a requirement on the discrete spaces where the pressure and velocity live. For the saddle-point problem (2.3) to be solvable, the discrete spaces require

that (see for e.g. Guermond et al. (2006))

$$\inf_{p} \sup_{\mathbf{v}} \frac{\int_\Omega p \nabla \cdot \mathbf{v}}{\|p\|_0 \|\mathbf{v}\|_1} \geq \beta, \tag{2.37}$$

for some $\beta > 0$, with the norms $\|\bullet\|_0 = \left(\int_\Omega \bullet^2\right)^{\frac{1}{2}}$, and $\|\bullet\|_1 = \left(\int_\Omega \nabla \bullet \cdot \nabla \bullet\right)^{\frac{1}{2}}$. If we integrate by parts the numerator we have

$$\begin{aligned}
\int_\Omega p \nabla \cdot \mathbf{v} &= \int_\Omega \nabla \cdot (p\mathbf{v}) - \int_\Omega \nabla p \cdot \mathbf{v}, \\
&= \int_{\partial\Omega} \hat{\mathbf{n}} \cdot (p\mathbf{v}) - \int_\Omega \nabla p \cdot \mathbf{v}, \\
&= -\int_\Omega \nabla p \cdot \mathbf{v},
\end{aligned}$$

and we see that this condition fails if the gradient of the pressure does not project completely unto the velocity-space (for uniform Dirichlet boundary conditions on the velocity). Fortunately, for discontinuous Galerkin finite elements this is not an issue because the pressure gradient and velocity spaces live in the same piece-wise polynomial space.

2.3 Spatial Discretization of Time-Split equations using HDG

In this section we spatially discretize the set of equation (2.4) - (2.11). We begin by defining our notation (also refer to the list of symbols starting after page 31). Then we complete the basic derivation of the scheme. Following this, we explain some of the less obvious terms in the discrete equations, that is the edge-space correction, and the rotational correction. Finally, we offer guidance on choosing the correct HDG stability parameter for the pressure poisson equation.

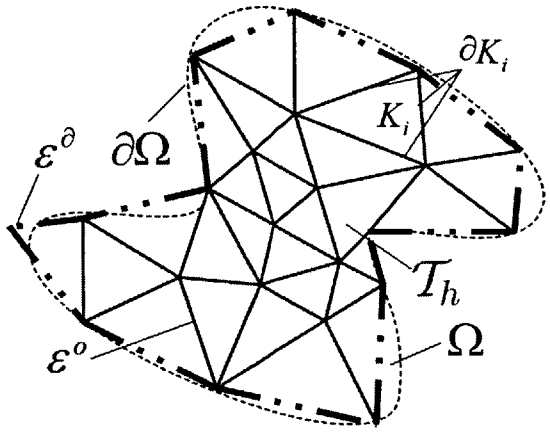


Figure 2-1: Notation for domain discretization.

2.3.1 Finite Element Definitions and Notation

We let $\mathcal{T}_h = \cup K_i$ be a finite collection of non-overlapping elements, K_i, that discretizes the entire computational domain Ω (Fig. [2-1]). Also, let $\partial\mathcal{T}_h = \{\partial K : K \in \mathcal{T}_h\}$ be the set of interfaces of all elements, where ∂K is the boundary of element K. For two elements sharing an edge K^+ and K^-, we define $e = \partial K^+ \cap \partial K^- \neq \emptyset$ as the unique interior interface between elements K^+ and K^-. For a single element K belonging to $\mathcal{T}_h$, if $e = \partial K \cap \partial\Omega \neq \emptyset$ it is a boundary interface. Let ε° and ε^∂ denote the set of unique interior and boundary interfaces, respectively, such that $\varepsilon = \varepsilon^\circ \cup \varepsilon^\partial$. We note that in the interior $\partial\mathcal{T}_h$ contains two interfaces, ∂K^+ and ∂K^-, at the same location (one for each element sharing the edge), whereas the set ε only contains a single interface, e, at the same location.

K^+ and K^- have outward pointing normals $\hat{\mathbf{n}}^+$ and $\hat{\mathbf{n}}^-$, respectively. We then let vector and scalar quantities $[\mathbf{a}^\pm, c^\pm]$ be the traces (i.e. the projections) of $[\mathbf{a}, c]$ on the interface e from the interior of $K^\pm$. The "mean" value $\{\{\bullet\}\}$ and "jumps" $[\![\bullet]\!]$ on the interior interfaces $e \in \varepsilon^\circ$ for scalar and vector quantities are then defined as

$$\{\{\mathbf{a}\}\} = (\mathbf{a}^+ + \mathbf{a}^-)/2 \qquad \{\{c\}\} = (c^+ + c^-)/2$$
$$[\![\mathbf{a}\cdot\hat{\mathbf{n}}]\!] = \mathbf{a}^+\cdot\hat{\mathbf{n}}^+ + \mathbf{a}^-\cdot\hat{\mathbf{n}}^- \qquad [\![c\hat{\mathbf{n}}]\!] = c^+\hat{\mathbf{n}}^+ + c^-\hat{\mathbf{n}}^-.$$

On the set of boundary interfaces $e \in \varepsilon^\partial$, (with outward facing normal $\hat{\mathbf{n}}$ on $\partial\Omega$), we

set these mean and jump quantities as

$$\{\{\mathbf{a}\}\} = \mathbf{a} \qquad \{\{c\}\} = c$$
$$[\![\mathbf{a} \cdot \hat{\mathbf{n}}]\!] = \mathbf{a} \cdot \hat{\mathbf{n}} \qquad [\![c\hat{\mathbf{n}}]\!] = c\hat{\mathbf{n}}.$$

since here $\mathbf{a}$ and c are single-valued. Note that the jump in a vector is a scalar (involving only the normal component of the vector), whereas the jump in a scalar is a vector. Additionally, the jump will be zero for a continuous function.

The main difference between continuous and discontinuous Galerkin lies in the approximation subspaces used. Discontinuous Galerkin uses bases that are in normed space $L^2(\Omega)$ while continuous Galerkin uses bases that are in the Hilbert space $H^1(\Omega)$, that is, the function has to be continuous across elements. For a function $f(\mathbf{x})$ to be in $L^2(\Omega)$, it has to satisfy $\int_\Omega f(\mathbf{x})^2 d\Omega < \infty$, whereas a function in $H^1(\Omega)$ has to belong to a smaller space satisfying $\int_\Omega f(\mathbf{x})^2 + \nabla f(\mathbf{x}) \cdot \nabla f(\mathbf{x}) d\Omega < \infty$.

Let $\mathcal{P}^{\mathrm{p}}(D)$ denote the set of polynomials of maximum degree p existing on a domain D. For example, we will be using $\mathrm{p} = 2$ to denote a second degree polynomial basis, which will result in a 3rd order accurate scheme. We introduce the discontinuous finite element bases we use on the element for scalars, vectors, and tensors, and these are defined as

$$\left\{\theta \in L^2(\Omega) : \theta\,|_K \in \mathcal{P}^{\mathrm{p}}(K), \forall K \in \mathcal{T}_h\right\}$$
$$\left\{\boldsymbol{\theta} \in (L^2(\Omega))^d : \boldsymbol{\theta}\,|_K \in (\mathcal{P}^{\mathrm{p}}(K))^d, \forall K \in \mathcal{T}_h\right\}$$
$$\left\{\boldsymbol{\Theta} \in (L^2(\Omega))^{d\times d} : \boldsymbol{\Theta}\,|_K \in (\mathcal{P}^{\mathrm{p}}(K))^{d\times d}, \forall K \in \mathcal{T}_h\right\},$$

respectively.

To use the HDG method, we will also require the traced finite element spaces

existing on the unique interfaces ε

$$\left\{\theta_\varepsilon \in L^2(\Omega) : \theta_\varepsilon \,|_e \in \mathcal{P}^{\mathrm{p}}(e), \forall e \in \varepsilon\right\},$$
$$\left\{\boldsymbol{\theta}_\varepsilon \in (L^2(\Omega))^d : \boldsymbol{\theta}_\varepsilon \,|_e \in (\mathcal{P}^{\mathrm{p}}(e))^d, \forall e \in \varepsilon\right\},$$
$$\left\{\boldsymbol{\Theta}_\varepsilon \in (L^2(\Omega))^{d\times d} : \boldsymbol{\Theta}_\varepsilon \,|_e \in (\mathcal{P}^{\mathrm{p}}(e))^{d\times d}, \forall e \in \varepsilon\right\}.$$

We also set $\{\theta_\varepsilon = \mathsf{P}\mathbf{g}_D \text{ on } \partial\Omega\}$, where P is the L^2 projection of the boundary condition $\mathbf{g}_D$ into the same space as θ_ε. Note that θ_ε is continuous on the interface, e, shared by K^+ and K^-, but discontinuous at the borders between different interfaces (that is, for a 1D–line–interface, discontinuities are the end–points of the line, see Fig. [2-3]). These HDG spaces will be used to define consistent fluxes that essentially serve as boundary conditions for the discontinuous elements. These consistent fluxes will be a function of $\boldsymbol{\lambda}$ variables (see §2.3.2), that live on these HDG spaces. Also, to obtain equations for $\bar{\boldsymbol{\lambda}}$ and $\lambda_{\delta p}$ in §2.3.2, we will invoke the conservation of fluxes across elements, which leads to global flux conservation (i.e. the fluxes on the interior ε° balance the fluxes from the boundary ε^∂).

Finally we define the inner products over continuous domains $D \in \mathbb{R}^d$ and $\partial D \in \mathbb{R}^{d-1}$ as

$$(\mathbf{a},\mathbf{b})_D = \int_D \mathbf{a}\cdot\mathbf{b}\,\mathrm{d}D \qquad (c,d)_D = \int_D cd\,\mathrm{d}D \tag{2.38}$$
$$\langle \mathbf{a},\mathbf{b}\rangle_{\partial D} = \int_{\partial D} \mathbf{a}\cdot\mathbf{b}\,\mathrm{d}\partial D \qquad \langle c,d\rangle_{\partial D} = \int_{\partial D} cd\,\mathrm{d}\partial D \tag{2.39}$$

for vector functions $\mathbf{a},\mathbf{b}$ and scalar functions c,d. Over discontinuous domains we also define

$$(\mathbf{a},\mathbf{b})_{\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h} (\mathbf{a},\mathbf{b})_K, \qquad \langle c,d\rangle_{\partial\mathcal{T}_h} = \sum_{K\in\mathcal{T}_h} \langle c,d\rangle_{\partial K}, \tag{2.40}$$

for vector functions $\mathbf{a}$, $\mathbf{b}$ defined on $\mathcal{T}_h$, and scalar functions c,d defined on $\partial\mathcal{T}_h$. We

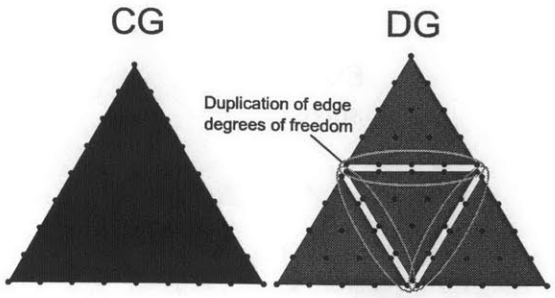


Figure 2-2: Discontinuos Galerkin have more degrees of freedom compared to continuous Galerkin on the same mesh.

will also require the additional inner product on the hybrid discontinuous domain

$$\langle \mathbf{a}, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon = \sum_{e \in \varepsilon} \langle \mathbf{a}, \boldsymbol{\theta}_\varepsilon \rangle_e \qquad \langle c, \theta_\varepsilon \rangle_\varepsilon = \sum_{e \in \varepsilon} \langle c, \theta_\varepsilon \rangle_e \tag{2.41}$$

for vector or scalar functions $\mathbf{a}$, c defined on ε.

The development of HDG methods (Nguyen et al., 2009a, Cockburn et al., 2009a) were motivated by the desire to improve the computational efficiency of discontinuous Galerkin methods compared to continuous Galerkin methods for elliptic problems. Often, the conclusion that DG is too expensive compared to CG is reached by comparing the number of degrees of freedom required for DG and CG on the same mesh. The DG discretization duplicates degrees of freedom on the edges of elements (Fig. [2-2]), which means that a larger matrix needs to be inverted compared to the CG case. This comparison is not necessarily fair, since the DG scheme may reach the same level of accuracy with a coarser mesh (or the same number of degrees of freedom). Nonetheless, it has been shown that the HDG method can be competitive with CG for elliptic problems (Waluga and Egger, 2012, Kirby et al., 2012). It does this by reducing the number of globally coupled degrees of freedom. The premise is that one can solve the equations of interest locally on an element as long as the initial and boundary conditions are properly specified. While the initial conditions for an element are specified as part of the problem, the boundary conditions on every element edge are not given. The HDG method specifies an equation for these unknown boundary conditions. To find this equation, the solution on the interior of

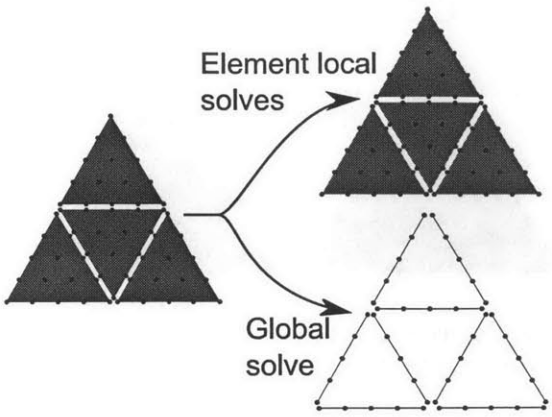


Figure 2-3: The HDG method splits the solution of the element local equations from the solution of the globally coupled problem for the boundary conditions.

an element is parameterized in terms of new variables which represent the boundary conditions for that element. These new variables live on the HDG edge-space θ_ε. The globally-coupled equation for the HDG edge variables are then found by solving for a boundary condition that would give conservative fluxes (that is, the same flux for both elements on either side of an edge). As such, the solutions of the element-local equations are split from the solutions of the globally-coupled equations for the edge degrees of freedom (Fig. [2-3]).

As an example, next we show the HDG discretization of the Poisson equation $\nabla^2\phi = f$. A standard step for DG methods is to introduce a new variable $\mathbf{q} = \nabla\phi$ and re-write this equation as a system of first-order PDE's:

$$\mathbf{q} - \nabla\phi = 0, \tag{2.42}$$

$$\nabla \cdot \mathbf{q} = f. \tag{2.43}$$

Now, the DG discretization of (2.42)–(2.43) using the notation introduced above is

$$(\mathbf{q}, \boldsymbol{\theta})_K - (\nabla\phi, \boldsymbol{\theta})_K - \left\langle \hat{\phi} - \phi, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K} = 0, \tag{2.44}$$

$$(\nabla \cdot \mathbf{q}, \theta)_K + \langle \hat{\mathbf{q}} - \mathbf{q}, \hat{\mathbf{n}}\theta \rangle_{\partial K} = (f, \theta)_K, \tag{2.45}$$

where the discretization is completed once we have defined the flux quantities $\hat{\phi}$ and

$\hat{\mathbf{q}}$. The HDG discretization uses the following flux definitions

$$\hat{\phi} = \lambda, \tag{2.46}$$

$$\hat{\mathbf{q}} = \mathbf{q} + \tau(\phi - \lambda)\hat{\mathbf{n}}. \tag{2.47}$$

When we substitute these flux definitions into the DG discretization (2.44)–(2.45), those equations become element-local if λ is known or given

$$(\mathbf{q}, \boldsymbol{\theta})_K - (\nabla\phi, \boldsymbol{\theta})_K + \langle\phi, \hat{\mathbf{n}} \cdot \boldsymbol{\theta}\rangle_{\partial K} = \langle\lambda, \hat{\mathbf{n}} \cdot \boldsymbol{\theta}\rangle_{\partial K}, \tag{2.48}$$

$$(\nabla \cdot \mathbf{q}, \theta)_K + \langle\tau\phi, \hat{\mathbf{n}}\theta\rangle_{\partial K} = (f, \theta)_K + \langle\tau\lambda, \hat{\mathbf{n}}\theta\rangle_{\partial K}. \tag{2.49}$$

To solve for λ, the HDG method defines a new equation on the HDG edge-space to conserve the diffusive flux, or, in the interior we have:

$$\langle[\![\hat{\mathbf{q}} \cdot \hat{\mathbf{n}}]\!], \theta_\varepsilon\rangle_\varepsilon = 0, \tag{2.50}$$

and when we substitute (2.48), this becomes

$$\langle[\![\mathbf{q} \cdot \hat{\mathbf{n}} + \tau(\phi - \lambda)]\!], \theta_\varepsilon\rangle_\varepsilon = 0. \tag{2.51}$$

The actual solution method is more involved; HDG methods require new implementation strategies (see Ueckermann (2009) and §3) and are inherently implicit (see §2.5). Next we proceed with the basic spatial discretization of the Stokes equations.

2.3.2 Discrete equations and their derivation

We start by introducing the variables (additional DG unknowns) $\mathbf{Q} = \frac{1}{\mathrm{Re}}\nabla\mathbf{v}$ and $\mathbf{q}_{\delta p} = \nabla\delta p$. We then rewrite (2.4) - (2.11) (excluding boundary conditions) as:

$$\mathrm{Re}\bar{\mathbf{Q}}^{k+1} - \nabla\bar{\mathbf{v}}^{k+1} = 0, \tag{2.52}$$

$$\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t} - \nabla\cdot\bar{\mathbf{Q}}^{k+1} + \nabla p^{k} = \mathbf{F}^{k,k+1}, \tag{2.53}$$

$$\mathbf{q}_{\delta p}^{k+1} - \nabla\delta p^{k+1} = 0, \tag{2.54}$$

$$-\nabla\cdot\mathbf{q}_{\delta p}^{k+1} = -\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \tag{2.55}$$

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}, \tag{2.56}$$

$$p^{k+1} = p^{k} + \delta p^{k+1} - \frac{1}{\mathrm{Re}}\nabla\cdot\bar{\mathbf{v}}^{k+1}. \tag{2.57}$$

These are the time–discretized split equations, and the starting point of our finite–element spatial discretization.

Next, we first give in (i) the basic DG discretization of the equations (2.52)–(2.57) in the strong form, and then derive it in subsection (ii) (see Hesthaven and Warburton (2008) for the difference between the strong and weak form DG schemes). This basic discretization does not define the edge–flux quantities, which differentiates different DG schemes. In (iii), we state the equations of our new HDG projection-method discretization. To start the derivation, in (iv), we provide the equations for the un-split method. In (v), we derive the element-local HDG equations and obtain preliminary HDG relations and flux equations. These HDG fluxes and stability parameter equations are then formally derived and justified in subsections §2.3.3 and §2.3.4.

(i) Discontinuous Galerkin spatial Discretization

The first step in the splitting scheme is to solve for the velocity–predictor (2.52)–(2.53). These equations are discretized as follows - derivations are in sub-section

(ii):

$$\begin{aligned}
\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1},\boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1},\boldsymbol{\Theta}\right)_K - \left\langle\hat{\bar{\mathbf{v}}}^{k+1}-\bar{\mathbf{v}}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} &= 0,\\
\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1},\boldsymbol{\theta}\right)_K + \left(\nabla p^k,\boldsymbol{\theta}\right)_K + \left\langle-(\hat{\bar{\mathbf{Q}}}^{k+1}-\bar{\mathbf{Q}}^{k+1})\cdot\hat{\mathbf{n}}+(\hat{p}^k-p^k)\hat{\mathbf{n}},\boldsymbol{\theta}\right\rangle_{\partial K} &= \left(\mathbf{F}^{k,k+1},\boldsymbol{\theta}\right)_K,
\end{aligned} \tag{2.58}$$

where we have used the $\hat{\bullet}$ notation to indicate that the solution on the edge is a combination of the solutions bordering that interface. Then, to find the projection of the predictor velocity onto the irrotational pressure space, we discretize (2.54)–(2.55) as:

$$\begin{aligned}
\left(\mathbf{q}_{\delta p}^{k+1},\boldsymbol{\theta}\right)_K - \left(\nabla\delta p^{k+1},\boldsymbol{\theta}\right)_K - \left\langle\widehat{\delta p}^{\,k+1}-\delta p^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} &= 0,\\
-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1},\theta\right)_K - \left\langle(\widehat{\mathbf{q}_{\delta p}}^{k+1}-\mathbf{q}_{\delta p}^{k+1})\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K} &= -\left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t},\theta\right)_K - \left\langle\frac{(\hat{\bar{\mathbf{v}}}_\star^{k+1}-\bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K}.
\end{aligned} \tag{2.59}$$

Finally, the velocity and pressure are corrected using the algebraic equations (2.56) and discretized version of (2.57)

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}, \tag{2.60}$$

$$\left(p^{k+1},\theta\right)_K = \left(p^k+\delta p^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla\cdot\bar{\mathbf{v}}^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left\langle(\hat{\bar{\mathbf{v}}}_\star^{k+1}-\bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}. \tag{2.61}$$

The boundary conditions for these equations are enforced through the flux quantities. For our new HDG discretization, they are given and derived next in subsections (iii) and (v).

(ii) Derivation of the generic DG discretization using the method of weighted residuals

Now we derive the basic DG discretization given in (2.58)–(2.61) using the method of weighted residuals (e.g. Chapra and Canale (2010)). We will derive the strong form simultaneously for sets of equations that are similar.

We begin with (2.52) and (2.54), multiplying these equations by tensor test-functions $\boldsymbol{\Theta}$ and vector test-functions $\boldsymbol{\theta}$, respectively, and then integrating over an

element K:

$$\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K = 0,$$
$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla\delta p^{k+1}, \boldsymbol{\theta}\right)_K = 0.$$

Integrate by parts the second term in both equations to recover the weak form:

$$\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K + \left(\bar{\mathbf{v}}^{k+1}, \nabla\cdot\boldsymbol{\Theta}\right)_K - \left\langle\hat{\bar{\mathbf{v}}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} = 0,$$
$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K + \left(\delta p^{k+1}, \nabla\cdot\boldsymbol{\theta}\right)_K - \left\langle\widehat{\delta p}^{\,k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = 0,$$

where we have used the $\hat{\bullet}$ notation to indicate that the solution on the edge is a combination of the solutions bordering that interface. Now, integrate by parts again, this time using the interior solution as the edge-value to recover the strong form:

$$\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left\langle\hat{\bar{\mathbf{v}}}^{k+1} - \bar{\mathbf{v}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} = 0, \tag{2.62}$$
$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla\delta p^{k+1}, \boldsymbol{\theta}\right)_K - \left\langle\widehat{\delta p}^{\,k+1} - \delta p^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = 0. \tag{2.63}$$

The strong form recovers the original equation, it highlights how DG methods penalize the jump in the solution across elements, and while it is mathematically the same as the weak form, we use it for implementation reason (see §3.3). This discretization is completed by the specification of the edge values, $\hat{\bar{\mathbf{v}}}^{k+1}$ and $\widehat{\delta p}^{\,k+1}$.

Proceeding as above, we discretize (2.53) and (2.55) by multiplying with test-functions $\boldsymbol{\theta}$ and θ, respectively, and then integrating over an element K:

$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\theta}\right)_K + \left(\nabla p^k, \boldsymbol{\theta}\right)_K = \left(\mathbf{F}^{k,k+1}, \boldsymbol{\theta}\right)_K,$$
$$-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1}, \theta\right)_K = -\left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \theta\right)_K.$$

As before we integrate by parts the second and third terms in the first equation, and

both terms in the second equation:

$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K + \left(\bar{\mathbf{Q}}^{k+1},\nabla\boldsymbol{\theta}\right)_K - \left(p^k,\nabla\theta\right)_K + \left\langle -\widehat{\bar{\mathbf{Q}}}^{k+1}\cdot\hat{\mathbf{n}} + \widehat{p}^k\hat{\mathbf{n}},\boldsymbol{\theta}\right\rangle_{\partial K} = \left(\mathbf{F}^{k,k+1},\boldsymbol{\theta}\right)_K,$$

$$\left(\mathbf{q}_{\delta p}^{k+1},\nabla\theta\right)_K - \left\langle\widehat{\mathbf{q}_{\delta p}}^{k+1}\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K} = \left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\nabla\theta\right)_K - \left\langle\frac{\widehat{\bar{\mathbf{v}}}_\star^{k+1}\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K},$$

where we have use $\widehat{\bar{\mathbf{v}}}_\star^{k+1}$ to differentiate the edge value for this equation from $\widehat{\bar{\mathbf{v}}}^{k+1}$ in (2.62), since these two need not be the same (i.e. there is no numerical consistency consideration, and DG methods allow some freedom in how flux quantities are chosen). Finally, integrate by parts a second time, using the interior solution on the edge

$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1},\boldsymbol{\theta}\right)_K + \left(\nabla p^k,\boldsymbol{\theta}\right)_K + \left\langle -(\widehat{\bar{\mathbf{Q}}}^{k+1} - \bar{\mathbf{Q}}^{k+1})\cdot\hat{\mathbf{n}} + (\widehat{p}^k - p^k)\hat{\mathbf{n}},\boldsymbol{\theta}\right\rangle_{\partial K} = \left(\mathbf{F}^{k,k+1},\boldsymbol{\theta}\right)_K,$$

$$-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1},\theta\right)_K - \left\langle(\widehat{\mathbf{q}_{\delta p}}^{k+1} - \mathbf{q}_{\delta p}^{k+1})\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K} = -\left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t},\theta\right)_K - \left\langle\frac{(\widehat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K}.$$

Again, this discretization is completed by the definition of the edge-terms.

(2.56) does not require any further modification, since it is a simple algebraic equation in terms of the unknown variables. For (2.57), we multiply by test–function θ, and integrate over an element K:

$$\left(p^{k+1},\theta\right)_K = \left(p^k + \delta p^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla\cdot\bar{\mathbf{v}}^{k+1},\theta\right)_K.$$

Integration by parts of the last term on the right–hand–side, twice (as before) gives

$$\left(p^{k+1},\theta\right)_K = \left(p^k + \delta p^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla\cdot\bar{\mathbf{v}}^{k+1},\theta\right)_K - \frac{1}{\mathrm{Re}}\left\langle(\widehat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}.$$

This completes the basic theoretical finite element DG discretization. We note that the discretized equations (2.60) and (2.61) can also be obtained by taking the difference between the DG-discretized equations of the full un-split system (not given) and the split system (2.58)-(2.59). This gives the same discretely-consistent element-space scheme as that given above. Such a derivation is completed for our HDG scheme in sub-section (v) and §2.3.3-§2.3.4. We also note that there is considerable effort required between the mathematical formulation and an actual implementation of the

algorithm. As such, specific details of our implementation are in §3. Next we provide our new HDG discretization.

(iii) Hybrid discontinuous Galerkin spatial Discretization: Equations

We state one of our main results, the new element–local set of equations, along with their global flux–conservation equations. All of the equations for our new scheme, including all fluxes to be derived later, are summarized in Fig. [2-4].

The element–local equations (with the HDG fluxes substituted) along with the global flux–conservation equations are obtained next. Note, we re–arrange the local equations such that locally–calculated quantities are on the left of the equal sign, while globally calculated (i.e. $\boldsymbol{\lambda}$'s) and known (or given) quantities are on the right. Also, note that the distinction between the usual DG and HDG lies in the definition of the numerical fluxes (see section v), where the $\boldsymbol{\lambda}$'s belong to the new HDG edge-space.

The element–local equations for $\bar{\mathbf{v}}^{k+1}$, which complete the DG discretization (2.58) are

$$\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K + \left\langle \bar{\mathbf{v}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} = \left\langle \bar{\boldsymbol{\lambda}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K}, \tag{2.64}$$

$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau\bar{\mathbf{v}}^{k+1}, \boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle \tau\bar{\boldsymbol{\lambda}}^{k+1}, \boldsymbol{\theta}\right\rangle_{\partial K} - \left(\nabla p^k, \boldsymbol{\theta}\right)_K + \left(\mathbf{F}^{k,k+1}, \boldsymbol{\theta}\right)_K, \tag{2.65}$$

where $\tau = 1$ is the HDG stability parameter (see §2.3.4). The global flux–conservation equations for $\bar{\boldsymbol{\lambda}}^{k+1}$ are

$$\left\langle \left[\!\left[\widehat{\bar{\mathbf{Q}}}^{k+1}\cdot\hat{\mathbf{n}}\right]\!\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon = \left\langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon$$

$$\Rightarrow \left\langle \left[\!\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}} - \tau\left(\bar{\mathbf{v}}^{k+1} - \bar{\boldsymbol{\lambda}}^{k+1}\right)\right]\!\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon = \left\langle \left[\!\left[p^k\hat{\mathbf{n}}\right]\!\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon + \left\langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon, \tag{2.66}$$

$$\bar{\boldsymbol{\lambda}}|_{\varepsilon_D^\partial}^{k+1} = \mathbf{g}_D \tag{2.67}$$

where $\mathbf{g}_D$ and $\mathbf{g}_N$ are the values of Dirichlet and Neumann boundary conditions for the momentum equations, respectively.

The element–local equations for δp^{k+1}, which complete the DG discretization of

1. Velocity predictor (momentum equations)

Element-Local equations:

$$\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K + \left\langle\bar{\mathbf{v}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} = \left\langle\bar{\boldsymbol{\lambda}}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K}$$

$$\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle\tau\bar{\mathbf{v}}^{k+1}, \boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\tau\bar{\boldsymbol{\lambda}}^{k+1}, \boldsymbol{\theta}\right\rangle_{\partial K} - \left(\nabla p^k, \boldsymbol{\theta}\right)_K + \left(\mathbf{F}^{k,k+1}, \boldsymbol{\theta}\right)_K$$

Edge-space global flux conservation equations:

$$\left\langle\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}} - \tau\left(\bar{\mathbf{v}}^{k+1} - \bar{\boldsymbol{\lambda}}^{k+1}\right)\right]\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon = \left\langle\left[\left[p^k\hat{\mathbf{n}}\right]\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon + \left\langle\mathbf{g}_N, \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon$$

$$\bar{\boldsymbol{\lambda}}\big|_{\varepsilon_D^\partial}^{k+1} = \mathbf{g}_D$$

Flux definitions:

$$\widehat{\bar{\mathbf{v}}}^{k+1} = \begin{cases} \mathrm{P}\mathbf{g}_D, & \text{on } \varepsilon^\partial \\ \bar{\boldsymbol{\lambda}}^{k+1}, & \text{on } \varepsilon^\circ \end{cases}$$

$$\widehat{\bar{\mathbf{Q}}}^{k+1} - \widehat{p}^k\mathbf{I} = \bar{\mathbf{Q}}^{k+1} - p^k\mathbf{I} - \tau\left(\bar{\mathbf{v}}^{k+1} - \widehat{\bar{\mathbf{v}}}^{k+1}\right)\hat{\mathbf{n}}$$

2. Pressure corrector (to enforce continuity)

Element-Local equations:

$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla\delta p^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle\delta p^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\lambda_{\delta p}^{k+1}, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K}$$

$$-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1}, \theta\right)_K + \left\langle\tau_p\delta p^{k+1}, \theta\right\rangle_{\partial K} = \left\langle\tau_p\lambda_{\delta p}^{k+1}, \theta\right\rangle_{\partial K} - \left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \theta\right)_K - \left\langle\frac{(\bar{\boldsymbol{\lambda}}^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t}, \theta\right\rangle_{\partial K}$$

Edge-space global flux conservation equations:

$$\left\langle\left[\left[\mathbf{q}_{\delta p}{}^{k+1}\cdot\hat{\mathbf{n}} - \tau_p\left(\delta p^{k+1} - \lambda_{\delta p}^{k+1}\right)\right]\right], \theta_\varepsilon\right\rangle_\varepsilon = \left\langle g_{N_p}, \theta_\varepsilon\right\rangle_\varepsilon$$

$$\lambda_{\delta p}\big|_{\varepsilon^\partial}^{k+1} = g_{D_p}$$

Flux definitions:

$$\widehat{\delta p}^{k+1} = \begin{cases} \mathrm{P}g_{D_p}, & \text{on } \varepsilon^\partial \\ \lambda_{\delta p}^{k+1}, & \text{on } \varepsilon^\circ \end{cases}$$

$$\widehat{\mathbf{q}_{\delta p}}^{k+1} = \mathbf{q}_{\delta p}{}^{k+1} - \tau_p\left(\delta p^{k+1} - \widehat{\delta p}^{k+1}\right)\hat{\mathbf{n}}$$

$$\widehat{\bar{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}$$

3. Velocity and pressure corrections

Element-Local correction:

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1},$$

$$\left(p^{k+1}, \theta\right)_K = \left(p^k + \delta p^{k+1}, \theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla\cdot\bar{\mathbf{v}}^{k+1}, \theta\right)_K - \frac{1}{\mathrm{Re}}\left\langle(\widehat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}, \theta\right\rangle_{\partial K}$$

Edge-space correction: $\boldsymbol{\lambda}^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}{}^{k+1} + a\Delta t\tau_p\left(\delta p^{k+1} - \widehat{\delta p}^{k+1}\right)\hat{\mathbf{n}}$

Consistent HDG stability parameter: $\tau_p = \frac{1}{a\Delta t\tau}$

Implicit flux definition:

$$\boldsymbol{\lambda}^{k+1} = \left\{\left\{\mathbf{v}^{k+1}\right\}\right\} - \frac{1}{2\tau}\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^k\hat{\mathbf{n}}\right]\right] + a\Delta t\frac{\tau_p}{2}\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right]$$

Figure 2-4: New HDG and projection method scheme. Plan boxes denote the main equations, while dashed boxes give flux definitions.

(2.59) are

$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla \delta p^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \delta p^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K} = \left\langle \lambda_{\delta p}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K}, \tag{2.68}$$

$$-\left(\nabla \cdot \mathbf{q}_{\delta p}^{k+1}, \theta\right)_K + \left\langle \tau_p \delta p^{k+1}, \theta \right\rangle_{\partial K} = \left\langle \tau_p \lambda_{\delta p}^{k+1}, \theta \right\rangle_{\partial K} - \left(\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \theta\right)_K - \left\langle \frac{(\bar{\boldsymbol{\lambda}}^{k+1} - \bar{\mathbf{v}}^{k+1}) \cdot \hat{\mathbf{n}}}{a\Delta t}, \theta \right\rangle_{\partial K}, \tag{2.69}$$

where we have used $\widehat{\bar{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}$ as defined by (2.83), and $\tau_p = \frac{1}{a\Delta t \tau}$ is the HDG stability parameter for the pressure–correction (see §2.3.4). The global flux–conservation equation for $\lambda_{\delta p}^{k+1}$ is

$$\left\langle \left[\!\left[\widehat{\mathbf{q}_{\delta p}}^{k+1} \cdot \hat{\mathbf{n}} \right]\!\right], \theta_\varepsilon \right\rangle_\varepsilon = \left\langle g_{N_p}, \theta_\varepsilon \right\rangle_\varepsilon$$

$$\Rightarrow \left\langle \left[\!\left[\mathbf{q}_{\delta p}{}^{k+1} \cdot \hat{\mathbf{n}} - \tau_p \left(\delta p^{k+1} - \lambda_{\delta p}^{k+1}\right) \right]\!\right], \theta_\varepsilon \right\rangle_\varepsilon = \left\langle g_{N_p}, \theta_\varepsilon \right\rangle_\varepsilon, \tag{2.70}$$

$$\lambda_{\delta p}|_{\varepsilon_D^\partial}^{k+1} = g_{D_p} \tag{2.71}$$

where g_{D_P} and g_{N_P} are the values of Dirichlet and Neumann boundary conditions for the pressure–correction, respectively (and these are often zero Neumann, §2.2.3).

The element-local DG velocity and pressure correction equations (2.60)-(2.61) remain unchanged for HDG,

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}^{k+1},$$

$$\left(p^{k+1}, \theta\right)_K = \left(p^k + \delta p^{k+1}, \theta\right)_K - \frac{1}{\mathrm{Re}}\left(\nabla \cdot \bar{\mathbf{v}}^{k+1}, \theta\right)_K - \frac{1}{\mathrm{Re}}\left\langle (\widehat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1}) \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K},$$

but a new correction, $\widehat{\mathbf{v}}_{\mathrm{cor}}^{k+1} = -a\Delta t \mathbf{q}_{\delta p}{}^{k+1} + a\Delta t \tau_p \left(\delta p - \widehat{\delta p}^{k+1}\right) \hat{\mathbf{n}}$, on the velocity edge–space is now required

$$\boldsymbol{\lambda}^{k+1} = \widehat{\bar{\mathbf{v}}}_\star^{k+1} + \widehat{\mathbf{v}}_{\mathrm{cor}}^{k+1} \tag{2.72}$$

$$= \bar{\boldsymbol{\lambda}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}{}^{k+1} + a\Delta t \tau_p \left(\delta p^{k+1} - \widehat{\delta p}^{k+1}\right) \hat{\mathbf{n}}, \tag{2.73}$$

where we have used $\widehat{\bar{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}$ as defined by (2.83). We note that a correction of

the pressure on the edge-space is not needed in the present scheme: only needed are local pressure corrections in (2.65) and jump terms corrections in (2.66). All of these HDG fluxes are provided and derived in sub-section (v).

(iv) Hybrid discontinuous Galerkin spatial Discretization: Un-split equations

As done in the explanation of the time-discrete projection method (§2.2.1), we now state the element–local and globally–coupled HDG discretization for the un–split equations. These element-local un-split equations with HDG fluxes substituted can be used to derive the velocity and pressure corrections for the split (projection method) equations. The un-split equations are:

$$\begin{aligned}
\left((\mathrm{Re})\mathbf{Q}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla \mathbf{v}^{k+1}, \boldsymbol{\Theta}\right)_K + \left\langle \mathbf{v}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} &= \left\langle \boldsymbol{\lambda}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K}, \\
\left(\frac{\mathbf{v}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla \cdot \mathbf{Q}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau \mathbf{v}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} + \left(\nabla p^{k+1}, \boldsymbol{\theta}\right)_K &= \left\langle \tau \boldsymbol{\lambda}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} + \left(\mathbf{F}^{k,k+1}, \boldsymbol{\theta}\right)_K, \\
\left(\nabla \cdot \mathbf{v}^{k+1}, \theta\right)_K - \left\langle \mathbf{v}^{k+1} \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K} &= -\left\langle \boldsymbol{\lambda}^{k+1} \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K}, \\
\left(p^{k+1}, \frac{1}{|K|}\right)_K &= |p|^{k+1}.
\end{aligned} \tag{2.74}$$

Note that these element–local Stokes equations are solvable once the velocity boundary conditions and average pressure are specified. The globally coupled equations for $\boldsymbol{\lambda}$ and the average pressure $|p| = \frac{1}{|K|}\int_K p \mathrm{d}K$ on the element (with volume $|K| = \int_K \mathrm{d}K$) are:

$$\begin{aligned}
\left\langle \left[\left[-\mathbf{Q}^{k+1} \cdot \hat{\mathbf{n}} + p^{k+1}\hat{\mathbf{n}} + \tau\left(\mathbf{v}^{k+1} - \boldsymbol{\lambda}^{k+1}\right)\right]\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \left\langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon, \\
\left\langle \boldsymbol{\lambda}^{k+1} \cdot \hat{\mathbf{n}}, \frac{1}{|\partial K|} \right\rangle_{\partial \mathcal{T}_h} &= 0, \\
\boldsymbol{\lambda}^{k+1}|_{\varepsilon_D^\partial} &= \mathbf{g}_D.
\end{aligned} \tag{2.75}$$

(v) Hybrid discontinuous Galerkin spatial Discretization: Derivations

Next, we first provide a brief derivation of the element-space HDG equations, then, give a few notes about the form of our flux definitions using HDG methods (Nguyen et al., 2009a, Cockburn et al., 2009a). We then obtain the new HDG fluxes

that are used to complete the discretization of (2.58)–(2.61).

Element-space HDG discretization: We start by implicitly deriving the element-local equations (2.64)–(2.65), (2.68)–(2.69) and especially the corrections (2.60)–(2.61). The derivation is implicit because these equations have already been derived from the continuous projection-method equations and their discretization: this is because they are equivalent to the DG-discretized equations of subsection (ii), up to the edge-space λ parameters. Instead of starting from the continuous projection-method equations, another formal derivation at the discrete level starts directly from the element-space discrete equations, so as to ensure that the corrections (2.60)–(2.61) are consistent at the discrete-level proper. To do so, one takes the difference between the un-split HDG-discretized equations (2.74) and the corresponding split HDG-discretized equations (2.64)–(2.65) and (2.68)–(2.69). The result is again the same as in subsection (ii), i.e. the HDG correction equations (2.60)–(2.61). This procedure of taking the difference between the un-split and split HDG-discretized equations is also employed in §2.3.4 to derive consistent edge-space corrections and stability parameters.

Edge-space HDG discretization and HDG fluxes: To justify (2.73), we first note that $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$ in (2.72) is a dummy variable to be determined and that this relation (2.72) is an algebraic equation correct both on the element–space and edge–space. The flux chosen in (2.73) is discretely consistent with the discretized divergence equation, and the consistency is proven in §2.3.3. Specifically, (2.73) was obtained by evaluating (2.60) on the edge-space, using the definition of $\widehat{\mathbf{v}}_{\text{cor}}^{k+1} = -a\Delta t\widehat{\mathbf{q}_{\delta p}}^{k+1}$ and assuming that (2.83) was used for $\widehat{\mathbf{v}}_{\star}^{k+1}$. Note that we do not need an equation to update the value of λ_p on the HDG edge–space, since this quantity is not used in the next time–step. We only need the pressure gradient on the element interiors (2.61). The formal derivation of (2.73) is provided in §2.3.3.

To complete the DG and HDG discretizations, we needed to define the fluxes and edge terms, $\widehat{\mathbf{v}}^{k+1}$, $\widehat{\mathbf{v}}_{\star}^{k+1}$, $\widehat{\delta p}^{k+1}$, $\widehat{\mathbf{Q}}^{k+1}$, $\widehat{p}^{k}$, and $\widehat{\mathbf{q}_{\delta p}}^{k+1}$. The diffusive fluxes for

discontinuous Galerkin schemes are normally reported in the form

$$\begin{aligned} \widehat{\mathbf{a}} &= \{\{\mathbf{a}\}\} - C_{11} [\![d\hat{\mathbf{n}}]\!] + \mathbf{C}_{12} [\![\mathbf{a} \cdot \hat{\mathbf{n}}]\!] , \\ \hat{d} &= \{\{d\}\} - \mathbf{C}_{12} \cdot [\![d\hat{\mathbf{n}}]\!] - C_{22} [\![\mathbf{a} \cdot \hat{\mathbf{n}}]\!] , \end{aligned} \tag{2.76}$$

for vectors and scalars $\mathbf{a}$ and d, respectively. However, the HDG flux definitions are more easily understood in a different form.

To understand how HDG fluxes are specified, it is important to grasp the underlying premise of HDG methods. The premise is that one can solve the set of equations (2.58) or (2.59) locally on an element as long as all the necessary boundary and initial conditions are specified. While the initial conditions are known, the boundary conditions for the implicit variables are not known, and we desire an equation to solve for these boundary conditions. To find this equation, the solution on the interior of an element is parameterized in terms of new variables which represent the boundary conditions for that element. For our equations, we need two new variables, $\bar{\boldsymbol{\lambda}}$ for (2.58) and $\lambda_{\delta p}$ for (2.59), which live in the same spaces as $\boldsymbol{\theta}_\varepsilon$ and θ_ε, respectively. That is, $\bar{\boldsymbol{\lambda}}$ and $\lambda_{\delta p}$ only exist on the new hybrid discontinuous edge–space, and do not have a value inside the element (i.e. no interior support). Finally, the globally–coupled equation for the boundary conditions are found by enforcing the conservation of fluxes across elements (more details about the implementation can be found in §3.4). As such, we will define our flux quantities in terms of the new HDG variables $\bar{\boldsymbol{\lambda}}$ and λ_p.

Additionally, we note that the original Stokes system only required knowledge of the velocity initial conditions, average pressure, and velocity–boundary conditions to be solvable. However, for the time–split system using projection methods we need to specify the initial velocity and pressure, and boundary conditions for the velocity and pressure–correction. While this may appear to be an over–specification due to the additional boundary conditions for the pressure–correction, the boundary conditions for velocity and pressure are intimately related, and require careful treatment to be numerically consistent. The proper specification of these HDG fluxes using Projection methods has not been previously addressed.

Each of the two main DG-discretized system of governing equations ((2.58) and (2.59), and their state variables) of the projection method has in principle a $\boldsymbol{\lambda}$ variable. These $\boldsymbol{\lambda}$'s are: $\bar{\boldsymbol{\lambda}}$ for $\bar{\mathbf{v}}$ and $\lambda_{\delta p}$ for δp. As a result, by algebraic consistency, one can also introduce the corrections $\boldsymbol{\lambda}$ for $\mathbf{v}$ and λ_p for p. These two $\boldsymbol{\lambda}$'s for the corrections are of course related to the other $\boldsymbol{\lambda}$'s and to corresponding fluxes since the two correcting equations are algebraic equations.

Now, our flux definitions for $\widehat{\bar{\mathbf{v}}}^{k+1}$ in (2.58) at time $k+1$ are:

$$\widehat{\bar{\mathbf{v}}}^{k+1} = \begin{cases} \mathsf{P}\mathbf{g}_D, & \text{on } \varepsilon^{\partial} \\ \bar{\boldsymbol{\lambda}}^{k+1}, & \text{on } \varepsilon^{\circ} \end{cases} \tag{2.77}$$

$$\widehat{\bar{\mathbf{Q}}}^{k+1} - \widehat{p}^{k}\mathbf{I} = \bar{\mathbf{Q}}^{k+1} - p^{k}\mathbf{I} - \tau\left(\bar{\mathbf{v}}^{k+1} - \widehat{\bar{\mathbf{v}}}^{k+1}\right)\hat{\mathbf{n}}, \tag{2.78}$$

where P is the L^2 projection of the boundary condition $\mathbf{g}_D$ into the same space as $\boldsymbol{\theta}_{\varepsilon}$, and $\mathbf{I}$ is the $d \times d$ identity tensor, and $\left(\bar{\mathbf{v}}^{k+1} - \widehat{\bar{\mathbf{v}}}^{k+1}\right)\hat{\mathbf{n}}$ is a $d \times d$ tensor. The appearance of an explicit variable p^k in (2.78) is justified in §2.3.4.

Similarly, the fluxes for the pressure–correction equation, (2.59), are expressed as:

$$\widehat{\delta p}^{k+1} = \begin{cases} \mathsf{P}g_{D_p}, & \text{on } \varepsilon^{\partial} \\ \lambda_{\delta p}^{k+1}, & \text{on } \varepsilon^{\circ} \end{cases} \tag{2.79}$$

$$\widehat{\mathbf{q}_{\delta p}}^{k+1} = \mathbf{q}_{\delta p}{}^{k+1} - \tau_p\left(\delta p^{k+1} - \widehat{\delta p}^{k+1}\right)\hat{\mathbf{n}}. \tag{2.80}$$

However, the choice for the velocity fluxes $\widehat{\bar{\mathbf{v}}}_{\star}^{k+1}$ in the continuity (divergence) term on the right–hand–side of (2.59) depends on the choice of the advective flux. For example, a central, upwind, or hybrid flux choice in the interior would yield

$$\widehat{\bar{\mathbf{v}}}_{\star}^{k+1} = \left\{\left\{\bar{\mathbf{v}}^{k+1}\right\}\right\}, \tag{2.81}$$

$$\widehat{\bar{\mathbf{v}}}_{\star}^{k+1} = \left\{\left\{\bar{\mathbf{v}}^{k+1}\right\}\right\} + \frac{1}{2}\text{sign}(\bar{\mathbf{v}}^{k+1}\cdot\hat{\mathbf{n}})\left[\!\left[\bar{\mathbf{v}}^{k+1}\hat{\mathbf{n}}\right]\!\right]\cdot\hat{\mathbf{n}}, \tag{2.82}$$

$$\widehat{\bar{\mathbf{v}}}_{\star}^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}, \tag{2.83}$$

respectively. In our experience, (2.83) gives the most accurate results, as such we will

use this flux.

Finally, our intent here was to summarize the order for solving the various systems of equations, but the actual solution procedure is more complex, and the element–local equations have to be solved multiple times (see §3.4).

2.3.3 Consistency of the velocity correction on the HDG edge-space: formal justification

In this section we show that the edge–space correction of the predictor velocity, (2.73), is consistent with the discrete divergence equation and the pressure–correction equations (2.68)-(2.69). The element-space derivation was already completed in §2.3.2-(v). Importantly, the edge correction is a gradient of δp on the edge, which is only continuous in the normal and not tangential direction: hence, even though $\widehat{\mathbf{v}}_\star^{k+1}$ was unique on the edge, the gradient of δp is only unique in the normal direction: this has important implications which we will discuss.

To prove the consistency of the flux, we will use (2.69) and (2.72) (which is valid on both the element and edge spaces). This will allow us to recover the discrete divergence equation for $\mathbf{v}^{k+1}$, which we want to satisfy numerically. By setting the discrete divergence to zero, we then show that the flux $\widehat{\mathbf{v}}_{\text{cor}}^{k+1} = -a\Delta t\widehat{\mathbf{q}_{\delta p}}^{k+1}$ is consistent.

We begin by re-arranging (2.69), where the goal is to combine terms to make the velocity correction equation (2.60) appear on the element–space and thus find the consistent edge-space flux condition for $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$ from that of $\mathbf{q}_{\delta p}$ (since $\mathbf{v}_{\text{cor}}^{k+1} = -a\Delta t{\mathbf{q}_{\delta p}}^{k+1}$ on the element–interior):

$$-\left(\nabla\cdot\mathbf{q}_{\delta p}^{k+1},\theta\right)_K+\left\langle\tau_p\delta p^{k+1},\theta\right\rangle_{\partial K}-\left\langle\tau_p{\lambda_{\delta p}}^{k+1},\theta\right\rangle_{\partial K}+\left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t},\theta\right)_K+\left\langle\frac{(\widehat{\mathbf{v}}_\star^{k+1}-\bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K}=0.$$

Multiplying by $a\Delta t$ and assembling terms we obtain,

$$\Rightarrow \left(\nabla\cdot\bar{\mathbf{v}}^{k+1} - a\Delta t\nabla\cdot\mathbf{q}_{\delta p}^{k+1},\theta\right)_K + \left\langle(\widehat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}),\theta\right\rangle_{\partial K} = 0,$$

$$\Rightarrow \left(\nabla\cdot(\bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}),\theta\right)_K + \left\langle(\widehat{\bar{\mathbf{v}}}_\star^{k+1} - (\bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}))\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}$$

$$\left\langle -a\Delta t\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}),\theta\right\rangle_{\partial K} = 0,$$

where we have added and subtracted $a\Delta t\mathbf{q}_{\delta p}^{k+1}$ in the edge terms. Now substitute for $\mathbf{v}^{k+1}$ and $\widehat{\mathbf{v}}^{k+1} = \widehat{\bar{\mathbf{v}}}_\star^{k+1} + \widehat{\mathbf{v}}_{\text{cor}}^{k+1}$

$$\left(\nabla\cdot\mathbf{v}^{k+1},\theta\right)_K + \left\langle(\widehat{\mathbf{v}}^{k+1} - \widehat{\mathbf{v}}_{\text{cor}}^{k+1})\cdot\hat{\mathbf{n}} - \mathbf{v}^{k+1}\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}$$

$$+\left\langle -a\Delta t\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}),\theta\right\rangle_{\partial K} = 0,$$

$$\Rightarrow \left(\nabla\cdot\mathbf{v}^{k+1},\theta\right)_K + \left\langle\widehat{\mathbf{v}}^{k+1}\cdot\hat{\mathbf{n}} - \mathbf{v}^{k+1}\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K}$$

$$+\left\langle -\widehat{\mathbf{v}}_{\text{cor}}^{k+1}\cdot\hat{\mathbf{n}} - a\Delta t\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}),\theta\right\rangle_{\partial K} = 0.$$

Now, set $\left(\nabla\cdot\mathbf{v}^{k+1},\theta\right)_K + \left\langle\widehat{\mathbf{v}}^{k+1}\cdot\hat{\mathbf{n}} - \mathbf{v}^{k+1}\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K} = 0$, since this is the discrete divergence equation. Finally we can solve for the edge-correction $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$ by projection on the normal:

$$\left\langle\widehat{\mathbf{v}}_{\text{cor}}^{k+1}\cdot\hat{\mathbf{n}},\theta\right\rangle_{\partial K} = \left\langle -a\Delta t\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}),\theta\right\rangle_{\partial K},$$

$$\Rightarrow \left(\widehat{\mathbf{v}}_{\text{cor}}^{k+1}\right)\cdot\hat{\mathbf{n}} = \left(-a\Delta t\mathbf{q}_{\delta p}^{k+1} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1})\hat{\mathbf{n}}\right)\cdot\hat{\mathbf{n}}.$$

Dropping the normal

$$\Rightarrow \widehat{\mathbf{v}}_{\text{cor}}^{k+1} = -a\Delta t\mathbf{q}_{\delta p}^{k+1} + a\Delta t\tau_p(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1})\hat{\mathbf{n}},$$

we recover (2.73).

A few remarks.

1. From (2.73) (and the above justification), the pressure correction leads to a continuous change of velocity in the normal direction, but not in the tangential one. This change in the normal direction is equal to $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}\cdot\hat{\mathbf{n}} = -a\Delta t{\mathbf{q}_{\delta p}}^{k+1}\cdot$

$\hat{\mathbf{n}} - a\Delta t\tau_p \left(\delta p^{k+1} - \lambda_{\delta p}^{k+1}\right)$, and is continuous because we solved (2.70). However, the change in the tangential direction is not expected to be continuous. Hence, the correction velocity is therefore discontinuous across the edge.

2. For the advective flux across elements, remark 1 has no consequence, since we only require the normal component of the velocity on the hybrid discontinuous edge-space. As such, we can use

$$\begin{aligned}\widehat{\mathbf{v}}_{\text{cor}}^{k+1}/a\Delta t = c\left\{-\mathbf{q}_{\delta p}{}^{+,k+1} + \tau_p\left(\delta p^{+,k+1} - \lambda_{\delta p}^{k+1}\right)\hat{\mathbf{n}}^{+}\right\} \\ + (1-c)\left\{-\mathbf{q}_{\delta p}{}^{-,k+1} + \tau_p\left(\delta p^{-,k+1} - \lambda_{\delta p}^{k+1}\right)\hat{\mathbf{n}}^{+}\right\}\end{aligned}$$

for arbitrary constant $0 \leq c \leq 1$.

3. For some implementations of the rotational correction term (see Appendix B), remark 1 has consequences, and require additional consideration.

Next we discuss the appropriate size for the HDG stability parameters.

2.3.4 Derivation of consistent HDG stability parameter for the pressure–correction

As of now, we have not given guidance on choosing the magnitude of the stability parameters τ and τ_p. Fortunately, the effect of varying the magnitude of τ has been well-studied (Nguyen et al., 2009a, Cockburn et al., 2009a,b), and from these studies $\tau = 1$ gives the most accurate fluxes. What remains is finding a consistent value for τ_p. In this section, we also justify the explicit pressure–flux present in (2.78).

To find τ_p's consistent value, we compare the fluxes of the split equations to that of the un–split equations. We first solve for the un–split $\boldsymbol{\lambda}$ in terms of element–local quantities. To compare the un–split fluxes to the split fluxes, we also have to express the edge–space variables $\bar{\boldsymbol{\lambda}}$ and $\lambda_{\delta p}$ in terms of element–local quantities, then form the final edge–space velocity using (2.73).

To solve for $\boldsymbol{\lambda}^{k+1}$ in the interior of the domain (note $\mathbf{g}_N = 0$ on the interior), we use the first equation in (2.75), then we expand the "jump" operator in terms of

element–local quantities on either side of the edge, and finally we recombine terms using the "jump" and "mean" operators:

$$
\begin{aligned}
0 &= \left[\!\left[-\mathbf{Q}^{k+1}\cdot\hat{\mathbf{n}} + p^{k+1}\hat{\mathbf{n}} + \tau\left(\mathbf{v}^{k+1} - \boldsymbol{\lambda}^{k+1}\right)\right]\!\right] \\
0 &= -\mathbf{Q}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} + \mathbf{Q}^{-,k+1}\cdot\hat{\mathbf{n}}^{+} + p^{+,k+1}\hat{\mathbf{n}}^{+} - p^{-,k+1}\hat{\mathbf{n}}^{+} + \tau(\mathbf{v}^{+,k+1} + \mathbf{v}^{-,k+1} - 2\boldsymbol{\lambda}^{k+1}) \\
\Rightarrow \boldsymbol{\lambda}^{k+1} &= \{\{\mathbf{v}^{k+1}\}\} - \frac{1}{2\tau}\left[\!\left[\mathbf{Q}^{k+1}\cdot\hat{\mathbf{n}}\right]\!\right] + \frac{1}{2\tau}\left[\!\left[p^{k+1}\hat{\mathbf{n}}\right]\!\right].
\end{aligned}
\tag{2.84}
$$

Now, we want to compare this to the $\boldsymbol{\lambda}^{k+1}$ obtained from the split equations. From the flux–conservation equation for $\bar{\boldsymbol{\lambda}}^{k+1}$, (2.66) (that is, using our flux definition (2.78), which includes the explicit pressure contribution), we proceed similarly

$$
\begin{aligned}
0 &= \left[\!\left[-\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}} + p^{k}\hat{\mathbf{n}} + \tau\left(\bar{\mathbf{v}}^{k+1} - \bar{\boldsymbol{\lambda}}^{k+1}\right)\right]\!\right] \\
0 &= -\bar{\mathbf{Q}}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} + \bar{\mathbf{Q}}^{-,k+1}\cdot\hat{\mathbf{n}}^{+} + p^{+,k}\hat{\mathbf{n}}^{+} - p^{-,k}\hat{\mathbf{n}}^{+} + \tau(\bar{\mathbf{v}}^{+,k+1} + \bar{\mathbf{v}}^{-,k+1} - 2\bar{\boldsymbol{\lambda}}^{k+1}) \\
\Rightarrow \bar{\boldsymbol{\lambda}}^{k+1} &= \{\{\bar{\mathbf{v}}^{k+1}\}\} - \frac{1}{2\tau}\left[\!\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\!\right] + \frac{1}{2\tau}\left[\!\left[p^{k}\hat{\mathbf{n}}\right]\!\right].
\end{aligned}
\tag{2.85}
$$

Note that the time–level of the jump in the pressure is different between the split flux (2.85) and the un–split flux (2.84) . To correct the velocity, we need to know the form of the flux $\widehat{\mathbf{q}_{\delta p}}^{k+1}$ (2.80). Starting from (2.70) we can find the form for $\lambda_{\delta p}^{k+1}$ (similar to before)

$$
\begin{aligned}
0 &= \left[\!\left[-{\mathbf{q}_{\delta p}}^{k+1}\cdot\hat{\mathbf{n}} + \tau_p\left(\delta p^{k+1} - {\lambda_{\delta p}}^{k+1}\right)\right]\!\right] \\
0 &= -\mathbf{q}_{\delta p}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} + \mathbf{q}_{\delta p}^{-,k+1}\cdot\hat{\mathbf{n}}^{+} + \tau_p(\delta p^{+,k+1} + \delta p^{-,k+1} - 2\lambda_{\delta p}^{k+1}) \\
\Rightarrow \lambda_{\delta p}^{k+1} &= \{\{\delta p^{k+1}\}\} - \frac{1}{2\tau_p}\left[\!\left[\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}}\right]\!\right].
\end{aligned}
\tag{2.86}
$$

Now, substituting (2.86) into (2.80) we find:

$$
\begin{aligned}
\widehat{\mathbf{q}_{\delta p}}^{k+1}\cdot\hat{\mathbf{n}}^{+} &= {\mathbf{q}_{\delta p}}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} - \tau_p\left(\delta p^{+,k+1} - \lambda_{\delta p}^{k+1}\right) \\
&= {\mathbf{q}_{\delta p}}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} - \tau_p\left(\delta p^{+,k+1} + \{\{\delta p^{k+1}\}\} - \frac{1}{2\tau}\left[\!\left[\mathbf{q}_{\delta p}^{k+1}\cdot\hat{\mathbf{n}}\right]\!\right]\right) \\
&= {\mathbf{q}_{\delta p}}^{+,k+1}\cdot\hat{\mathbf{n}}^{+} - \tau_p\left(\delta p^{+,k+1} - \frac{\delta p^{+,k+1} + \delta p^{-,k+1}}{2} - \frac{1}{2\tau_p}(\mathbf{q}_{\delta p}^{+,k+1} - \mathbf{q}_{\delta p}^{-k+1})\cdot\hat{\mathbf{n}}^{+}\right) \\
&= \{\{{\mathbf{q}_{\delta p}}^{k+1}\}\}\cdot\hat{\mathbf{n}}^{+} + \frac{\tau_p}{2}\left[\!\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\!\right]\cdot\hat{\mathbf{n}}^{+}
\end{aligned}
\tag{2.87}
$$

Finally, we construct the final edge–velocity for the split equations by substituting for (2.85) and (2.87) into (2.73):

$$\begin{aligned}
\boldsymbol{\lambda}^{k+1} &= \bar{\boldsymbol{\lambda}}^{k+1} - a\Delta t\widehat{\mathbf{q}_{\delta p}}^{k+1} \\
&= \{\{\bar{\mathbf{v}}^{k+1}\}\} - \frac{1}{2\tau}\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^k\hat{\mathbf{n}}\right]\right] - a\Delta t\left\{\left\{\mathbf{q}_{\delta p}^{k+1}\right\}\right\} + a\Delta t\frac{\tau_p}{2}\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right], \\
&= \{\{\bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}\}\} - \frac{1}{2\tau}\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^k\hat{\mathbf{n}}\right]\right] + a\Delta t\frac{\tau_p}{2}\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right], \\
&= \{\{\mathbf{v}^{k+1}\}\} - \frac{1}{2\tau}\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^k\hat{\mathbf{n}}\right]\right] + a\Delta t\frac{\tau_p}{2}\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right]. \qquad (2.88)
\end{aligned}$$

Equating (2.84) and (2.88) we have

$$-\frac{1}{2\tau}\left[\left[\bar{\mathbf{Q}}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^k\hat{\mathbf{n}}\right]\right] + \frac{a\Delta t}{2}\tau_p\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right] = -\frac{1}{2\tau}\left[\left[\mathbf{Q}^{k+1}\cdot\hat{\mathbf{n}}\right]\right] + \frac{1}{2\tau}\left[\left[p^{k+1}\hat{\mathbf{n}}\right]\right]. \qquad (2.89)$$

Note that p^{k+1} in (2.89) is the un-split pressure. If we multiply both sides by 2 and re-arrange we have

$$a\Delta t\tau_p\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right] = \frac{1}{\tau}\left[\left[-(\mathbf{Q}^{k+1} - \bar{\mathbf{Q}}^{k+1})\cdot\hat{\mathbf{n}} + (p^{k+1} - p^k)\hat{\mathbf{n}}\right]\right],$$

then if we substitute for $p^{k+1} - p^k$ from (2.57) and re-arrange we have

$$\begin{aligned}
a\Delta t\tau_p\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right] &= \frac{1}{\tau}\left[\left[-(\mathbf{Q}^{k+1} - \bar{\mathbf{Q}}^{k+1})\cdot\hat{\mathbf{n}} + \left(\delta p^{k+1} - \frac{1}{\mathrm{Re}}\nabla\cdot\bar{\mathbf{v}}^{k+1}\right)\hat{\mathbf{n}}\right]\right], \\
\Rightarrow \tau_p &= \frac{1}{a\Delta t\tau}\frac{\left[\left[\delta p^{k+1}\hat{\mathbf{n}}\right]\right](1 + [\![\epsilon\hat{\mathbf{n}}]\!])}{[\![\delta p^{k+1}\hat{\mathbf{n}}]\!]}, \\
&= \frac{1 + [\![\epsilon\hat{\mathbf{n}}]\!]}{a\Delta t\tau}, \qquad (2.90)
\end{aligned}$$

where the value of $[\![\epsilon\hat{\mathbf{n}}]\!]$ is

$$\begin{aligned}
[\![\epsilon\hat{\mathbf{n}}]\!] &= \left[\left[\frac{1}{\delta p}(\bar{\mathbf{Q}} - \mathbf{Q})\cdot\hat{\mathbf{n}} - \frac{1}{\delta p}\frac{1}{\mathrm{Re}}(\nabla\cdot\bar{\mathbf{v}})\hat{\mathbf{n}}\right]\right] \\
&\sim \mathcal{O}\left(\frac{\Delta t^r\Delta x^{p+1}}{\mathrm{Re}}\right),
\end{aligned}$$

where $r \geq 0$ depends on the accuracy of the time-integration scheme.

If we set $[\![\epsilon \hat{\mathbf{n}}]\!] \approx 0$ we can take the correct value of τ_p as a constant. However, then the split and un-split fluxes are different because the diffusive fluxes are not exactly the same; the split equation's diffusive flux still contains a contribution due to the non-divergence of $\bar{\mathbf{v}}^{k+1}$. This contribution is partly removed by the rotational correction, but there are additional terms present in the normal vector $(\mathbf{Q} - \bar{\mathbf{Q}}) \cdot \hat{\mathbf{n}}$. As with any projection method scheme, this splitting error is expected to be small, particularly for large Reynolds numbers (as we will show in §3.8.3).

Finally, we note that the explicit pressure flux present in (2.78) is crucial to derive a consistent value for τ_p. If this flux was not included, we would have a ratio $\tau_p = \frac{1}{a\Delta t \tau} \frac{[\![p\hat{\mathbf{n}}]\!]}{[\![\delta p \hat{\mathbf{n}}]\!]}$, which would depend on the time-integration scheme. There are additional advantages to our flux choices, which will be discussed next.

2.3.5 Discussion and justification for discretization choices

In this section we discuss some of the discretization choices made, and explain their repercussions. Specifically, we will focus on our choice for $\widehat{p}^k$ in (2.58), and how that affects the stability parameter τ_p and the rotational correction. In general, it is important to treat the time–split equations as a single system. Treating the individual equations or steps in the time–split scheme as a single system ensures the consistency of the method. Failure to do so can lead to relatively poor numerical results, even though the underlying scheme is consistent and stable. In Appendix A we described alternative splitting schemes, unique to HDG methods, which attempt to retain the un–split system of equation on the element, while time–splitting the HDG boundary values. Here we deal with HDG discretization choices for the time–split equations using Projection methods.

In our discretization we chose to include the explicit edge-flux $\widehat{p}^k$ as part of the definition for the predictor equations for the velocity gradient tensor (2.78). However, it is tempting to choose $\widehat{p}^k = \lambda_p^k$ instead. This alternative choice seems reasonable because, in the time–split equations, we know the pressure at time–level k including its value on the HDG edge–space. A numerical discretization that uses this choice is

presented in Appendix B. The repercussions of such a flux are discussed next.

Effect of varying τ_p

The first issue with choosing $\widehat{p}^k = \lambda_p^k$ arises when solving for the consistent HDG stability parameter τ_p. As indicated in §2.3.4, this results in a value of the stability parameter that depends on the time–integration scheme, $\tau_p = \frac{1}{a\Delta t\tau}\frac{[\![p\hat{\mathbf{n}}]\!]}{[\![\delta p\hat{\mathbf{n}}]\!]}$. This value of the stability constant needs to be carefully chosen for consistent results at low resolutions. Additionally, as we will demonstrate, using a consistent value for τ_p gives an improved answer over using a value such as $\tau_p = 1$, which would be result if the second step in the time–splitting scheme, (2.59), was treated in isolation of the first (that is, not treating the time–split equations as a single system).

We can show the effect of the different values for τ_p with some simple numerical experiments. For the first experiment we calculate one time-step of the lock-exchange flow (described in §3.8). When using $\tau_p = 1$ we see that the solution is discontinuous across the elements bordering the top/bottom boundary when the solution is not well-resolved (Fig. [2-5]-top). An interesting feature is that the maximum velocity is well-approximated by this discontinuous solution. When using the consistent value for τ_p, the solution is properly numerically diffused to give a smooth field, and the maximum velocity is lower than indicated by high-resolution runs. When the resolution is increased, both choices of τ_p give approximately the same solution. These results show that increasing τ_p effectively increases the numerical dissipation in the scheme. From a theoretical perspective, as $\tau_p \to \infty$ the solution approaches a continuous Galerkin discretization (Cockburn et al., 2009a). Since continuous elements belong to a smaller function space, this means that increasing τ_p decreases the size of the solution's function space. As such, for the saddle-point maximization-minimization problem between velocity and pressure, we would expect a larger maximum velocity magnitude when τ_p is smaller, and this is exactly what we observe (Fig. [2-5]).

Additionally, this situation arises because the dominant velocity is aligned with the grid. The tangential portion of the velocity-correction is not directly penalized by the numerical scheme. That is, $\mathbf{q}_{\delta p} \cdot \hat{\mathbf{t}}$ is free to vary. Increasing τ_p, then, removes

the discontinuity in the pressure field, which indirectly also removes the discontinuity in $\mathbf{q}_{\delta p} \cdot \hat{\mathbf{t}}$ across edges. This is somewhat reminiscent of the inf–sup problem §2.2.4, where the tangential pressure-correction gradient is free to vary in this case. That is, its projection unto the velocity space is small. If we used a mesh of triangles instead of squares, (Fig. [2-6]), the discontinuity is not nearly as severe, since in this case the u-velocity component is not tangent to the diagonal edge of the triangles, and does get penalized.

There are additional repercussions of using the inconsistent value for τ_p, particularly at low-resolution. First, we show in Fig. [2-7] the result of the lock–exchange problem using our scheme from §2.3.2 with a consistent HDG stability parameter (as derived in §2.3.4). Now, to illustrate the importance of a consistent HDG stability parameter, we will utilize the scheme reported in Appendix B. In Fig. [2-8] we show three low-resolution simulations of the lock-exchange problem for different values of τ_p for the scheme reported in Appendix B. In this case, only the Fig. [2-8](b) case gives a reasonable answer, while the low τ_p case (a) does not develop the Kelvin-Helmholtz instabilities, and the high τ_p case (c) develops spurious vortices. To show the impact of the time discretization scheme, we also plot the same simulations using a non-incremental (Guermond et al., 2006) projection method (Fig. [2-9]), since the non-incremental projection method also has a constant consistent τ_p. Note, the difference is that the non-incremental scheme does not use a predictor pressure, or $\nabla p^k = 0$ in (2.4). Here the consistent value of τ_p is larger than before, by a factor of Δt. With the non-incremental method, the small value of τ_p does not develop instabilities Fig. [2-9](a) while the consistent value gives a reasonable answer Fig. [2-9](c). Additionally, when using half the consistent value Fig. [2-9](b), the non-incremental scheme still gives a reasonable answer, although the vortices are smaller. This indicates that the relative magnitude of τ_p is less sensitive for the non-incremental method. Also, comparing Fig. [2-8](c) and Fig. [2-9](c), we see that the results for the consistent τ_p (specific to those schemes) are not the same. Our numerical experiments indicate that the size of the vortices for this benchmark are sensitive to small numerical perturbations at this spatial resolution. When the spatial resolution

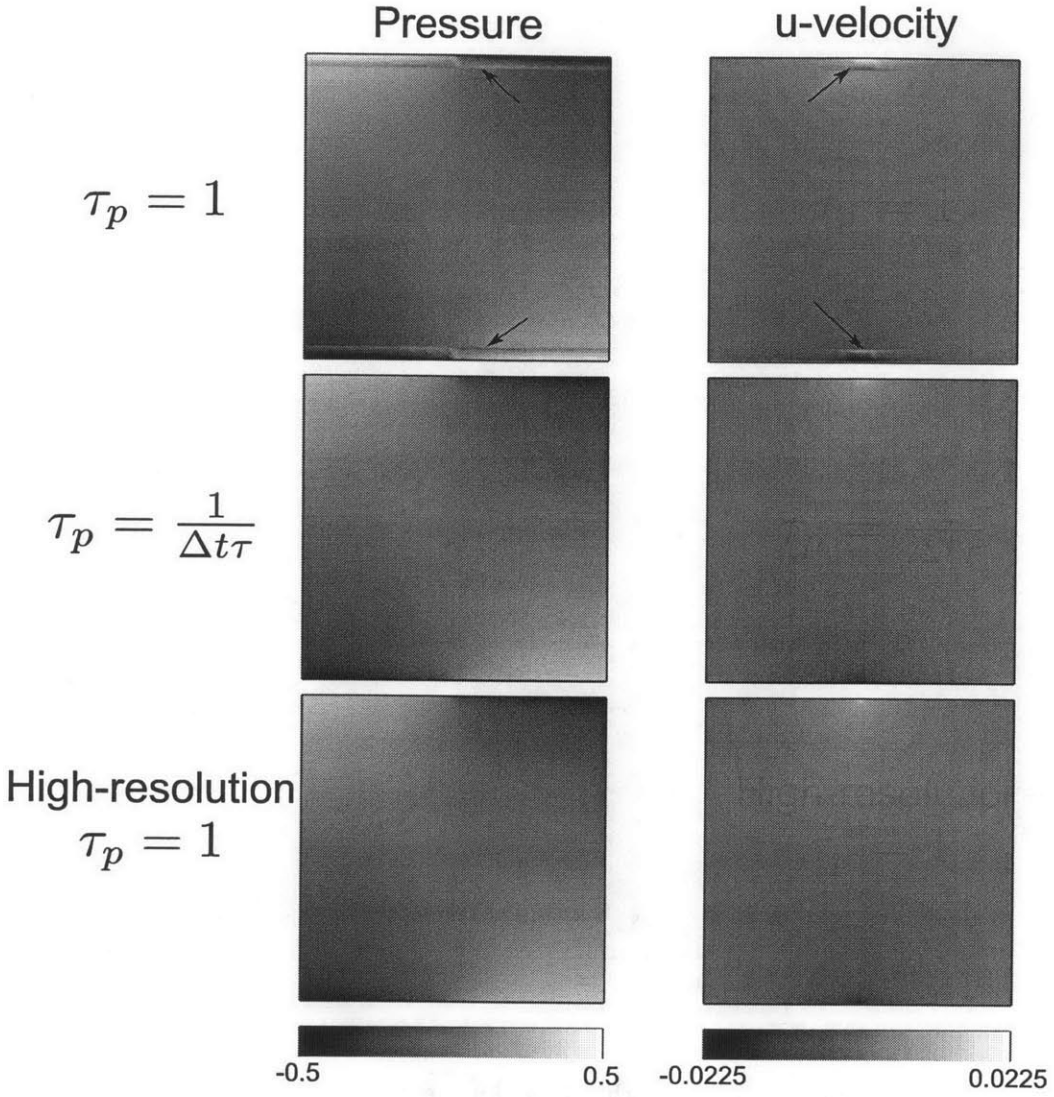


Figure 2-5: The pressure and u-velocity after 1 time-step $\Delta t = 0.01$ of the lock-exchange problem (see §3.8) using a low-resolution (p = 1, $\Delta x = 0.04$) mesh of square elements with $\tau_p = 1$ (top), $\tau_p = \frac{1}{\Delta t \tau}$ (middle), and a high-resolution (p = 5, $\Delta x = 0.01$) square mesh. The $\tau_p = \frac{1}{\Delta t \tau}$ case (middle) has a lower u-velocity magnitude and the solution is smooth, while the low resolution $\tau_p = 1$ case (top) does capture the maximum velocity, but the solution has large discontinuities at the locations indicated with arrows. When the solution is properly resolved, the $\tau_p = 1$ and $\tau_p = \frac{1}{\Delta t \tau}$ (not shown) cases give essentially the same solution.

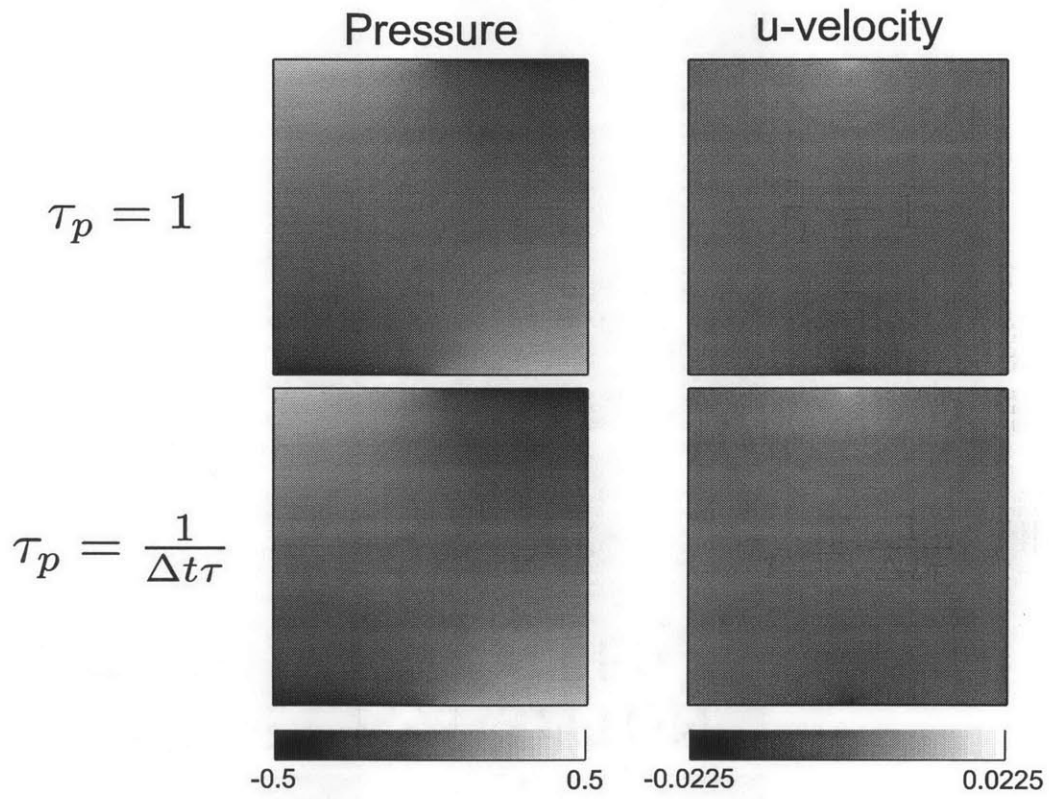


Figure 2-6: As in Fig. [2-5] but with triangular elements. The triangular mesh does not have a large discontinuity for $\tau_p = 1$ because the tangential correction velocity is penalized.

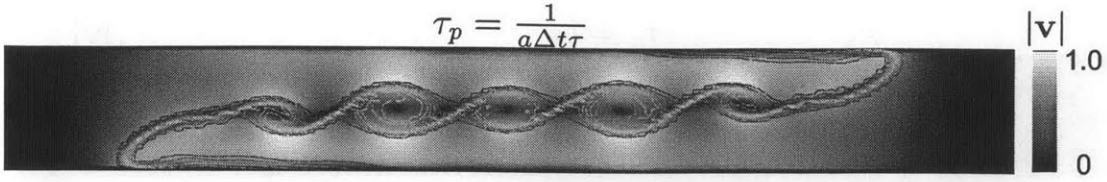


Figure 2-7: Density contours over velocity magnitude at time 10 of the lock-exchange problem (see §3.8) using a time step of $\Delta t = 0.001$, a mesh of 100×25 linear elements and a first-order accurate incremental pressure-correction scheme derived in §2.3.2.

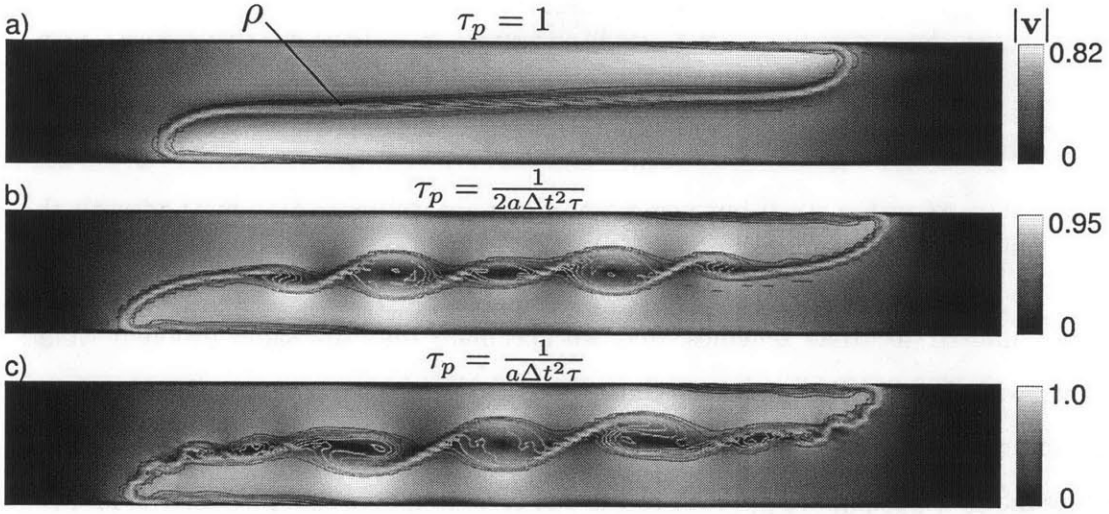


Figure 2-8: As in Fig. [2-7], but for the Appendix B scheme with (a) $\tau_p = 1$, (b)$\tau_p = \frac{1}{2\Delta t^2}$, (c) $\tau_p = \frac{1}{\Delta t^2}$.

is increased to 400 × 100, both schemes give the same answer (not shown). Finally, when we use the scheme derived in §2.3.2, and use its consistent τ_p, we see a result very similar to the non–incremental version of the Appendix B scheme (compare Fig. [2-9](c) and Fig. [2-7]). Regardless, these results show that the consistent value for τ_p gives qualitatively improved results over an inconsistent value of τ_p at coarse spatial resolutions.

We have shown that the magnitude of τ_p can significantly affect the numerical solution for hybrid discontinuous schemes when the flow is not well-resolved and the dominant flow is aligned with the element edges. We have derived the consistent magnitude for this parameter, and we have shown how the solution of the lock-exchange problem is affected by using different values. We suspect that this issue

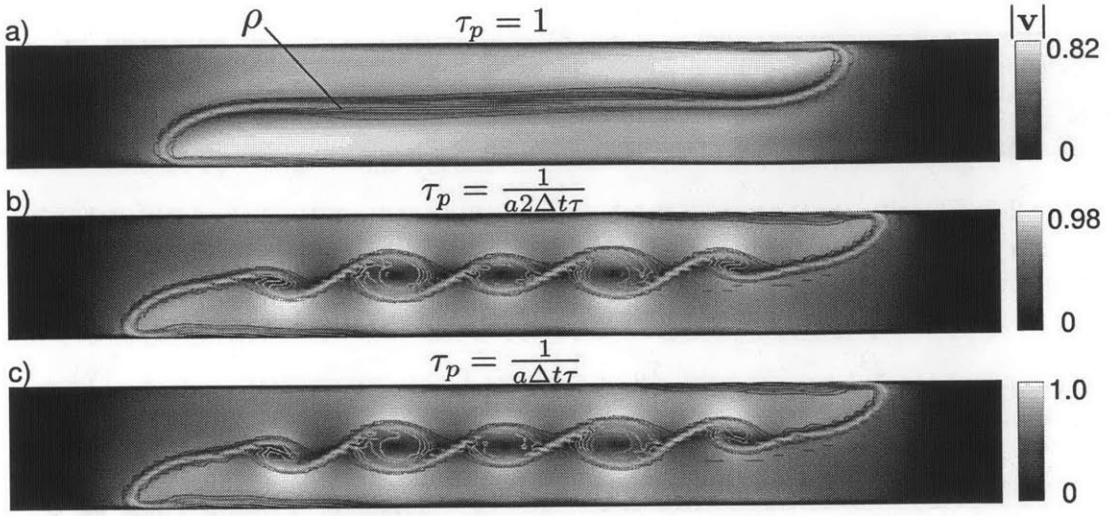


Figure 2-9: As in Fig. [2-7] but using a non-incremental projection method with the Appendix B scheme, and (a) $\tau_p = 1$, (b)$\tau_p = \frac{1}{2\Delta t}$, (c) $\tau_p = \frac{1}{\Delta t}$.

is not limited to HDG schemes, and we speculate that the same problem would occur with locally discontinuous Galerkin (LDG), internal penalty (IP), or other discontinuous Galerkin methods. If we are correct and the problem does arise with other methods, a similar approach to the one presented here could be used to find the correct scaling of penalty terms. That is, the time–split equations need to be treated as a single system, and consistent penalty terms have to be used in each case.

The rotational correction

The second issue with choosing $\widehat{p}^k = \lambda_p^k$ arises when applying the rotational correction term. With our choice of including the explicit pressure jump in the implicit flux for the velocity–predictor (2.65), we avoid having to correct the pressure on the HDG edge–space. When using the scheme in Appendix B, either the edge–space of the pressure needs to be corrected (since it is required to calculate the pressure gradient), or the pressure gradient needs to be corrected directly.

The problem with correcting the edge–space of the pressure is that the rotational correction, $\nabla \cdot \bar{\mathbf{v}}$, is not uniquely defined on the edge–space. It can be constructed using components of $\widehat{\bar{\mathbf{Q}}}$, but only $\widehat{\bar{\mathbf{Q}}} \cdot \hat{\mathbf{n}}$ is uniquely defined (since we solve the global equation (2.66)), while $\widehat{\bar{\mathbf{Q}}} \cdot \hat{\mathbf{t}}$ is discontinuous. Using an inconsistent or non–conservative

value for the rotational correction on the edge–space can lead to instabilities or poor solutions. Thus a strategy that corrects the pressure gradient directly is employed in Appendix B, and details can be found therein. The rotational–correction should be explicitly calculable from known quantities, but because the HDG method is inherently implicit, a global solution is required to correct the edge–space pressure or pressure–gradient with the rotational term (see Appendix B). Our scheme in §2.3.2 essentially combines the global calculation of the consistent edge–space rotational correction with the calculation of the predictor velocity.

2.4 High-order time discretization using IMEX-RK schemes

Once our equations have been spatially discretized, there are various ordinary differential equation solvers that could be used to advance the equations in time. Designing new higher-order accurate time-discretization schemes are beyond the scope of this thesis, as such we intend to use existing methods (Kennedy and Carpenter, 2003, Ascher et al., 1997). Unfortunately we cannot simply apply existing methods. This is, due to the nature of projection methods, which already introduce a time-discretization, and HDG methods, which have an inherent implicit component. A question also arises as to whether the pressure should be treated implicitly or explicitly. As a result, we describe the modifications required to solve our equations using semi-implicit Runge-Kutta (RK) methods. RK methods are attractive because they are self-starting, and allow for variable time-step sizes.

We are interested in IMplicit EXplicit (IMEX) Runge-Kutta (RK) time-stepping schemes with s stages that are of the form [1]:

$$\phi^{k+1} = \phi^k + \Delta t \sum_{i=0}^{s-1} b_i^{im} f^{im}(\phi_i) + \Delta t \sum_{i=0}^{s-1} b_i^{ex} f^{ex}(\phi_i), \tag{2.91}$$

[1]For MATLAB and FORTRAN programmers, note that Python/C/C++ indices are used, starting at 0 and ending at $s-1$ for an s-stage scheme.

where the stage variables are solved using

$$\phi_i = \phi^k + \Delta t \sum_{j=0}^{i} a_{ij}^{im} f^{im}(\phi_j) + \Delta t \sum_{j=0}^{i-1} a_{ij}^{ex} f^{ex}(\phi_j)$$
$$(1 + \Delta t a_{ii}^{im} f^{im})\phi_i = \phi^k + \Delta t \sum_{j=0}^{i-1} a_{ij}^{im} f^{im}(\phi_j) + \Delta t \sum_{j=0}^{i-1} a_{ij}^{ex} f^{ex}(\phi_j), \tag{2.92}$$

where ϕ is some field satisfying the equation $\frac{\partial \phi}{\partial t} = f^{im} + f^{ex}$ and $\phi_0 = \phi^k$. IMEX schemes treat one part of the right-hand-side implicitly (usually stiff terms such as diffusion) and the other part explicitly (2.92). As such, we require two Butcher Tableaus, one for the implicit terms, and one for the explicit terms. Also, we will only consider schemes with Butcher Tableaus of the form:

$$\begin{array}{c|cccc} 0 & 0 & \dots & \dots & 0 \\ dt_1 & a_{1,0}^{ex} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & a_{(s-1),0}^{ex} & \dots & a_{(s-1),(s-2)}^{ex} & 0 \\ \hline & b_0 & \dots & b_{s-2} & a \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ dt_1 & a_{1,0}^{im} & a & \dots & 0 \\ \vdots & \vdots & \ddots & & 0 \\ 1 & a_{(s-1),0}^{im} & \dots & a_{(s-1),(s-2)}^{im} & a \\ \hline & b_0 & \dots & b_{s-2} & a \end{array} \tag{2.93}$$

where these schemes have the following properties:

1. $b_i = b_i^{ex} = b_i^{im}$
2. $dt_{s-1} = 1$
3. $a_{ij}^{ex} = 0 \ \ \forall j \geq i$
4. $a_{ij}^{im} = 0 \ \ \forall j > i$
5. $a_{ii}^{im} = a \ \ \forall 0 < i < s$

Item 1 is an often-used constraint in the IMEX RK literature (e.g. (Kennedy and Carpenter, 2003)). Item 2 is useful when using the HDG framework, as will be explained below. Item 3 is a necessary condition for an explicit RK scheme. Finally, items 4

and 5 are important for an efficient implicit method, where a single diagonal entry allows for the creation of a single implicit matrix, and zero entries above the diagonal allow different stages to be solved sequentially (as opposed to simultaneously).

The time-stepping procedure is complicated by the projection method's time-splitting which needs to happen within every RK stage. If we write the momentum equations for the true velocity from (2.3) as

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{F}^{im} + \mathbf{F}^{ex} - \nabla p, \tag{2.94}$$

where $\mathbf{F}^{im} = \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v}, \mathbf{F}^{ex} = \mathbf{F}_{\partial t}$, then we can write a typical IMEX RK stage calculation as:

$$\begin{aligned} \mathbf{v}_i - a\Delta t \bar{\mathbf{F}}_i^{im} &= \mathbf{v}^k + \Delta t \sum_{j=0}^{i-1} a_{i,j}^{im} \mathbf{F}_j^{im} + \Delta t \sum_{j=0}^{i-1} a_{i,j}^{ex} \mathbf{F}_j^{ex} \\ &\quad - \Delta t \sum_{j=0}^{i-2} a_{i,j}^{ex} \nabla p_j - \Delta t a_{i,i-1}^{ex} \nabla p_{i-1}, \\ \Rightarrow \left(1 - a\Delta t \nabla \frac{1}{\mathrm{Re}} \cdot \nabla\right) \mathbf{v}_i &= \mathbf{v}^k + \Delta t \sum_{j=0}^{i-1} a_{i,j}^{im} \mathbf{F}_j^{im} + \Delta t \sum_{j=0}^{i-1} a_{i,j}^{ex} \mathbf{F}_j^{ex} \\ &\quad - \Delta t \sum_{j=0}^{i-2} a_{i,j}^{ex} \nabla p_j - \Delta t a_{i,i-1}^{ex} \nabla p_{i-1}, \end{aligned} \tag{2.95}$$

where $\mathbf{F}_i^{ex}$ and $\mathbf{F}_i^{im}$ contain the explicit and implicit terms calculated at previous stages, respectively, and we are treating the pressure explicitly. However, for the split equations, we would not yet know the correct ∇p_{i-1} at each stage that would balance the divergent parts of the explicit terms which will render $\nabla \cdot \mathbf{v}_i = 0$. Hence, we now have to explain the modifications needed to the IMEX-RK integration to incorporate our time–splitting.

As discussed in §2.2.2, the purpose of the pressure is to balance the divergent terms in the right-hand-side of the equation, and the implicit terms evaluated using a divergence-free velocity should remain divergence-free (see §2.2.1). As such, the explicit contribution, $\mathbf{F}_{i-1}^{ex}$ which would have been newly calculated from the velocity

at the previous stage $\bar{\mathbf{v}}_{i-1}$, is the only source of divergence in the right-hand-side. Unfortunately, at this point in the stage calculation we do not yet know the correct ∇p_{i-1} that would balance the divergent parts of the explicit terms which will render $\nabla \cdot \mathbf{v}_i = 0$. As such, using the incremental pressure-correction method, we predict the pressure at stage $i-1$ using a function of the previously calculated pressures,

$$\nabla p_{i-1} \approx \nabla p_{i\star} = \nabla F(p_0, \ldots, p_j),\ j < i-1.$$

A typical, time-split, IMEX RK stage calculation for the predictor velocity $\bar{\mathbf{v}}_i$ can then be written as follows, where we have also divided by Δt

$$\begin{aligned}\left(\frac{1}{\Delta t} - a\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\right)\bar{\mathbf{v}}_i = \frac{\bar{\mathbf{v}}^k}{\Delta t} + \sum_{j=0}^{i-1} a_{i,j}^{im}\mathbf{F}_j^{im} + \sum_{j=0}^{i-1} a_{i,j}^{ex}\mathbf{F}_j^{ex} \\ - \sum_{j=0}^{i-2} a_{i,j}^{ex}\nabla p_j - a_{i,i-1}^{ex}\nabla p_{i\star}.\end{aligned} \tag{2.96}$$

Proceeding as in §2.2, we perform the projection step (2.8)

$$-\nabla^2\delta p_i = -\frac{\nabla\cdot\bar{\mathbf{v}}_i}{\Delta t},$$

and correct the stage velocity as in (2.10)

$$\mathbf{v}_i = \bar{\mathbf{v}}_i - \Delta t\nabla\delta p_i.$$

To find what the pressure correction should be, we follow the same procedure in §2.2.1 and find that we require a new scaling in the correction terms

$$p_{i-1} = p_{i\star} + \frac{1}{a_{i,i-1}^{ex}}\delta p_i - \frac{a}{a_{i,i-1}^{ex}}\frac{1}{\mathrm{Re}}\nabla\cdot\bar{\mathbf{v}}_i.$$

To prove this new result, substitute for $\mathbf{v}_i$ into (2.96)

$$\left(\frac{1}{\Delta t} - a\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\right)(\bar{\mathbf{v}}_i + \Delta t\nabla\delta p_i) = \frac{\bar{\mathbf{v}}^k}{\Delta t} + \sum_{j=0}^{i-1} a_{i,j}^{im}\mathbf{F}_j^{im} + \sum_{j=0}^{i-1} a_{i,j}^{ex}\mathbf{F}_j^{ex} - \sum_{j=0}^{i-2} a_{i,j}^{ex}\nabla p_j - a_{i,i-1}^{ex}\nabla p_{i\star}.$$

Then subtract out the un-split equations

$$\left(\frac{1}{\Delta t} - a\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\right)\Delta t\nabla\delta p_i = -a_{i,i-1}^{ex}\nabla p_{i\star} + a_{i,i-1}^{ex}\nabla p_{i-1}.$$

Now solve for ∇p_{i-1}, followed by p_{i-1} after substituting (2.8)

$$\begin{aligned} a_{i,i-1}^{ex}\nabla p_{i-1} &= a_{i,i-1}^{ex}\nabla p_{i\star} + \nabla\delta p_i - a\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\Delta t\nabla\delta p_i, \\ \Rightarrow \nabla p_{i-1} &= \nabla\left\{p_{i\star} + \frac{1}{a_{i,i-1}^{ex}}\delta p_i - \frac{a}{a_{i,i-1}^{ex}}\nabla\frac{1}{\mathrm{Re}}\cdot\nabla\Delta t\delta p_i\right\} \\ \Rightarrow p_{i-1} &= p_{i\star} + \frac{1}{a_{i,i-1}^{ex}}\delta p_i - \frac{a}{a_{i,i-1}^{ex}}\frac{1}{\mathrm{Re}}\nabla\cdot\bar{\mathbf{v}}_i. \end{aligned}$$

Note that this gives the pressure at the present stage time.

While this procedure allows us to calculate intermediate divergence-free stage variables, what the IMEX RK scheme actually requires are the implicit and explicit function evaluations. Thus, the explicit pressure contributions need to be updated to replace the guessed values $\nabla p_{i\star}$ in the right-hand side of the momentum equations. While we could evaluate the implicit diffusion terms using $\mathbf{v}_i$, it is more efficient computationally to solve for $\mathbf{F}^{im}$ from (2.95) as follows

$$\begin{aligned} \mathbf{v}_i - a\Delta t\mathbf{F}_i^{im} &= \mathbf{v}^k + \Delta t\sum_{j=0}^{i-1} a_{i,j}^{im}\mathbf{F}_j^{im} + \Delta t\sum_{j=0}^{i-1} a_{i,j}^{ex}\mathbf{F}_j^{ex} \\ &\quad - \Delta t\sum_{j=0}^{i-2} a_{i,j}^{ex}\nabla p_j - \Delta t a_{i,i-1}^{ex}\nabla p_{i-1}, \\ \Rightarrow \mathbf{F}_i^{im} &= \frac{\mathbf{v}_i - \mathbf{v}^k}{a\Delta t} - \frac{1}{a}\left\{\sum_{j=0}^{i-1} a_{i,j}^{im}\mathbf{F}_j^{im} + \sum_{j=0}^{i-1} a_{i,j}^{ex}\mathbf{F}_j^{ex} - \sum_{j=0}^{i-1} a_{i,j}^{ex}\nabla p_j\right\} \end{aligned} \tag{2.97}$$

With this procedure, the implicit terms should be completely divergence-free, that is $\nabla \cdot \mathbf{F}_i^{im} = 0$, and also the explicit terms $\nabla \cdot \left\{ \mathbf{F}_j^{ex} - \nabla p_j \right\} = 0$. If not for the splitting error, this shows that the pressure is an explicit contribution, and should be accurate up to the explicit order of accuracy.

Unfortunately, from the discussions in §2.2.2 and §2.2.3, we will show that the pressure also balances divergent components of the implicit terms, which arise due to the boundary conditions. This means that the implicit term is not divergence-free by itself, but instead $\nabla \cdot \left\{ a\mathbf{F}_i^{im} + a_{i,i-1}^{ex}\mathbf{F}_{i-1}^{ex} - a_{i,i-1}^{ex}\nabla p_j \right\}$. This has unfortunate consequences, since in subsequent stages, the ratio $a : a_{i,i-1}^{ex}$ is not maintained, and the pressure calculated at the subsequent stage will have to balance the divergence from right-hand-side contributions of previous stages. This is particularly evident at the final solution stage, and leads to the requirement that $dt_{s-1} = 1$ (item 2 above).

Consider the final re-combination step (2.91). For this step, we need $\mathbf{F}_{ex}(\mathbf{v}_{s-1})$, which is newly calculated after solving for the final stage values. For the Navier-Stokes equations, this means we have to evaluate the non-linear advection terms, which will be divergent. As such, we need to calculate the pressure p_{s-1}, which will balance these divergent terms, and lead to a correct divergence-free velocity $\mathbf{v}^{k+1}$. To do so we have to define $\widehat{\widehat{\mathbf{v}}}_\star^{k+1}$ (see §2.3.2), and if we have chosen to use the HDG flux $\widehat{\widehat{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}$ (2.83), we have to solve for the solution on the HDG edge space at the recombination step. To do so, we ignore the already-calculated $\mathbf{F}_{s-1}^{im}$, which is the implicit terms evaluated at $dt_{s-1} = 1$, and recompute this implicit contribution, similar to the procedure followed for the stage calculations. That is, after the pressure-correction step we have:

$$\mathbf{v}^{k+1} - a\Delta t\mathbf{F}^{im,k+1} = \mathbf{v}^k + \Delta t \sum_{j=0}^{s-1} b_j \left\{ \mathbf{F}_j^{im} + \mathbf{F}_j^{ex} - \nabla p_j \right\}. \tag{2.98}$$

This allows us to use the HDG flux for $\widehat{\widehat{\mathbf{v}}}_\star^{k+1}$, at the cost of one additional matrix-inversion for the implicit diffusion term. However, we do not expect this to add significantly to the cost of the scheme. This is because the solution of the pressure equation generally dominates the total cost, and we have an excellent guess for $\bar{\boldsymbol{\lambda}}^{k+1}$,

since $\boldsymbol{\lambda}_{s-1}$ is known from the stage calculation, and it is is an approximation of the velocity on the HDG edge space at the $k+1$ time-level.

In summary, to use IMEX RK schemes with the HDG Projection method discretization, two new modifications are required. First, the projection step has to be carried out at every stage and at the recombination stage. Second, the implicit diffusion term has to be re-evaluated at the recombination stage, if the HDG flux is chosen for $\widehat{\mathbf{v}}_{\star}^{k+1}$.

2.5 Explicit hybrid discontinuous Galerkin schemes

In the §2.4 we saw that we needed to implicitly solve the HDG system of equations at the recombination step. This was because we did not know what the solution was on the HDG edge space. A very similar problem arose in §B.2, where we had to define the velocity correction, $\boldsymbol{\lambda}_{\text{cor}}$ on the HDG edge space. In both cases it would have been useful if we could have evaluated the HDG system explicitly. One might hope that since it is possible to either implicitly or explicitly evaluate other schemes, such as the locally discontinuous Galerkin method, that the same could be done for HDG methods. However, we will show that even an explicit evaluation of the HDG fluxes requires a matrix inversion. This is because the HDG edge–space is global, and hence inherently implicit regardless of the time–integration procedure. The following argument is carried out for readers unfamiliar with this property of HDG methods.

Consider the first time-step where both $\mathbf{v}$ and $\mathbf{Q}$ are known. Based on this we can calculate $\bar{\boldsymbol{\lambda}}$ from (2.85). This is enough information to advance the velocity for one time-step at the local-element level. Can we find an explicit time evolution equation for the velocity on the edge space at an edge-local level? If we take the time-derivative of (2.85), we obtain:

$$\frac{\partial \bar{\boldsymbol{\lambda}}}{\partial t} = \left\{\left\{\frac{\partial \bar{\mathbf{v}}}{\partial t}\right\}\right\} - \frac{1}{2\tau}\left[\left[\frac{\partial \bar{\mathbf{Q}}}{\partial t}\cdot\hat{\mathbf{n}}\right]\right],$$

(where we have dropped the explicit pressure term for simplification). From this we

see that we also need an equation for the time-evolution of $\mathbf{Q}$. We can obtain an element-local equation for this by taking the time-derivative of the discrete gradient equation (2.65):

$$\left((\mathrm{Re})\frac{\partial \bar{\mathbf{Q}}^{k+1}}{\partial t}, \boldsymbol{\Theta}\right)_K - \left(\nabla\frac{\partial \bar{\mathbf{v}}^{k+1}}{\partial t}, \boldsymbol{\Theta}\right)_K - \left\langle \frac{\partial \bar{\boldsymbol{\lambda}}^{k+1}}{\partial t} - \frac{\partial \bar{\mathbf{v}}^{k+1}}{\partial t}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} = 0$$

Substituting for the time-evolution equation of $\boldsymbol{\lambda}$ we obtain

$$\left((\mathrm{Re})\frac{\partial \bar{\mathbf{Q}}^{k+1}}{\partial t}, \boldsymbol{\Theta}\right)_K - \left(\nabla\frac{\partial \bar{\mathbf{v}}^{k+1}}{\partial t}, \boldsymbol{\Theta}\right)_K - \left\langle \left\{\left\{\frac{\partial \bar{\mathbf{v}}}{\partial t}\right\}\right\} - \frac{1}{2\tau}\left[\left[\frac{\partial \bar{\mathbf{Q}}}{\partial t} \cdot \hat{\mathbf{n}}\right]\right] - \frac{\partial \bar{\mathbf{v}}^{k+1}}{\partial t}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} = 0.$$

This equation highlight the inherent problem. We hope to write the above equation as an element-local equation, but the jump of the gradient includes contributions from bordering elements. As such, this problem becomes a globally coupled one. That is, a globally coupled inverse is needed to solve this equation.

We can now restate the problem as follows. Given a discontinuous field $\mathbf{v}$, what is it's HDG gradient? We can solve this problem by defining the following element-local system

$$\begin{aligned} ((\mathrm{Re})\mathbf{Q}, \boldsymbol{\Theta})_K - (\nabla\mathbf{v}, \boldsymbol{\Theta})_K + \langle \mathbf{v}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \rangle_{\partial K} &= \langle \boldsymbol{\lambda}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \rangle_{\partial K}. \\ (\mathbf{v}, \boldsymbol{\theta})_K &= (\mathbf{v}, \boldsymbol{\theta})_K \end{aligned} \tag{2.99}$$

with the globally coupled equations for $\boldsymbol{\lambda}$

$$\begin{aligned} \langle [\![\mathbf{Q} \cdot \hat{\mathbf{n}} - \tau (\mathbf{v} - \boldsymbol{\lambda})]\!], \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon &= \langle g_N, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon, \\ \boldsymbol{\lambda}|_{\partial\Omega_D} &= \mathbf{g}_D. \end{aligned} \tag{2.100}$$

Solving this set of equation for the HDG edge velocity $\boldsymbol{\lambda}$ will allow one to calculate the HDG derivatives of any field.

2.6 Summary

In this chapter we formulated and derived a new Boussinesq solver using the novel HDG Method, a projection method, and IMEX-RK time-integration schemes.

For the spatial discretization we mathematically derived the proper form of the HDG edge–space velocity correction and the proper form of the HDG stability parameter for the pressure. We also provided guidance on applying the rotational correction to the pressure. Finally, we detailed how we incorporate the projection method time-split within a standard IMEX-RK time-stepping scheme.

Next we will extend this scheme for the equation governing ocean flows (§4). This requires the extension to a free-surface boundary condition, and a reformulation of the Boussinesq equations appropriate for thin-fluids. Future research into alternative splitting methods based on the HDG method may be fruitful. In particular, we suggest an alternative splitting scheme in Appendix A, which will not be dependent on a properly chosen HDG stability parameter for the pressure.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Implementation of the 3D Incompressible Navier-Stokes Equations, Algorithmic Properties and Numerical Verifications

In this chapter we explain some of the details pertaining to our particular implementation of the scheme described in §2.3.2. There is often a considerable leap between the mathematical formulation and the implementation. For help with low-level implementation details (such as, how to create an orthogonal polynomial basis on a triangle), we refer to the text by Hesthaven and Warburton (2008). First we will discuss the choice of finite element basis, speaking on the advantages and disadvantages of a modal or nodal approach. Specifically we justify some of the choices made, and explain how to overcome some problems that will arise. Second, we discuss the method for discrete integration, contrasting quadrature-free and quadrature-based integration. We also discuss how to formulate a quadrature-free method that will work in conjunction with the hybrid discontinuous Galerkin (HDG) space. Third, we describe our nodal limiting procedure. Fourth, we explain how to implement the HDG method without storing the globally-coupled matrix. Fifth, we explain the

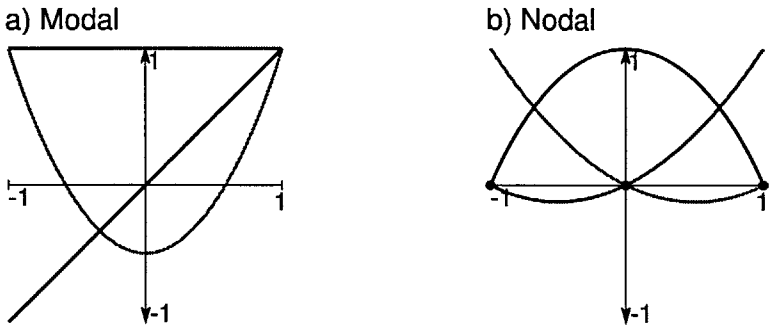


Figure 3-1: Sketch of the first three 1D (a) modal Legendre and (b) nodal Lagrange polynomial bases.

design philosophy of our python framework, and outline the primary function of different modules, as well as the main program data-flow. Then we perform verification benchmarks to demonstrate the performance of our quadrature-free HDG scheme, and our selective nodal limiter. Follow this, we describe and employ examples that contrast the continuous properties of Projection methods with the numerical equivalents. Lastly, we perform numerical tests and convergence studies to evaluate the discretized system of equations for Stokes/Navier-Stokes.

3.1 Modal vs. nodal bases and test-functions

The choice between modal or nodal bases is not an obvious one, and either choice has its own merits and pitfalls. Here we briefly describe the properties of both approaches, discussing advantages, disadvantages, and implementation issues. We mainly concentrate on polynomial bases, and issues related to Galerkin test-functions, but we also include some discussion on other choices. In this section, we will consider a function $\phi(\mathbf{x}, t)$ approximated by a finite element basis as $\phi(\mathbf{x}, t) = \sum \phi_i(t)\theta_i(\mathbf{x})$. We will simplify the notation as $\phi = \phi_i\theta_i$, where no confusion should arise. Note that our discussion will focus on scalar bases, θ, but the same properties will also apply to our vector $\boldsymbol{\theta}$ and tensor Θ bases.

3.1.1 Modal basis

A modal basis (Fig. [3-1](a)) has a number of important properties that influence the implementation of the numerical scheme. These are summarized as follows:

1. Modal bases are orthonormal with respect to the Galerkin inner product: $(\theta_i, \theta_j)_K = \delta_{ij}$, where δ is the Dirac-delta function in this case.

2. Modal bases have a natural hierarchy: as an example, for a 1D basis, $\theta_i \in \mathcal{P}^i$.

3. The number of modal bases required to fully describe a p degree polynomial depends only on the spatial dimension of the problem. It does not depend on the type of element.

4. To evaluate the value of a function at a point in the element, every modal basis needs to be evaluated at that point: $\phi(\mathbf{x}, t) = \sum \phi_i(t)\theta_i(\mathbf{x})$.

The main advantage of the modal approach is the orthonormality property (1). The orthonormality of the polynomials may be defined in terms of any chosen inner product, but in Finite Elements it is convenient to choose the basis such that the mass-matrix is diagonal. This is very convenient, since the mass-matrix needs to be inverted often, and a diagonal operator can be trivially inverted. Additionally, the derivative matrices will be sparse. Thus, modal bases have sparser numerical operators, which then require fewer floating point operations to compute. Next, the modal basis also has a natural hierarchy (2), where each basis is associated with a particular polynomial degree. For example, the first basis is usually a constant, for the p = 0 degree polynomial. The next basis, then, has linear terms, and is associated with the p = 1 degree polynomial. Each level in the hierarchy, then, is related to the derivatives of the solution. For example, the first derivative of the first basis is zero, and non-zero for the rest. This is extremely convenient for filtering or nodal limiting operations because a specified decay of the solution's derivatives can be imposed by directly modifying the coefficients of the modal basis. Additionally, p-adaptivity is easily handled with modal bases: when the order of the polynomial has to decrease or increase, simply decrease or increase the number of modal bases used. This can

potentially be implemented using a "masking" procedure. The next property (3) is convenient for meshes composed of different finite elements (for example triangles and quadrilaterals in the same mesh). Since both element types have the same number of degrees of freedom, the same data-structure can be used for both. This makes the implementation simpler because the same strides can be used in the array for all element types.

The main disadvantage to the modal approach involves interpolation, or property (4). If the solution needs to be evaluated at a point, all bases need to be evaluated at that point. For discontinuous finite element schemes, the solution on the element is frequently evaluated on the boundary of the element, so a modal basis requires more evaluations than a nodal basis. This also has major repercussions for coordinate transformations, quadrature-based integration schemes, and non-linear function evaluations with a quadrature-free integration scheme.

In the case of coordinate transformations, the Jacobians, for example, can be conveniently calculated using a nodal basis. The Jacobians from the nodal basis then have to be transformed to the modal basis, but this requires additional computation. For straight-sided simplexes (lines, triangles and tetrahedrals), this is not an issue because the Jacobians are constant. For curved elements, such as those obtained from moving mesh calculations (for example, a free-surface ocean model), the Jacobians are polynomials, and the transformations add expense at every time-step.

Next, for quadrature-based integration, the solution has to be evaluated at quadrature points. When integration over the element boundary is required, we have to evaluate every modal basis to calculate the value of the function at each of the quadrature points. The first property gave sparse operators, but the Vandermonde matrix needed to evaluate the basis at quadrature points is dense. Now, from an implementation perspective it would be convenient to have a single Vandermonde matrix for each edge that will do this evaluation. Unfortunately, the most efficient quadrature points are not necessarily symmetric on edges (Fig. [3-2](a)). Since two edges can meet at arbitrary orientations, we need a unique Vandermonde matrix for every orientation of the quadrature-points on every edge. Therefore, an implementation needs to

select between the appropriate versions of the Vandermonde matrix for each edge on an element, which adds to the complexity of data-structures and program flow. Alternatively, the polynomial can be projected from the volume-space modal basis θ unto a modal polynomial restricted to the edge-space θ_ε (Fig. [3-2](b)). In this case the implementation is simplified, but additional calculations are required on the edge-space to evaluate its polynomial at the non-symmetric quadrature points.

Finally, for equations with non-linear functional forms, quadrature-free implementations are expensive. Consider the integration of $(f(\phi), \theta_j)_K$, where f is some non-linear function (such as $\sin(\phi), |\phi|, \frac{1}{a+\phi}, ...$). In a quadrature-free approach, the coefficients ϕ_i can normally be taken out of the integration sign, and then the remaining integration of known bases θ_i and test-functions θ_j can be pre-calculated analytically (§3.2). Because of the non-linear function, this separation is not possible. As such, one has to resort to somehow projecting or interpolating the function $f(\phi)$ unto a polynomial basis $f(\phi) \approx f_i\theta_i$. Once the f_i coefficients are known, the integration can, again, be pre-calculated analytically. The projection step could be done using quadrature, but then the advantages of the quadrature-free approach is eliminated. Interpolation of the function is most easily achieved using a nodal basis, and once known, can then be transformed to a modal basis, but this requires computation which is unnecessary if using a nodal basis at the onset.

3.1.2 Nodal Basis

The important properties of nodal bases that influence the implementation of numerical schemes are summarized as follows:

1. Nodal bases are defined with respect to a set of nodal points $\mathbf{x}_i$, such that $\theta_j(\mathbf{x}_i) = \delta_{ij}$.

2. The number of nodal bases required depends on the type of element.

3. To evaluate the value of a function at an arbitrary point in the element, every nodal basis needs to be evaluated at that point: $\phi(\mathbf{x}, t) = \sum \phi_i(t)\theta_i(\mathbf{x})$.

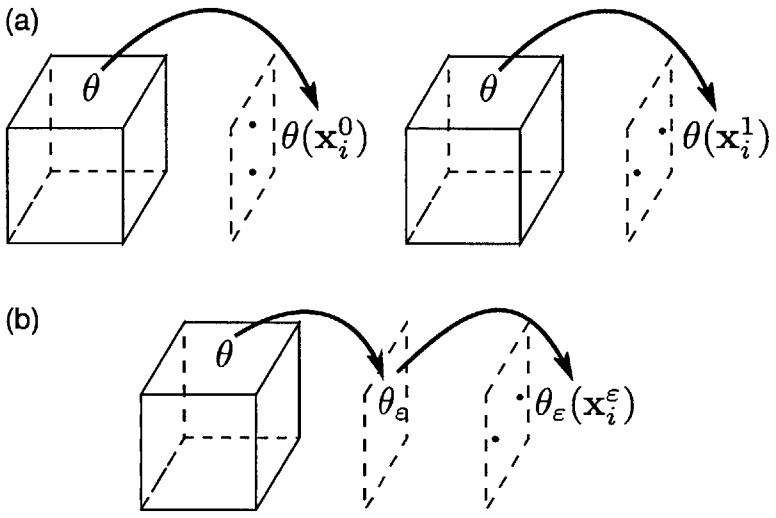


Figure 3-2: Function evaluations required for quadrature-based integration over an element edge. Each arrow represents a unique matrix operator, either a Vandermonde or projection matrix. In (a), multiple Vandermonde matrices are required for each edge to account for different orientations of the rotationally non-symmetric quadrature points $\mathbf{x}_i^0$ and $\mathbf{x}_i^1$. In (b), one projection matrix is required for each edge, but every edge also has one Vandermonde matrix, which increases the number of operations. Only one Vandermonde matrix is required because the orientation of the quadrature points $\mathbf{x}_i^\varepsilon$ are defined with respect to the orientation of the edge (and not the neighboring elements).

The main advantage of the nodal approach is property (1). This property means that the coefficients of the bases ϕ_i are equal to the value of the function at the nodal points $\mathbf{x}_i$. Therefore, if we want to evaluate the function at nodal points, no computation is required. This stands in contrast to property (3), if we want to evaluate the function at a point $\mathbf{x} \neq \mathbf{x}_i \; \forall \; i$ (at a quadrature point, for example), then this requires the same number of computations as the modal approach. This means that the difference between the efficiency of modal and nodal approaches for quadrature-based integration should be small. For quadrature-free integration, however, the nodal approach can naturally approximate the non-linear function by interpolating its values at the nodal points, with no additional computation. If the non-linear function is projected unto the basis instead of interpolated, again the modal and nodal approaches would be similar in cost. The final advantage of the nodal approach is cheap evaluation of the polynomials on element edges. In Fig. [3-2](b), the first arrow representing a projection of the volume solution unto the edge required a dense matrix operator. In the nodal case, this operator is very sparse, amounting to an indexing problem that does not involve all of the nodal polynomials. From an implementation side, the nodal approach has additional advantages. In particular, the calculation of transformation coefficients (for example the Jacobians) is naturally handled by the nodal basis. For moving-mesh applications, the implementation is therefore simplified. Also, plotting is simplified because the solution does not have to be evaluated at spatial points. In summary, the nodal approach has computational advantages over the modal approach when the scheme requires the polynomial to be evaluated at the specified nodal points. This suggests the use of quadrature-free integration to take full advantage of this property.

The main disadvantages of the nodal approach are: the mass matrix is full, there is no natural hierarchy, and different element types require different number of bases (2). The first disadvantage is a consequence of the desirable property (1). The computational penalty can largely be eliminated by pre-calculating the mass-matrix inverse, and using quadrature-free integration. Because the nodal approach does not have a natural hierarchy, selective filtering or limiting of the solution does require a

transformation to a modal basis, which increases the overall cost of the scheme. From an implementation side, because different element types require different number of bases, the data-structures are more complex. The different element types require different strides, or have to be stored in different arrays. Now, when evaluating the solution on edges, one side of the edge may have a different element type than the other. Thus, if the coefficients for different elements are stored in different arrays, the data required on the edges will come from areas that are far apart in memory. While localized memory access is always a problem with the edge operations, it is amplified in the nodal case with mixed element types.

As a final note, we could consider the nodal basis as a modal one where the inner product is defined in terms of a collocation-type projection. That is, if we choose the test-functions as $\delta(\mathbf{x}_i)$, then the nodal basis is orthonormal to the inner-product $(\theta_i, \delta(\mathbf{x}_j))_K = \delta_{ij}$. However, because we chose the Galerkin approach, the nodal bases are not orthonormal to themselves using the Galerkin inner-product $(\theta_i, \theta_j)_K \neq \delta_{ij}$.

3.1.3 Discussion of choice between modal and nodal bases

Based on the the properties of the two different types of bases, we initially pursued the modal approach. The hierarchical structure and simplified data structures for mixed element-type meshes were attractive. However, because we were also interested in quadrature-free schemes, it became clear that the nodal approach is simpler to implement in the case where we have non-linear functional forms. Additionally, because a nodal basis is most convenient for calculating coordinate transformations, it simplified the implementation of moving meshes. Therefore, our code uses a nodal basis.

From a computational efficiency perspective, it is difficult to say which approach would be more efficient. It depends on the type and polynomial degree of the elements, the implementation, and even the ordering of elements in a particular mesh. We could count the number of floating point operations required for various element types and orders, but the bottleneck for modern-day computational fluid dynamics simulations is the memory transfer bandwidth, not the number of floating point calculations per

second (McCalpin, 1995). Analyzing the cache complexity of the different approaches is beyond the scope of this thesis. As such, our decision mainly considered ease of implementation for our specific problems of interest.

3.2 Quadrature-free operators

For discontinuous Galerkin finite elements discretizations, we have to integrate polynomials over elements. That is, we have to evaluate inner products such as $(f(\phi), \theta_j)_K$. The difference between quadrature-based and quadrature-free methods is how these integrals are evaluated.

Quadrature-based integration calculates an integral by evaluating the function at quadrature points. The function evaluations at these points are then combined as a weighted sum to give the integral. That is:

$$(f(\phi), \theta_j)_K \approx \sum_{k=0}^{k<N_{qp}} f(\phi(\mathbf{x}_k))\theta_j(\mathbf{x}_k)w_k, \tag{3.1}$$

where N_{qp} are the number of quadrature points, and w_k are the quadrature weights. In 1D, Gaussian quadrature schemes will exactly integrate polynomials of degree $\mathrm{p} = 2N_{qp} - 1$.

Quadrature-free integration calculates an integral by first projecting the function unto the polynomial basis. The resulting coefficient can then be taken out of the integral, which is then calculated analytically:

$$\begin{aligned}(f(\phi), \theta_j)_K &= (f_i\theta_i, \theta_j)_K \\ &= f_i\,(\theta_i, \theta_j)_K\end{aligned} \tag{3.2}$$

where $f(\phi) \approx f_i\theta_i$ and $f_i = \left[(\theta_i, \theta_j)_K\right]^{-1} (f(\phi), \theta_j)_K$ is the coefficients of the projection of the function $f(\phi)$ unto the polynomial basis θ.

For linear functions $f(\phi)$, there is no theoretical difference in the accuracy between quadrature-based and quadrature-free integration, that is, both are exact if enough

quadrature-points are used. Computationally, the costs are also similar. For example, to calculate the integral of the function $f(\phi) = \phi$, we have to compute the integrals $(\phi_i\theta_i, \theta_j)_K$. In the quadrature-based scheme, we first need to evaluate the polynomials at the quadrature points, then we need to do the weighted sum. For this we need $N_{qp} = \mathrm{p} + \frac{1}{2}$ quadrature points in 1D for exact integration, which gives an operation count of $\mathcal{O}(4(\mathrm{p}+0.5)(\mathrm{p}+1)) \sim \mathcal{O}(4\mathrm{p}^2)$ (Ueckermann and Lermusiaux, 2010). Since the operator is pre-calculated in the quadrature-free case, the operation count is $\mathcal{O}(2(\mathrm{p}+1)(\mathrm{p}+1)) \sim \mathcal{O}(2\mathrm{p}^2)$, or 2 times fewer than the quadrature-based scheme.

For polynomial non-linearities, the costs of exact integration remain the same. However, for the non-linear advection terms in the Navier-Stokes equations, exact integration using high-order elements becomes prohibitively expensive, with little added accuracy. Consider the integration of ϕ^2 on a curved domain. In that case, we also have a non-linear Jacobian. Then our integral becomes $(f(\phi), \theta_j)_K = ((\phi_i\theta_i)(\phi_k\theta_k)(J_l\theta_l), \theta_j)_K$, where J_l are the coefficients of the spatially variable Jacobian. For an exact quadrature-based integration, we would require $N_{qp} = 2\mathrm{p} + \frac{1}{2}$ quadrature points in 1D, which results in $\mathcal{O}(8\mathrm{p}^2)$ calculations, or twice the number seen before. For 3D, it means that we need 8 times more calculations. The quadrature-free integration would have a similar additional cost if correctly implemented. The additional expense of exact integration may not come at much improved accuracy for smooth functions. As such, many researchers use fewer quadrature points than would be required for exact integration.

Approximate integration gains computational efficiency at the cost of reduced accuracy. Here we discuss the size of the resulting errors. If, for example, we use only enough quadrature points to exactly integrate linear functions, aliasing will occur. That is, when we try to project the solution unto the polynomial of degree p, higher degree polynomials that should integrate to zero will be aliased and contribute to the calculation. If the solution is smooth, the size of the next-highest polynomial will be of $\mathcal{O}\left(\frac{\Delta x^{\mathrm{p}+1}}{(\mathrm{p}+1)!}\left.\frac{\partial^{\mathrm{p}+1} f}{\partial x^{\mathrm{p}+1}}\right|_{\mathrm{x}}\right)$, from the Taylor series expansion. If we take this as a constant over the element, and integrate this error over the element, the final error is $\mathcal{O}\left(\frac{\Delta x^{\mathrm{p}+2}}{(\mathrm{p}+1)!}\left.\frac{\partial^{\mathrm{p}+1} f}{\partial x^{\mathrm{p}+1}}\right|_{\mathrm{x}}\right)$. For smooth functions, we can approximate

$\left.\frac{\partial^{p+1} f}{\partial x^{p+1}}\right|_x = \mathcal{O}(e^{-(p+1)})$, and on the master element we have $\Delta x \approx 1$, which gives an approximate error of $\mathcal{O}\left(\frac{e^{-(p+1)}}{(p+1)!}\right)$. Evaluated for p = [0, 1, 2, 3, 4] we have errors of the order $\mathcal{O}$([1, 0.37, 0.067, 0.0083, 0.00076]), for smooth functions. So, an order of magnitude in accuracy is gained for each subsequent polynomial degree. Additionally, this error will be on the order of the underlying accuracy of the scheme. The argument for inexact integration, then, is as follows: while we could eliminate the inexact integration error by increasing the number of computations, but for the same cost we could increase the number of elements (or increase polynomial order) which should decrease the overall error of the discretization. Therefore, we choose to approximately evaluate integrals of non-linear functions.

The final remaining question is how the approximations differ between quadrature-based and quadrature-free integration schemes. In both we resort to interpolating the function unto a polynomial. In the quadrature-based method, this polynomial is evaluated at the N_{qp} quadrature points, and then integrated exactly for polynomials of $p = 2N_{qp} - 1$. In the quadrature-free method, this polynomial is evaluated at the nodal points (that is interpolated, which is not exact, but will have errors as discussed above), and then integrated exactly. Since well-chosen nodal points often coincide with quadrature points, we argue that the result should be similar. If a nodal basis is used, the additional step of interpolating the function from the nodal basis unto the quadrature points may introduce additional dissipation, which is useful for stabilizing the scheme. However, it is possible to add dissipation to the nodal approach using an appropriate filter or nodal limiter (see §3.5). As such, we chose to implement a quadrature-free integration scheme that exactly integrate linear functions.

Next we describe how to implement quadrature-free integration on curved elements when also using the HDG method.

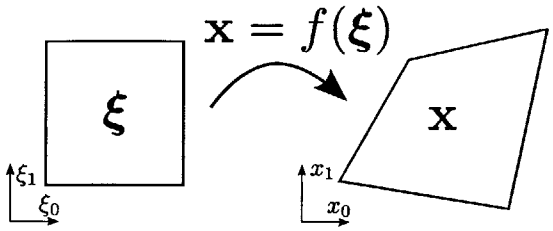


Figure 3-3: Coordinate transformation from the reference coordinate system to the physical coordinate system.

3.3 Quadrature-free integration consistent with HDG schemes

In §2.3.2 we formulated the scheme to use the strong form of the equations. The reason for this is directly tied to the quadrature-free implementation of the HDG method on general curved meshes, where the coordinate transformation factors are not constant on the element (but treated as polynomials). Here we explain why the strong form of the equations is preferred for quadrature-free HDG implementations. We first introduce some new notation, then describe our quadrature-free scheme which is compatible with HDG schemes, and finally we explain the problems with other choices.

We use $\boldsymbol{\xi}$ for the coordinates in the reference finite element (Fig. [3-3]). The coordinate transformation can then be described through a vector function $\mathbf{x} = f(\boldsymbol{\xi})$, which can also be discretized using a truncated polynomial expansion, $\mathbf{x}(\boldsymbol{\xi}) \approx \sum_i \mathbf{x}_i \boldsymbol{\theta}_i(\boldsymbol{\xi})$. In this case, a nodal basis becomes particularly useful since the coefficients of this polynomial will be equal to the real-space coordinate at the nodal points of the reference element, or $\mathbf{x}_i = \mathbf{x}(\boldsymbol{\xi}_i)$. With this polynomial representation of the coordinate transformation, we can perform the necessary numerical derivatives, integrals, and other operations after computing: the entries of the $\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}$ matrix for every element; the determinant of the $\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}$ matrix for all elements and edges; and the normal vector $\hat{\mathbf{n}}$ for every edge. We shall refer to individual entries in the first matrix as the "Jacobian factors," and to the determinant of the second matrix as the "Jacobian."

It is simple to calculate derivatives $\frac{\partial}{\partial \boldsymbol{\xi}}$ on the master element (since we simply

take the derivative of the known polynomial on the reference element), but we are interested in calculating derivatives in the physical space $\frac{\partial}{\partial \mathbf{x}}$. Numerically, we always calculate derivatives on the reference element, and then use the chain rule to obtain the desired derivative $\frac{\partial}{\partial \mathbf{x}} = \frac{\partial}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}$. To calculate the Jacobian factors, then, we use the identity:

$$\frac{\partial x_i}{\partial \xi_j} \frac{\partial \xi_j}{\partial x_k} = \delta_{ik} \tag{3.3}$$

$$\frac{\partial \xi_j}{\partial x_k} = \left[\frac{\partial x_i}{\partial \xi_j}\right]^{-1} \delta_{ik} \tag{3.4}$$

However, when using a quadrature-free scheme, it is important that this property is maintained discretely. Similar to the derivation in §2.3.2, we can show that the gradient of a scalar function ϕ (giving a vector function) can be taken discretely in the strong form and weak forms using vector $\boldsymbol{\theta}$'s as:

$$(\nabla \phi)_{\text{strong}} \approx \left[(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_K\right]^{-1} \left\{(\nabla \phi, \boldsymbol{\theta}_j)_K + \left\langle \hat{\phi} - \phi, \hat{\mathbf{n}} \cdot \boldsymbol{\theta}_j \right\rangle_{\partial K}\right\} \tag{3.5}$$

$$(\nabla \phi)_{\text{weak}} \approx \left[(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_K\right]^{-1} \left\{-(\phi, \nabla \cdot \boldsymbol{\theta}_j)_K + \left\langle \hat{\phi}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta}_j \right\rangle_{\partial K}\right\} \tag{3.6}$$

Let us define some discrete matrices

$$\mathbf{M}_e = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_{\partial K^{\text{ref}}}, \qquad \mathbf{M}_\varepsilon = (\boldsymbol{\theta}_{\varepsilon,i}, \boldsymbol{\theta}_{\varepsilon,j})_{\varepsilon^{\text{ref}}},$$

$$\mathbf{S} = (\boldsymbol{\theta}_i, \nabla \boldsymbol{\theta}_j)_{K^{\text{ref}}}, \qquad \mathbf{M} = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)_{K^{\text{ref}}},$$

$$\mathbf{D} = \mathbf{M}^{-1}\mathbf{S}^T, \qquad \mathbf{L} = \mathbf{M}^{-1}\mathbf{M}_e,$$

$$\mathbf{J} = \det\left[\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right]_K, \qquad \mathbf{J}_e = \det\left[\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right]_{\partial K},$$

where K^{ref} is the reference element. Note that $\mathbf{M}_\varepsilon$ is a matrix of size $N_{b,e} \times N_{b,e}$ and $\mathbf{M}_e$ is a matrix of size $N_b \times (\sum_{i=0}^{i<N_e} N_{b,e,i})$, where N_b is the number of bases on the element, $N_{b,e,i}$ is the number of bases on edge i of the element, and N_e is the number of edges in an element. Also, $\mathbf{J}$ and $\mathbf{J}_e$ are diagonal matrices, where each element of the diagonal is the Jacobian at that particular nodal point.

We can now write the discrete analogue of (3.3) using the strong form (3.5) as

$$\frac{\partial x_i}{\partial \xi_j}\frac{\partial \xi_j}{\partial x_k} \approx [\mathbf{D}_j\mathbf{x}_i]\frac{\partial \xi_j}{\partial x_k} + \mathbf{J}^{-1}\mathbf{L}(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e\hat{\mathbf{n}}_k = \delta_{ik},$$
$$[\mathbf{D}_j\mathbf{x}_i]\frac{\partial \xi_j}{\partial x_k} = \delta_{ik} - \mathbf{J}^{-1}\mathbf{L}(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e\hat{\mathbf{n}}_k,$$

where, the Jacobian factor $\frac{\partial \xi_j}{\partial x_k}$ takes a value at each nodal point in the domain, and $\hat{\mathbf{n}}_k$ is the k^{th} component of the normal. Numerically, $\mathbf{D}_j\mathbf{x}_i$ is a matrix-vector multiplication, with the Jacobian factor being multiplied term-by-term to the result (i.e. Hadamard/Schur product). We can solve for the Jacobian factors using the discrete analogue of (3.4) as:

$$\frac{\partial \xi_j}{\partial x_k} = [\mathbf{D}_j\mathbf{x}_i]^{-1}\left[\delta_{ik} - \mathbf{J}^{-1}\mathbf{L}(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e\hat{\mathbf{n}}_k\right], \tag{3.7}$$

which simplifies to

$$\frac{\partial \xi_j}{\partial x_k} = [\mathbf{D}_j\mathbf{x}_i]^{-1} \tag{3.8}$$

for a continuous mesh. This discrete form is not unique, and some other choices will be discussed. First a few remarks about the present form:

1. The Jacobians satisfy the identity (3.3) analytically in the volume term: $[\mathbf{D}_j\mathbf{x}_i]\frac{\partial \xi_j}{\partial x_k} \equiv \left[(\mathbf{M}^{-1})(\mathbf{S}^T\mathbf{x}_i)\right]\frac{\partial \xi_j}{\partial x_k}$.

2. The edge-term $\mathbf{J}^{-1}\mathbf{L}(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e\hat{\mathbf{n}}_k = \mathbf{J}^{-1}\mathbf{M}^{-1}\mathbf{M}_e(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e\hat{\mathbf{n}}_k$ does not require a Jacobian factor since $\hat{\mathbf{n}}$ is the real-space normal, and the edge Jacobian is used.

3. The edge-term calculated here (the element-local equation for the gradient) matches the discrete form of the HDG edge-conservation equation $\langle [\![\hat{\mathbf{q}_{\delta p}} \cdot \hat{\mathbf{n}}]\!], \theta_e\rangle_e \approx \mathbf{M}_e [\![\hat{\mathbf{q}_{\delta p}} \cdot \hat{\mathbf{n}}]\!] \mathbf{J}_e$. The HDG flux-conservation equation is used to enforce the continuity constraint (2.70), and needs to be consistent with the discrete continuity constraint in the element-local equations (2.69) in order to satisfy continuity nu-

merically.

Remark (1) holds analytically for the continuous operators and leads to a convenient simplification for the numerical scheme without loss of accuracy. Remark (2) reflects an important choice made for this scheme, and this choice leads to the result of remark (3). The challenge with a quadrature-free implementation of the HDG scheme (which remark (3) solves) is maintaining numerically-consistent edge integration terms. Without consistent edge integrals, the conservative flux calculated on the HDG space will not be numerically conserved in the element-local calculations. This makes the numerical solution of the Navier-Stokes equations unstable.

To appreciate the advantages of (3.7), consider the same approach using the weak form of the equations. This leads to the following for the Jacobian factors:

$$\frac{\partial \xi_j}{\partial x_k} = \left[-\mathbf{M}^{-1}\mathbf{S}_j\mathbf{x}_i\right]^{-1}\left[\delta_{ik} - \mathbf{J}^{-1}\mathbf{L}(\hat{\mathbf{x}}_i)\mathbf{J}_e\hat{\mathbf{n}}_k\right].$$

While this may appear reasonable, the weak derivative $[-\mathbf{M}^{-1}\mathbf{S}_j\mathbf{x}_i]$ is singular. So, using the same strategy with a weak formulation fails. Alternatively, if we evaluate the edge-integrals in the reference space, this gives:

$$\frac{\partial \xi_j}{\partial x_k} = \left[\mathbf{D}_j\mathbf{x}_i + \mathbf{L}(\hat{\mathbf{x}}_i - \mathbf{x}_i)\mathbf{J}_e^{\text{ref}}\hat{\mathbf{n}}_k^{\text{ref}}\right]^{-1}.$$

While this approach works for both weak and strong forms, these edge integrals are no longer consistent with the HDG integrals. Even though the form of the edge-integrals in the HDG method could be modified, the above weak-form approach is less efficient than the strong formulation (3.7). This approach requires d^2 edge integrals, while the strong form (3.7) only requires d calculations, where d is the dimension of the problem.

The calculation of the Jacobians and edge normals follow the usual approach in quadrature-free methods. As such, the only challenge was dealing with the Jacobian factors and the quadrature-free derivative terms. We showed that the strong formulation has distinct advantages over the weak formulation in this case. We verify that

this approach works in §3.7, using a simple steady diffusion problem.

3.4 Implementation of HDG schemes

The implementation of a HDG scheme is challenging, and as such we elucidate some of the subtleties. We first explain how the solution on an element is parameterized in terms of the boundary conditions following Nguyen et al. (2009a). Then we outline some of the difficulties implementing the HDG flux conservation equation. Additionally, we suggest a matrix-free implementation, suitable for iterative schemes.

Consider the time-dependent heat diffusion equation for $\phi = \phi^{k+1}$ discretized using HDG finite elements (where we have dropped the superscripts for simpler notation):

$$\left(\kappa^{-1}\mathbf{q}_\phi, \boldsymbol{\theta}\right)_K - \left(\nabla\phi, \boldsymbol{\theta}\right)_K + \left\langle \phi, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle \lambda_\phi, \hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K}, \tag{3.9}$$

$$\left(\frac{\phi}{a\Delta t}, \theta\right)_K - \left(\nabla\cdot\mathbf{q}_\phi, \theta\right)_K + \left\langle \tau\phi, \theta\right\rangle_{\partial K} = \left\langle \tau\lambda_\phi, \theta\right\rangle_{\partial K} + \left(F_\phi, \theta\right)_K, \tag{3.10}$$

where its derivation is very similar to that of (2.64)-(2.65). To match the quadrature-free derivatives (§3.3), we have to multiply these element-local equations by an inverse mass-matrix, which gives the following discrete matrix system of equations in $d = 3$

$$\mathbf{A}^{\text{loc}}\begin{bmatrix} \mathbf{q}_{0,\phi} \\ \mathbf{q}_{1,\phi} \\ \mathbf{q}_{2,\phi} \\ \phi \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_0\mathbf{J}_e)\lambda_\phi \\ \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_1\mathbf{J}_e)\lambda_\phi \\ \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_2\mathbf{J}_e)\lambda_\phi \\ F_\phi + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\mathbf{J}_e\tau)\lambda_\phi \end{bmatrix}, \tag{3.11}$$

where $\mathbf{A}^{\text{loc}}$ is defined as

$$\mathbf{A}^{\text{loc}} = \begin{bmatrix} \kappa^{-1}\mathbf{I} & \emptyset & \emptyset & -\text{diag}\left(\frac{\partial\xi_j}{\partial x_0}\right)\mathbf{D}_j + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_0\mathbf{J}_e) \\ \emptyset & \kappa^{-1}\mathbf{I} & \emptyset & -\text{diag}\left(\frac{\partial\xi_j}{\partial x_1}\right)\mathbf{D}_j + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_1\mathbf{J}_e) \\ \emptyset & \emptyset & \kappa^{-1}\mathbf{I} & -\text{diag}\left(\frac{\partial\xi_j}{\partial x_2}\right)\mathbf{D}_j + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_2\mathbf{J}_e) \\ -\text{diag}\left(\frac{\partial\xi_j}{\partial x_0}\right)\mathbf{D}_j & -\text{diag}\left(\frac{\partial\xi_j}{\partial x_1}\right)\mathbf{D}_j & -\text{diag}\left(\frac{\partial\xi_j}{\partial x_2}\right)\mathbf{D}_j & (a\Delta t)^{-1}\mathbf{I} + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\mathbf{J}_e\tau) \end{bmatrix},$$

and where diag(•) indicates a diagonal matrix, with entries • on the diagonal. The first three rows in $\mathbf{A}^{\text{loc}}$ are the discretized version of (3.9), where the final column takes the discrete x-, y-, and z-derivatives of ϕ for the first, second, and third rows,

respectively, and the contributions from the HDG edge space come through the right-hand-side. The final row of $\mathbf{A}^{\text{loc}}$ discretizes (3.10), and its first three columns calculate the divergence of $\mathbf{q}$, whereas its last column contains the time derivative term, and the element-local edge contribution. Again, the HDG edge space contributions come through the right-hand-side, where the forcing terms are also located.

Each element in the mesh will have a unique $\mathbf{A}^{\text{loc}}$ with a unique $[\mathbf{A}^{\text{loc}}]^{-1}$. These matrices can either be built and inverted each time they are used, or the inverses could be stored for each element. In case of storing the matrices, we can write the inverse more compactly by substituting for the gradients, which are a function of ϕ and λ_ϕ, or $\mathbf{q}_{i,\phi} = \kappa \left(\text{diag}\left(\frac{\partial \xi_j}{\partial x_i}\right) \mathbf{D}_j - \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_i \mathbf{J}_e) \right) \phi + \kappa \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_i \mathbf{J}_e)\lambda_\phi$:

$$\mathbf{A}_\star^{\text{loc}} = \left[\begin{array}{c} -\sum_i \text{diag}\left(\frac{\partial \xi_k}{\partial x_i}\right) \mathbf{D}_k \kappa \left(\text{diag}\left(\frac{\partial \xi_j}{\partial x_i}\right) \mathbf{D}_j - \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_i \mathbf{J}_e) \right) + (a\Delta t)^{-1}\mathbf{I} + \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\mathbf{J}_e \tau) \end{array} \right], \tag{3.12}$$

$$\mathbf{A}_\star^{\text{loc}} \left[\begin{array}{c} \phi \end{array} \right] = \left[\begin{array}{c} F_\phi + \left(\mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\mathbf{J}_e \tau) + \sum_i \text{diag}\left(\frac{\partial \xi_k}{\partial x_i}\right) \mathbf{D}_k \kappa \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_i \mathbf{J}_e) \right) \lambda_\phi \end{array} \right], \tag{3.13}$$

and then recomputing $\mathbf{q}_\phi$ after solving for ϕ.

In either case, the next step is to create the discrete global matrix that solves for the boundary conditions λ_ϕ:

$$\left\langle [\![\mathbf{q}_\phi \cdot \hat{\mathbf{n}} - \tau(\phi - \lambda_\phi)]\!], \theta_\varepsilon \right\rangle_\varepsilon = \left\langle g_{N_\phi}, \theta_\varepsilon \right\rangle_\varepsilon . \tag{3.14}$$

The final step, once λ_ϕ is known, is to solve the local equations (3.11) again, with the boundary conditions and forcing terms substituted. This is not the only approach, but is amongst the least expensive.

Next we will describe a matrix-based solution scheme, followed by an iterative matrix-free approach.

Matrix-based solve: The matrix-based solution scheme requires the construction of a global matrix. Constructing the global matrix, $\mathbf{A}^{\text{glob}}$, is challenging, so we offer some guidance. Since the global (based on (3.14)) and local equations (3.11) are linear, we can solve them for decomposed right-hand-sides and sum the partial

solutions to obtain the full solution. The strategy, then, is to decompose the solution $\phi = \phi^F + \sum_i \phi^{\lambda_i}$ and $\mathbf{q}_\phi = \mathbf{q}_\phi^F + \sum_i \mathbf{q}_\phi^{\lambda_i}$ (Nguyen et al., 2009a), where the $\bullet^F$ portion solves the local equations with the known right-hand-side forcing

$$\mathbf{A}^{\text{loc}} \begin{bmatrix} \mathbf{q}_{0,\phi}^F \\ \mathbf{q}_{1,\phi}^F \\ \mathbf{q}_{2,\phi}^F \\ \phi^F \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ F_\phi \end{bmatrix}, \tag{3.15}$$

and the $\bullet^{\lambda_i}$ portion solves the local equations with the unknown forcing from the boundary conditions

$$\mathbf{A}^{\text{loc}} \begin{bmatrix} \mathbf{q}_{0,\phi}^{\lambda_i} \\ \mathbf{q}_{1,\phi}^{\lambda_i} \\ \mathbf{q}_{2,\phi}^{\lambda_i} \\ \phi^{\lambda_i} \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_0\mathbf{J}_e)\theta_{\varepsilon,i} \\ \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_1\mathbf{J}_e)\theta_{\varepsilon,i} \\ \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\hat{\mathbf{n}}_2\mathbf{J}_e)\theta_{\varepsilon,i} \\ \mathbf{J}^{-1}\mathbf{L}\,\text{diag}(\mathbf{J}_e\tau)\theta_{\varepsilon,i} \end{bmatrix}, \tag{3.16}$$

where $\theta_{\varepsilon,i}$ is the i^{th} basis function on $\partial K \cap \varepsilon$. In other words, (3.16) solves for the effect of the coefficient $\lambda_{\phi,i}$ on the solution of $\phi, \mathbf{q}_\phi$ (similar to a Green's function approach). The global equations then become:

$$\mathbf{A}_{ij}^{\text{glob}} = \left\langle \left[\!\left[\mathbf{q}_\phi^{\lambda_i} \cdot \hat{\mathbf{n}} - \tau(\phi^{\lambda_i} - \delta_{ij}\theta_{\varepsilon,i}) \right]\!\right], \theta_{\varepsilon,j} \right\rangle_\varepsilon \tag{3.17}$$

$$\mathbf{A}_{ij}^{\text{glob}} \lambda_{\phi,i} = \left\langle g_{N_\phi}, \theta_\varepsilon \right\rangle_\varepsilon - \left\langle \left[\!\left[\mathbf{q}_\phi^F \cdot \hat{\mathbf{n}} - \tau(\phi^F) \right]\!\right], \theta_{\varepsilon,j} \right\rangle_\varepsilon . \tag{3.18}$$

As a note, (3.18) can be simplified if conforming meshes with equal polynomial orders are used throughout. In that case, both sides of the equation are multiplied by the same mass-matrix, $\mathbf{M}_\varepsilon$, which requires a matrix-vector multiplication. This matrix-vector multiplication can be eliminated if both sides of the equation are multiplied by the inverse mass-matrix $\mathbf{M}_\varepsilon^{-1}$. In either case, the global matrix needs to be carefully constructed. In particular, we note that when $j \neq i$, the $\theta_{\varepsilon,i}$ term is zero. Also, constructing this matrix is expensive. The right-hand-side requires one inversion of

the local equations, but to build the matrix we have to invert the local equations once for every degree of freedom on the HDG edge-space, or $N_e \times N_{b,e}$ times. For example, a 2D triangle with a linear or quadratic polynomial requires 6 or 9 local solves, respectively. Thus, for a time-varying matrix, the actual construction of the matrix will be expensive. As such, matrix-free iterative schemes become more attractive. However, it is not immediately obvious how to construct a matrix-free version of the HDG method.

Iterative matrix-free solve: Iterative methods require the evaluation of the residual $\mathbf{F} - \mathbf{A}^{\text{glob}}\lambda_\phi$. The key to an iterative method is abandoning the Green's function approach. As such, first we need to invert the local equations (3.11) using the present iteration's guess for λ_ϕ to obtain the present guess values for ϕ and $\mathbf{q}_\phi$. These local solves could also be done iteratively, or the local matrix can be constructed and inverted. With all the local values known at an iteration, the residual of the flux-conservation equation (3.14) can be evaluated algebraically. In essence, this procedure gives the result of a matrix-vector multiplication of the global matrix with the present guess for $\mathbf{A}^{\text{glob}}\lambda_\phi$, which is what we desire. As such, the matrix-free method may be more efficient, since the global matrix $\mathbf{A}^{\text{glob}}$ never has to be created. The solution procedure is summarized in Fig. [3-4] for both the matrix-based and matrix-free iterative method. Note that either method can use the smaller $\mathbf{A}^{\text{loc}}$ matrix ((3.12)–(3.13)) instead.

We can relate the iterative procedure to the matrix-free implementation of LDG (for example Dedner et al. (2012)). In LDG, first the flux $\hat{\phi}$ is calculated based on the present iteration's guess, this requires non-local memory access (Fig. [3-5]). Second, $\mathbf{q}_\eta$ is calculated element-locally. Third, $\hat{\mathbf{q}}_\eta$ is calculated, again requiring non-local memory access. Fourth, ϕ is calculated for the present iteration, again using element-local operations. Last the new residual is calculated, and ϕ is updated for the next iteration of the scheme. In HDG, there are many element-local operations to invert $\mathbf{A}^{\text{loc}}$ and these are nearly the same as steps 2 and 3 of LDG if an iterative solver is used for the local operations. When the local variables have converged, global communication is necessary to calculate the update for λ_ϕ, and following this the

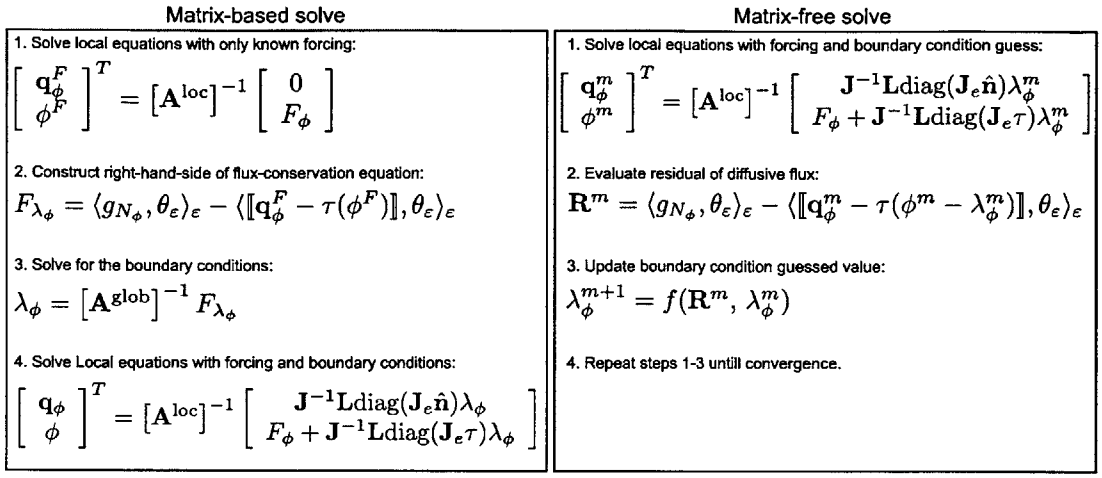


Figure 3-4: Summary of the matrix-based (left) and iterative matrix-free (right) solution methods of diffusion equations using HDG methods.

residual is calculated. Thus, for each LDG iteration, there are two operations using element-local memory access, and two operations using non-local memory access. For each HDG iteration, there are many operations using element-local memory access, and one operation using non-local memory access. Thus, HDG pushes more computation toward local memory access. If LDG and HDG converge in the same number of iterations and non-local memory access (i.e. global communication) dominates the cost of each iteration, then HDG will be more efficient. If the element local operations dominate the cost of each iteration, then LDG will be more efficient (because the HDG scheme requires more local computations per iteration). Whether non-local memory access or local operations dominate the cost of an iteration will depend on the particular computer architecture. Naturally, if one scheme consistently converges in fewer iterations, a clear choice may emerge. Hence, even though the matrix-free HDG solution method is complex, it looks computationally similar to existing methods.

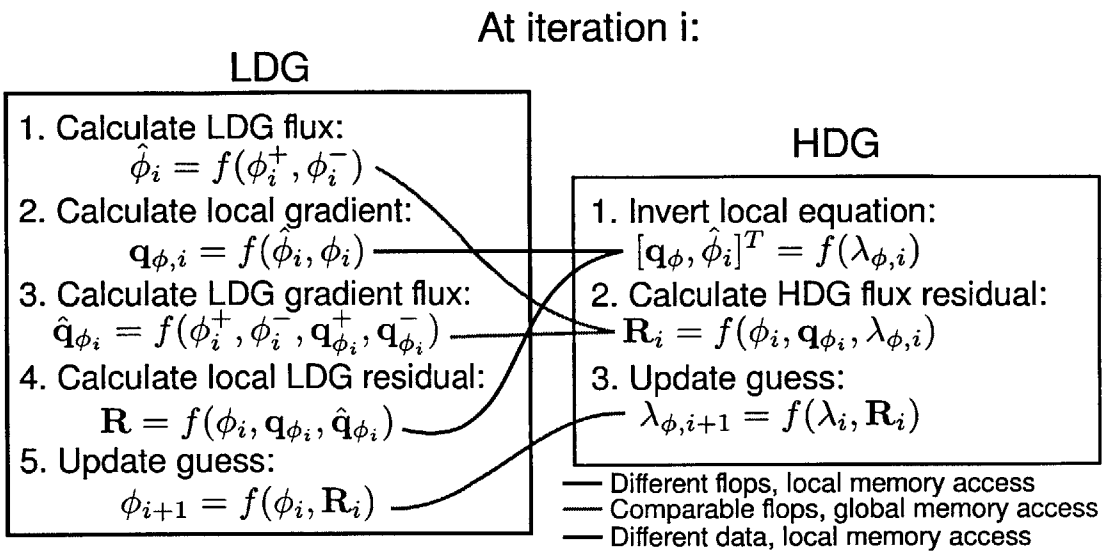


Figure 3-5: Comparison of matrix-free iterative schemes for LDG and HDG. (Note, the colors match those of Fig. [3-11]).

3.5 Dealing with Oscillations: Filtering and Limiting

The non-linear advection terms in the Navier-Stokes equations offer unique challenges. In particular, high order methods can develop non-physical oscillations that can lead to numerical instabilities. As such, any successful high-order numerical scheme has to deal with these oscillations. Applying a filter or a limiter are two approaches that can be used to suppress oscillations. A filter damps the modal polynomial coefficients according to an arbitrary spectrum, where higher-frequency modes are damped more. A limiter ensures that the solution remains within calculated bounds, usually determined from neighboring elements or nodes. While limiting traditionally focuses on reconstructing completely oscillation-free solutions, we are primarily interested in enhancing the stability of our numerical solutions. The selective filters and limiters that we have evaluated and developed for this purpose are described in this section.

Our initial selective filtering approach is described in detail in (Ueckermann and Lermusiaux, 2010). There we used an exponential filter (Hesthaven and Warburton,

2008, Hesthaven and Kirby, 2008), where the modal coefficients are modified using a function σ that decays exponentially with the polynomial degree (Mavriplis, 1989). The selectivity of the filter was determined by comparing the decay of the modal coefficient to a reference spectrum. In Ueckermann and Lermusiaux (2010), there was only a selective filter, while in this thesis, we have a new limiter and an improved selective filter. These limiters and filters are derived next. First we will describe our nodal limiting procedure, then our new selectivity criterion.

(i) Nodal Limiter: Our limiting procedure is based on previous nodal limiting methods used for second-order accurate methods (Hoteit et al., 2004). We have modified this method for high-order nodal DG. There are some drawbacks to our modifications, in particular our procedure is Total Variation Bounded (TVB) as opposed to TVD, and without the selectivity, it does not remain high-order accurate. However, it does successfully stabilize the numerical solution by suppressing spurious oscillations, and with the selectivity criterion the solution does remain high-order accurate.

Our procedure can be understood in 5 steps (Fig. [3-6]). The first step is to find the limits, or the initial total variation of the solution. That is, we determine the allowed maximum and minimum values for each element. Presently, this is done by finding the maximum and minimum values of the solution in the present and neighboring elements.

$$\phi_{\max} = \max(\phi)_{K^{\pm}}, \qquad \phi_{\min} = \min(\phi)_{K^{\pm}}, \tag{3.19}$$

where $K^{\pm}$ includes the element K, and all its immediate neighbors. Using the terminology in Hoteit et al. (2004), this is similar to choosing $\alpha = 1$. While our present step 1 may cause the nodal limiter to fail the Hoteit et al. (2004) "stair-step" numerical example, a small modification should guarantee the correct solution: if the maximum and minimum values of the function is determined solely by the present and upwind neighboring elements, then our nodal limiter should correctly solve the "stair-step" problem. Another possible modification is to determine the maximum

and minimum values for individual nodes by looking at the values of neighboring nodes. This reduces the allowed variation for each node, which could also improve accuracy. Nonetheless, our primary concern is stability, so these questions are left for future research. The present nodal limiter is efficient, simple to implement, and guarantees that the solution will remain bounded. For example, if the density is positive everywhere in the domain, these limits will never be negative. Once the bounds have been determined, the limiting procedure can continue.

The first step determined the limits or total variation of the field before evolving it in time, and this was the only operation that requires information from neighboring elements. The remaining steps are all element-local.

In the second step of the limiting procedure, the solution is evolved using the right-hand-side forcing without limiting

$$\bar{\phi}^{k+1} = \phi^k + \Delta t F_\phi. \tag{3.20}$$

The right-hand-side forcing terms, particularly the advection terms, can introduce oscillations. Thus, this new solution may exceed the limits calculated in step 1.

The third step, limits the nodal values. That is, we find nodes where the evolved solution exceeds the limits determined in step 1, and we calculate a forcing (which is the first predictor for the limiter forcing) $\bar{F}_\phi^{\text{limit}}$ that sets those nodes equal to the appropriate maximum or minimum values.

$$\bar{\bar{\phi}}^{k+1} = \phi^k + \Delta t F_\phi + \Delta t \bar{F}_\phi^{\text{limit}} \tag{3.21}$$

In this step, however, the mean of the initial solution in the element could be modified. As it is important to conserve mass, the mean in the element has to be re-adjusted. Thus, the change in the value of the mean in the element K caused by the adjustment is calculated

$$\Delta\text{mean}\left(\bar{\bar{\phi}}\right)_K = \text{mean}(\bar{F}_\phi^{\text{limit}})_K. \tag{3.22}$$

In other words, we want the final limiter forcing to have: $\text{mean}(F_\phi^{\text{limit}})_K = 0$.

The fourth step finds weights that determine by how much different nodes can move to help with the re-adjustment of the mean in the element. While previous researchers have developed sophisticated ways to minimize the error of this adjustment, here we use a heuristic approach. If the mean was lowered or raised, we calculate the maximum upward or downward adjustment allowed for each node, respectively, $F_{\text{max, adjust}}$. For example, if a node is already at the maximum value, it cannot be adjusted upwards, and will therefore have a zero effective weight. The nodes furthest away from the bounds will have the largest weight.

In the fifth and final step, we scale the maximum adjustment weight calculated in step four by the required amount to correct the mean of the limiter forcing. That is, we can now calculate the final limiter forcing

$$F_\phi^{\text{limit}} = \bar{F}_\phi^{\text{limit}} - \frac{\text{mean}(\bar{F}_\phi^{\text{limit}})_K}{\text{mean}(F_{\text{max, adjust}})_K} F_{\text{max, adjust}}, \tag{3.23}$$

where the sign of the final adjustment depends on the sign of the calculated maximum adjustment.

In the graphical representation of these steps Fig. [3-6], we illustrate a normal and degenerate case. For the normal case, note that while the solution is limited at the nodes, between the nodes the polynomial is allowed to exceed the bounds. The problem with the degenerate case is that the initial mean of the solution exceeds the calculated bounds. As such the final solution is a constant equal to the original mean; In other words, all oscillations are removed. This situation can occur for the advection operator if it is treated explicitly, and the Courant-Friedrichs-Lewy (CFL) condition is violated. In this situation, our limiting method can actually stabilize the solution and prevent instability if the CFL condition is not violated by too large a margin. The forcing due to implicit diffusion can also create a degenerate situation. In this case, the support of the diffusion operator is global, but the bounds for the nodal limiter are only calculated based on local neighbors. As such, the nodal limiter bounds cannot capture variation in the solution that may be valid. For example,

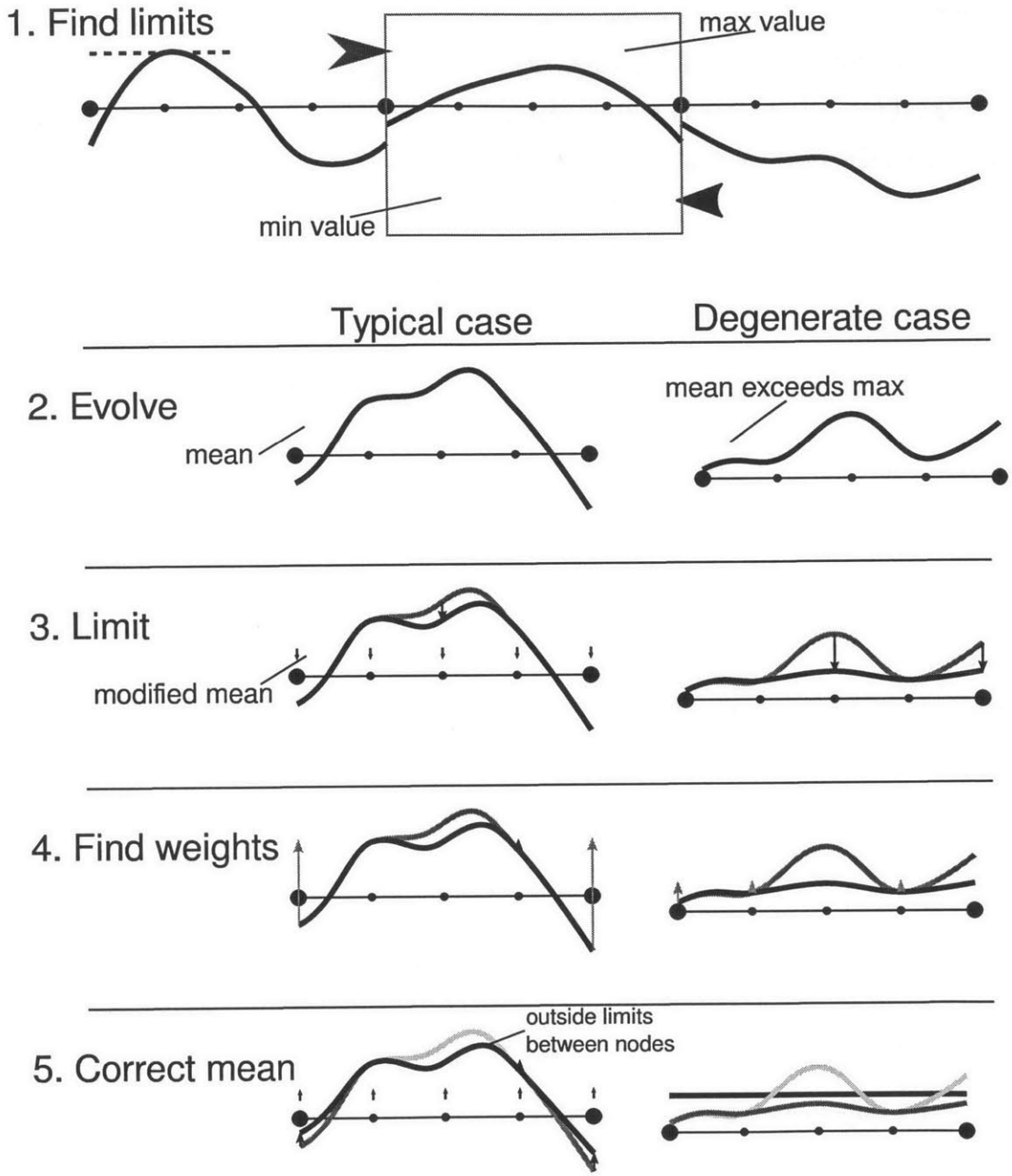


Figure 3-6: Graphical representation of the nodal limiting procedure from our high-order selective nodal limiter. The nodal limiting procedure is sketched for a typical and degenerate case in the left and right columns, respectively.

when diffusing a step function, the immediate neighbors will know about the step, but further away the bounds would suggest a constant solution. As the length-scale for the diffusion could extend beyond immediate neighbors, some bounds will not be correctly calculated. It is still important to limit the diffusive term, because if the length-scale of the diffusion is small, the step function can develop non-physical oscillations. This is because the small slope introduced by the diffusion may not be resolved by the discretization, resulting in Gibbs oscillations. This suggests that different bounds should be used for different parts of the forcing. In particular, we note that some forcing terms should not be limited, as they should introduce new variation in the solution. However, these forcing terms should be well-resolved, or appropriately smoothed to avoid oscillations.

(ii) Filter: We have described how to calculate the forcing term F_ϕ^{limit} that will appropriately limit the solution to be within the calculated bounds. A similar function for the filter can also be calculated, F_ϕ^{filter}. For details on the filter forcing, see Ueckermann and Lermusiaux (2010).

(iii) Selective limiting/filtering: At this stage both the nodal limiter and filter would be fully applied everywhere in the domain. What remains is finding an appropriate weighting function, $\alpha(\mathbf{x}, t)$, that selectively applies this forcing in parts of the domain where required. For this we use a discontinuity sensor similar to the one used by Huerta et al. (2012), which was proposed in Persson and Peraire (2006), Nguyen et al. (2007), where the difference is that we do not include the coefficient of the zero-degree polynomial in the denominator of the sensor (see below).

The discontinuity sensor works as follows. First, the coefficients of the nodal basis are transformed to modal-basis coefficients. To do so, we need to form the Vandermonde matrix $\mathcal{V}_{ij} = \theta_j^M(\mathbf{x}_i)$, where θ_j^M is the j^{th} modal polynomial, and $\mathbf{x}_i$ is the i^{th} nodal point. The modal coefficients can then be found as $\phi_i^M = \mathcal{V}^{-1}\phi_i$. We can then compare the decay of the modal coefficients to the decay of reference spectra.

To do so, we follow Huerta et al. (2012), and define the weight as:

$$\alpha^\star = \frac{1}{\beta^{\text{top}} - \beta^{\text{bot}}} \left\{ \log_{10} \left(\frac{\sum\limits_{\{i:\theta_i^M \in \mathcal{P}^{\mathrm{p}^\star}\}} (\phi_i^M)^2}{\sum\limits_{i>0} (\phi_i^M)^2} \right) - \beta^{\text{bot}} \right\}, \tag{3.24}$$

$$\alpha = \min(\max(\alpha^\star, 0), 1) \tag{3.25}$$

where $\mathrm{p}^\star \geq \mathrm{p}$, $\beta^\star = \log_{10}\left(\frac{\sum_{\{i:\theta_i^M \in \mathcal{P}^{\mathrm{p}^\star}\}} (\phi_i^{M,\star})^2}{\sum_{i>0} (\phi_i^{M,\star})^2}\right)$ and $\phi_i^{M,\star}$ are the modal coefficients for the $\star$ reference spectrum. What this indicator does is compare the sum of the coefficients for the highest degree polynomial to the sum of the coefficients for the polynomials of degree greater than zero. An notable difference between our indicator and that defined in Huerta et al. (2012) is that we do not include the coefficient of the zero-degree polynomial in the denominator. This is because the constant term can be arbitrarily scaled (based on non-dimensionalization, for example) and should not have an impact on the smoothness indicator. Note that α is time-variable, spatially variable, and a constant in every element. Two reference spectra and the ranges for α are sketched in Fig. [3-7]. Depending on the smoothness of the modal coefficients of the numerical solution, the weights for α can be anywhere in [0, 1]. Our approach is different from that of Huerta et al. (2012) as we do not use the Mach number for the discontinuity sensor, and we do not decompose the high-order element into low-order sub-domains.

Note that we calculate the selectivity index by examining the field before the advection term is added (that is, at the start of the IMEX-RK stage). At each subsequent stage in the IMEX-RK time-stepping procedure, the selectivity index is updated. Alternatively, we could calculate the selectivity index after the advection is added (that is, advanced in time due to advection at that IMEX-RK stage). Another option is calculating the smoothness index based on the advection term itself, or any combination of the above-mentioned options. Additionally, to increase efficiency, the selectivity index could only be updated at the first or final stage of the IMEX-RK time-stepping procedure. However, we do not examine the effect of these choices.

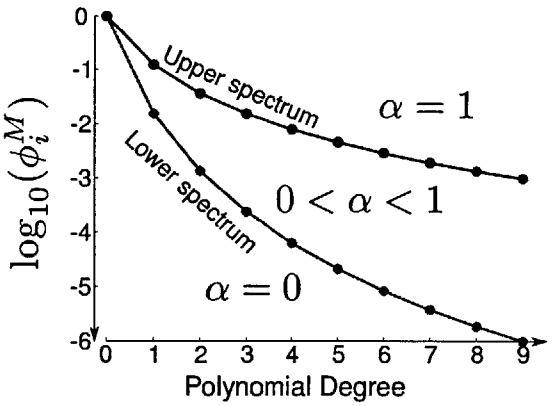


Figure 3-7: Sketch of the selectivity criterion. The solution is fully limited, partially limited, or unmodified if the modal-polynomial-coefficient-decay is slower than the top reference spectrum $\log_{10}([\mathrm{p}+1]^{-3})$, between the two reference spectra, and faster than the bottom spectrum $\log_{10}([\mathrm{p}+1]^{-6})$, respectively.

Finally, the selective nodal limiter can be applied as a weighted forcing term, which gives the final evolution of ϕ as:

$$\phi^{k+1} = \phi^k + \Delta t F_\phi + \Delta t \alpha^s F_\phi^{\text{limit}}, \tag{3.26}$$

where $s = 0$ gives a non-selective nodal limiter, and $s = 1$ gives a linear weighting between the reference spectra. For $s > 1$, the solution is weakly limited close to the lower spectrum, and for $0 < s < 1$ the weight quickly increases. The same approach can be used for the forcing calculated from the filter.

In summary, we have derived a high-order selective nodal-limiting/filtering procedure. Our selective nodal limiter is based on existing nodal-based limiters, but we have extended it higher-order polynomials, and we use an inexpensive heuristic to ensure mass conservation. We combined this nodal limiter with a smoothness indicated to selectively limit the solution spatially and temporally. The selectivity criterion could also be applied to a filter, instead of a limiter. This selective nodal limiter is tested in §3.7, to ensure that high-order convergence is recovered when the solution is sufficiently resolved.

3.6 Code design philosophy and development using Python/C++

In this section we justify why we developed our finite element framework in Python, for later optimization in C++. We also briefly outline the organization of our code, explaining the design purpose of the most important modules and classes.

First, why develop a scientific computing code using a scripting language? With the increase in computational power, and due to the rapid spread of knowledge (through the Internet) there has been a shift in programming practices (Langtangen, 2009). No longer is it necessary for researchers to spend their time meticulously writing low-level code in languages like C++ or Fortran just to test a new algorithm or idea. With the speed of current processors, even a poorly optimized code will execute fast enough to evaluate the merits of a new idea. The near-ubiquitous usage of MATLAB in the scientific community strongly supports this claim. Instead, the focus has shifted to rapid development and flexibility; where computational power once was the limiting resource, now shortening development/research time is the primary concern.

While low-level solver optimization will always have its place in scientific computing, researchers can benefit from developing portions of their code in higher-level scripting languages. In large-scale scientific computing, computational resources will always remain a restriction, but the computationally-expensive solver-portion is not the only component of the code. Built around this solver are input, output, setup, plotting, formatting, and reporting functionality. These portions are not limiting in terms of computational time, and could save significant developer and user time if written in high-level languages. Additionally, during algorithm development, the solver-portion of the code is frequently updated, tested, and debugged using small benchmarks. During development, significant developer time can be saved from debugging grammatical and logical mistakes if using a high-level language with benefits like interactive plotting. As such, even large-scale scientific computing project can benefit from high-level scripting languages.

Thus far we have implied that high-level languages are less computationally efficient than their low-level counterparts. While it is true that a new specialized algorithm implemented in high-level languages, such as Python, may be an order of magnitude slower than its C++ counterpart, many established algorithms have already been implemented in Python. Or, more accurately, low-level optimized implementations of established algorithms have been interfaced with Python. A lone developer working in C++ would be hard-pressed to match the efficiency of the functions available in Python. While the C++ developer can resort to libraries, we have found that the Python packages are generally simpler to install and interface with our codes. As such, from both a computational and developer efficiency, Python has significant benefits.

Our strategy, then, has been to develop the numerical algorithms and schemes entirely in Python. We chose Python because it is open-source, and was written specifically with C++ extensions as part of the design (Langtangen, 2009). Once the development was complete, the next step in our strategy was to optimize the solver using C++. Functionality such as the setup, inputs, and outputs remains in Python, and these components interface with the C++ portion using a common binary file format.

Second, we briefly outline the important portions of our Python framework. In Python terminology, modules are a collection of files containing function, class, and variable definitions. We designed our modules to have specialized purposes, and when grouped together they constitute our 3DDG "package." The different objectives from the major modules as they are used in our program flow are shown in Fig. [3-8]. Next, we describe the design purpose of each of these modules and summarize how they work together. Finally, we delve a little deeper into the code, and explain some of the important methods and members of the most important objects.

msh: The `msh` module contains all class and function definitions related to the creation, storage, and manipulation of the underlying finite element mesh. This includes creating the connectivity information. The primary objective of this class is to create and manipulate one of the `Mesh` objects. As of writing this, there are

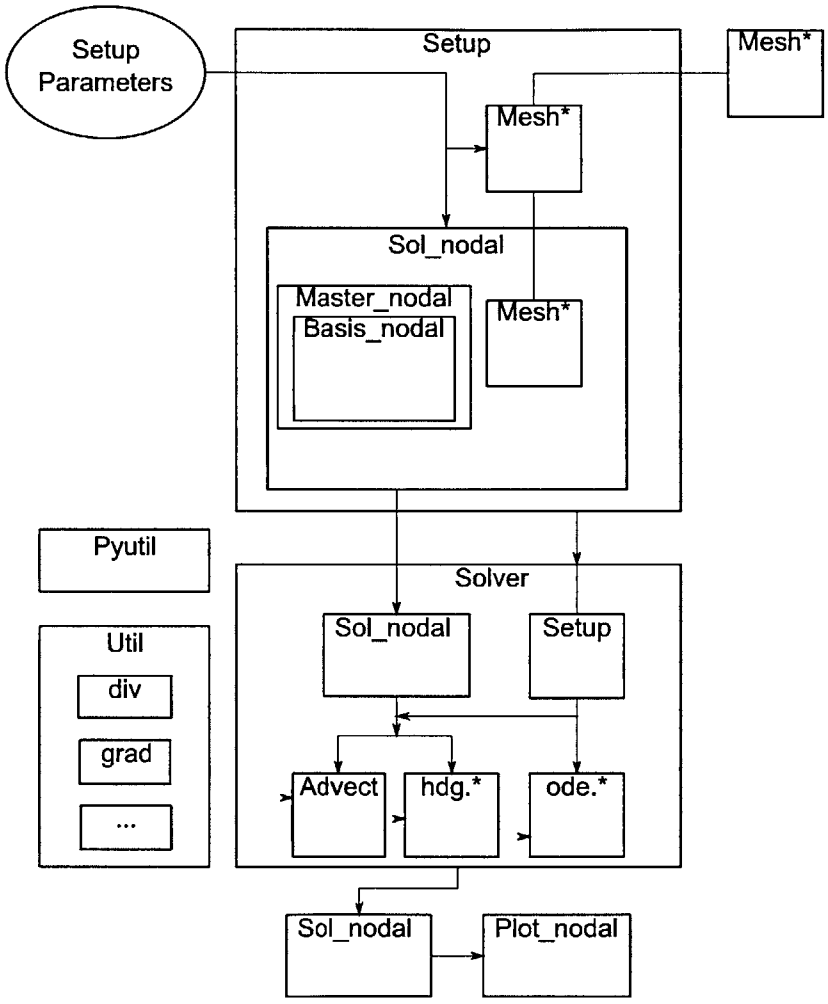


Figure 3-8: Program flow diagram with major objects from python modules. Objects instantiated inside other objects are contained within their borders. Here * is used as a wild card, indicating various possible choices.

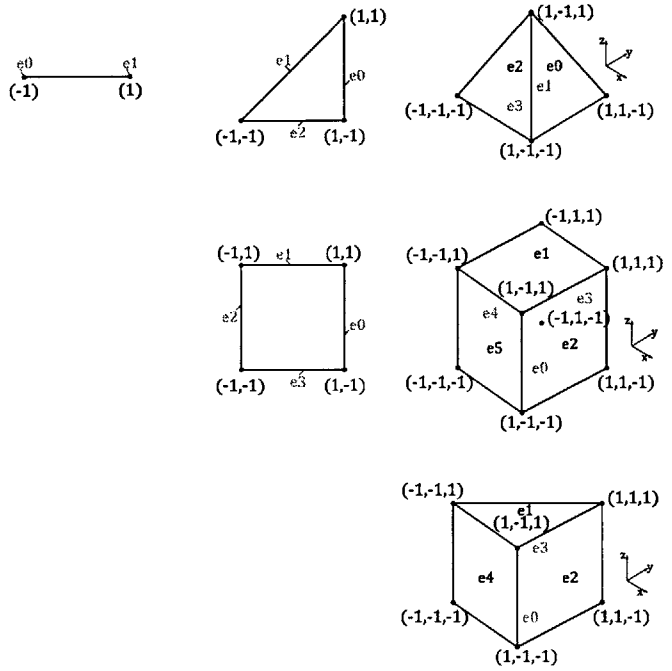


Figure 3-9: The various different element types supported by the master module. In the code, simplexes, quadrilaterals, and prisms are of types 0, 1, and 2 respectively.

three mesh classes, one for 2D, 3D, and 3D extruded meshes. Note, this module is not specific to any type of finite element method, and does not depend on the degree of polynomial.

master: The `master` module deals entirely with the creation, evaluation, computation, and storage of polynomials on the reference or master element. It also contains functionality to calculate appropriate coordinate transformation factors to deal with real-space calculations (see §3.2). The two major classes in this module are named `Master_nodal` and `Basis_nodal`. The `Basis_nodal` class deals only with the creation, evaluation, and storage of polynomials on reference elements. As such, the numbering conventions associated with the various different elements are defined in this class Fig. [3-9]. Since the finite element method does not usually deal directly with the polynomial bases, the `Basis_nodal` object is contained within the `Master_nodal` object (i.e. it is a member of `Master_nodal`). The `Master_nodal` object calculates commonly-used finite element operators and stores them. Finite element operators then directly interface with the `Master_nodal` objects.

sol: The `sol` module ties together the `master` and `mesh` modules. It contains a number of sub-modules and classes. One of these, the `Sol_nodal` class, contains the solution vectors, and merges the functionality of the `Master_nodal` and `Mesh` classes. For example, the `Sol_nodal` class handles the projection of polynomials (defined inside `Master_nodal`) on the element to the edges (where the connectivity is defined inside `Mesh`). Within the `sol` module, there are also specific modules related to specific differential operators. For example, the `advect` module handles advection, the `hdg` module handles HDG operators (such as diffusion), and the `ode` module implements various ordinary differential equation solvers, such as Runge–Kutta. Finally, the `fluids` module contains a `Setup` class and `Solver` class for fluids problems. The `sol` module, then, acts as a high-level interface between lower-level operations. This allows equations to be built using high-level abstraction, without the need for delving into low-level integration.

plot: The `plot` module is a high-level interface for some specialized plots. Presently, the `sol.plot_nodal.Sol_nodal` object is created using a `sol` object as one of its inputs, and specialized plots are then called using fields from `sol` as inputs.

util: The `util` module contains various finite element helper functions and classes. In particular, the `operators` sub-module handles divergence, gradient, z-integral, and other low-level operations. These would be utilized by, for example, the `sol.advect` sub-module. Additionally, `util`'s `filter` sub-module handles filtering and limiting of the solution (§3.5).

pyutil: The `pyutil` module is similar to the `util` module, but here the helper functions are not specific to finite elements. That is, these functions and classes relate to general programming problems, such as finding the unique elements in a list, parsing string inputs, and writing outputs.

The general program flow in Fig. [3-8] requires the user to interact primarily with a `Setup` object to define a mesh and various parameters. Using this information, the `Setup` object builds a `Sol_nodal` object, which gets passed to a `Solver` object. The `Sol_nodal` object internally builds `Master_nodal` objects, which build `Basis_nodal` objects. Continuing, the `Solver` object uses the parameters from the `Setup` object

to build the appropriate finite element differential operators. Calling the `solve()` routine from the `Solver` object will integrate the solution in time or solve the steady state problem, and return the final answer as part of a `Sol_nodal` object.

Delving deeper, Fig. [3-10] shows the important members (i.e. variables) and methods (i.e. functions) for the major objects. The user interacts with the `Setup` object primarily through a series of `set*()` methods, where the * is a used as a wild card in this case. The `Setup` object informs the user using a series of `print*()` methods. The mesh and parameters (`*params`) set by the user will be stored in the `Setup` object to be used by the `Solver` object. After setting the solver options, `Setup` will internally call the `Setup.check_solver_options()` method, which ensures that the user has selected compatible solver flags. Once the `Setup` object is complete, the user has to call the `Setup.validate()` method, which internally calls `Setup.check_solver_options()`, and also ensures that all the required parameters are set by the user. Any errors or warnings will be reported to the user at this stage. Finally, the user has to call `Setup.make_sol()`, which creates the solution data structure.

Next, the `Solver` object is initialized using the `Setup` object. `Solver.__init__()` again calls the `Setup.validate()` method to ensure that the setup is valid. Based on the parameters in `Setup`, `Solver.__init__()` also creates the necessary objects to handle different differential operators. Following this, the user invokes the `Solver.solve()` method, which will solve the specified problem. Internally, the `Solver` object passes itself to the `time_integrate` object, which in Fig. [3-10] is an instantiation of the `IMEX` class. The `IMEX` class is the most general ordinary differential equation solver, since it requires both implicit and explicit function evaluation. Therefore it interfaces with the `Solver` object through the `save0`, `F_ex`, `F_im_ex`, `F_im`, and `update` methods. The primary purpose of the `time_integrate` object is the combine the right-hand-sides of the ordinary differential equation at different time levels, `time_integrate.mk_rhs()`, to yield a higher-order accurate solution. Internally, the important data-structures are: the right-hand-side `rhs` at the present stage; the explicit function evaluations `f_ex` for all stages; the implicit function evaluations `f_im` for all stages; and the

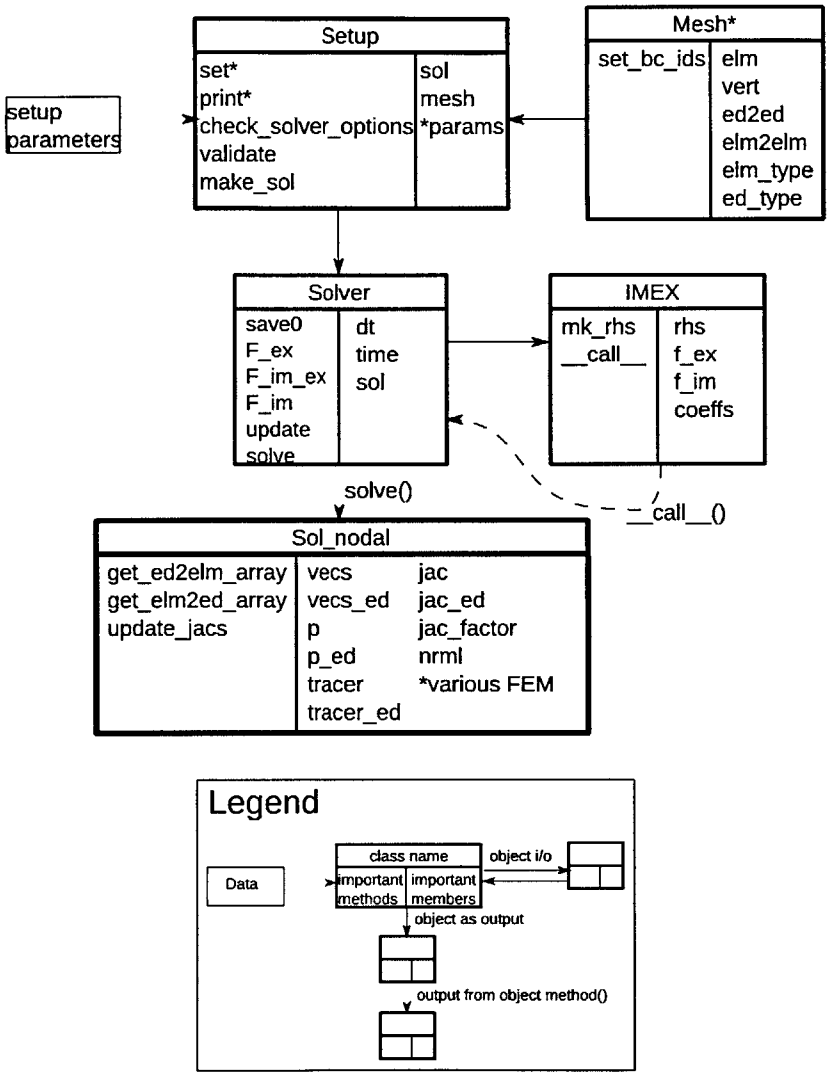


Figure 3-10: The major classes from Fig. [3-8], but with more details, giving the important members (variables) and methods (functions). Here '*' is used as a wild-card for various similar constructs.

Runge–Kutta coefficients `coeffs`. The `time_integrate.__call__()` method returns the `Solver` object. The `Solver.solve()` method returns the `Sol_nodal` object to the user.

The `Sol_nodal` object acts as a high-level interface between the `Mesh` and `Master` objects. Its most important methods `get_elm2ed_array()`, `get_ed2elm_array()`, and `update_jacs()` projects the element polynomials on to the edge-space, distributes the edge-space polynomials to the `ed2elm`-space, and updates the coordinate transformation factors, respectively. In the code, there are three major storage array structures: the `elm`-space array, the `ed`-space array, and the `ed2elm`-space array (Fig. [3-11]). The first two are convenient for storing the solution on the element-space and the HDG edge-space, respectively. The third, `ed2elm`-space is a convenient intermediate array used to calculate the edge-integrals on the element-space, for example $\langle \phi, \theta \rangle_{\partial K}$. The `Sol_nodal` object handles these transformations because it has the mesh connectivity information from the `Mesh` object and the details of the polynomials on the reference elements from the `Master` object. Apart from these important functions, the `Sol_nodal` object also stores the solution data on the elements and edges, which gets manipulated by the methods mentioned above. Additionally, it stores the coordinate transformation factors, and, for convenience, it also stores some specialized finite element matrices for the reference element (such $\mathbf{M}$, $\mathbf{D}$ and $\mathbf{L}$ from §3.3).

This concludes the philosophy and justification behind our code design, including the brief description of our Python Framework.

3.7 Verification of HDG diffusion and selectively-limited advection

Verification of a new code is a necessary to ensure that it solves the intended equations (Oreskes et al., 1994, Roache, 1998). To verify that the proposed schemes work, we perform convergence studies on simple equations. To show that our quadrature-

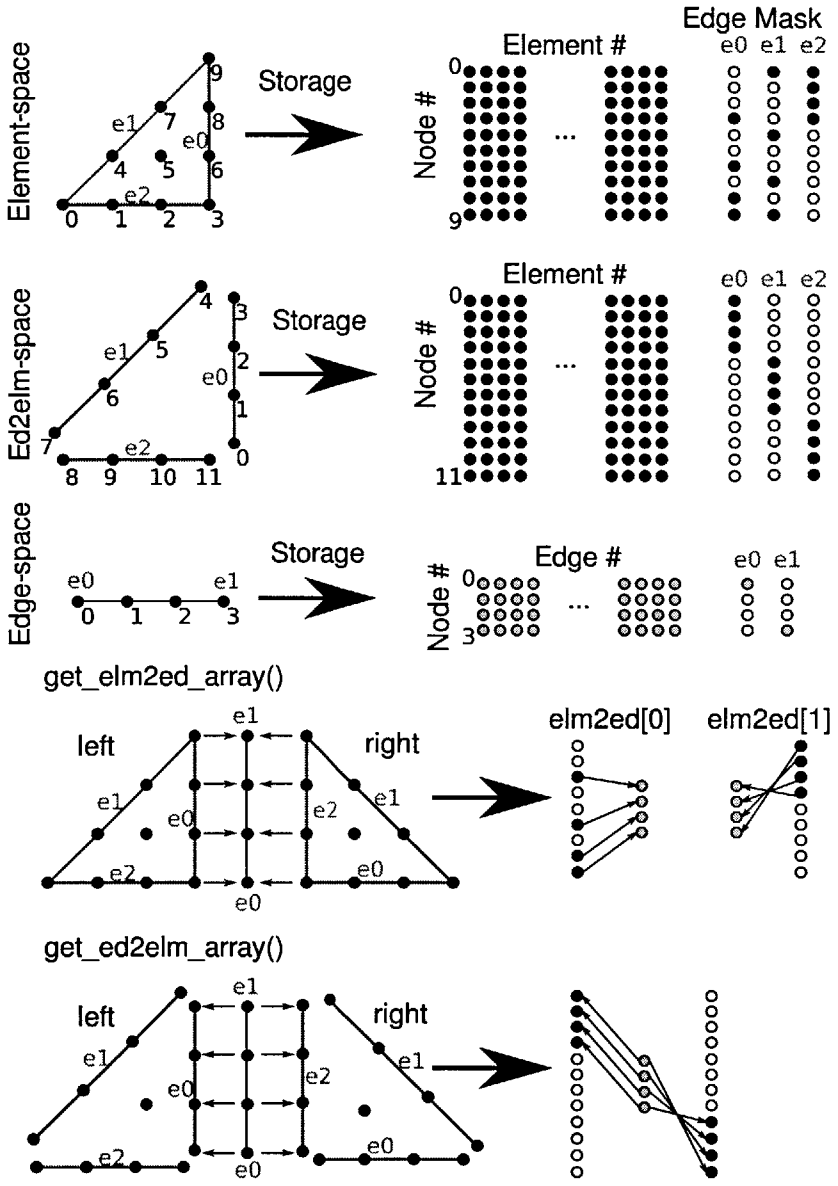


Figure 3-11: The three important storage array structures in the Python code and how they are related through methods in `Sol_nodal`. This is a particular example for a 2D triangle of p = 3. For mixed element types, the array structures are more complex. Note that `get_elm2ed_array()` gives two edge arrays, one for data from the left element `elm2ed[0]`, and one for data from the right element `elm2ed[1]`.

free scheme works for HDG schemes, we perform a convergence study on straight and curved meshes. Then we verify that the selective nodal limiter recovers high-order convergence rates when the solution is adequately resolved.

3.7.1 Verification of quadrature-free hybrid discontinuous Galerkin scheme

To verify that our HDG implementation works on curved meshes, we perform a convergence study on a steady diffusion problem

$$\nabla^2 \phi = f \quad \text{on} \quad \Omega, \tag{3.27}$$

$$\phi = g_D \quad \text{on} \quad \partial\Omega_D, \tag{3.28}$$

$$(\nabla\phi) \cdot \hat{\mathbf{n}} = g_N \quad \text{on} \quad \partial\Omega_N, \tag{3.29}$$

where

$$f = \sin(\pi(x + x_0)) \sin(\pi(y + y_0)), \tag{3.30}$$

$[x_0, y_0] = [0.3, 0.3]$, and the bottom and right boundaries are Dirichlet ($\partial\Omega_D$), while the top and left boundaries are Neumann ($\partial\Omega_N$) on the domain $\Omega \in [-1, 1] \times [-1, 1]$. We use both straight and curved meshes. The curved mesh for $\Delta x = 0.5$, p = 4 is shown in Fig. [3-12], and it is made up of a mixture of triangular and rectangular elements. We also perform the convergence study for two different values of the HDG stability parameter $\tau = [1, 1000]$.

Both the straight-sided and curved mesh simulations converge near-optimally for both values of τ (Fig. [3-13]). The error level is generally lower for the straight-sided mesh. While the error levels are generally similar for the different values of τ, the p = 2 result using $\tau = 1000$ seems to converge faster for both meshes. However, the p = 5 result using $\tau = 1000$ suggests the larger value of τ reaches machine precision earlier, possibly due to a larger condition number in the matrix.

These results verify that our scheme works for straight and curved meshes with

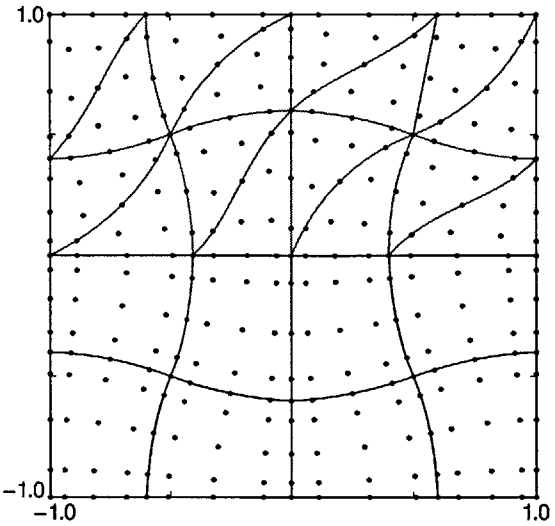


Figure 3-12: Curved mesh for the $\Delta x = 0.5$, p = 4 simulation. Triangular and rectangular elements are colored green and blue, respectively.

mixes element types.

3.7.2 Verification of selective nodal limiter

In §3.5 we developed a selective nodal limiter and filter, and here we test the effect of the selective nodal limiter. To do so, we study a modification of the swirl problem in chapter 5 of Durran (1999). We do not include results for the exponential filter because the nodal limiter is more robust, accurate, and does not require tuning. We solve the unsteady advection problem

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{v}\phi) = 0 \quad \text{on} \quad \Omega,$$
$$\phi = 0 \quad \text{on} \quad \partial\Omega_D,$$

on the domain $\Omega = [0,\, 1] \times [0,\, 1]$ with Dirichlet boundary conditions everywhere, over the time interval $T = [0, 10]$. The time-varying velocity is specified as

$$\mathbf{v} = \sin\left(\frac{\pi}{5}t\right)\left[\frac{1}{2}\sin(2\pi y)\sin^2(\pi x),\, -\frac{1}{2}\sin(2\pi x)\sin^2(\pi y)\right],$$

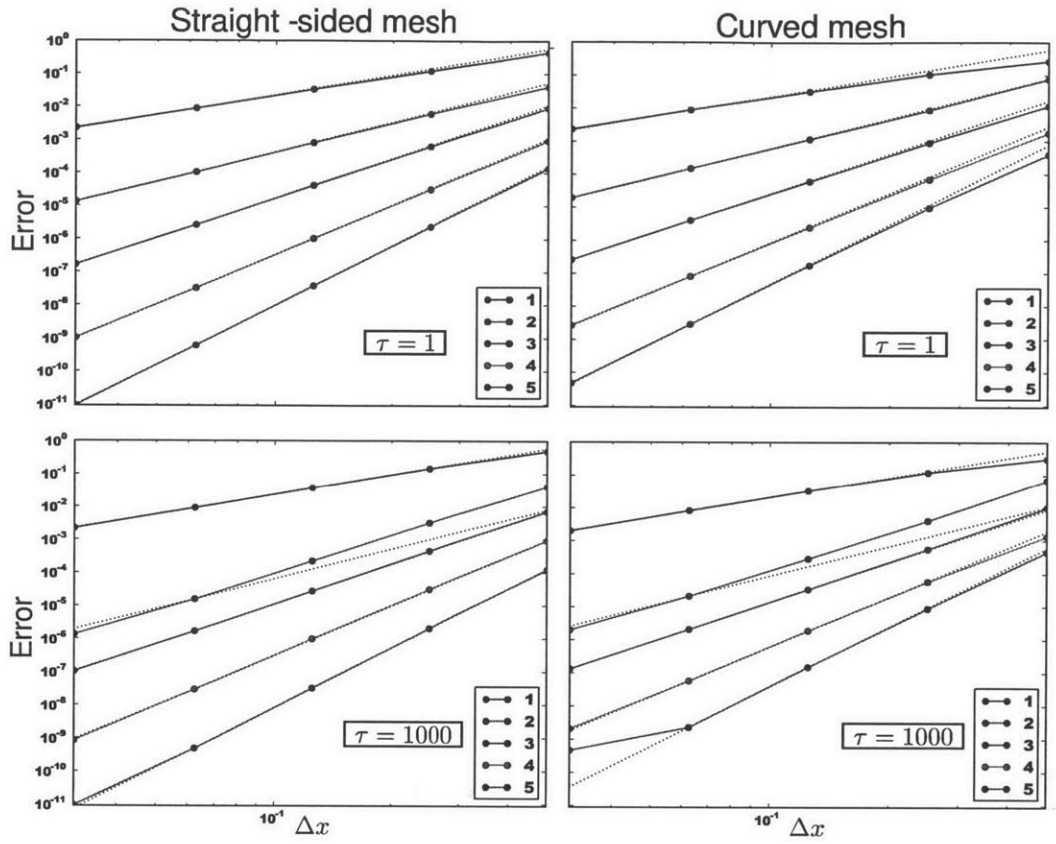


Figure 3-13: Spatial convergence of diffusion straight (left) and curved (right) meshes.

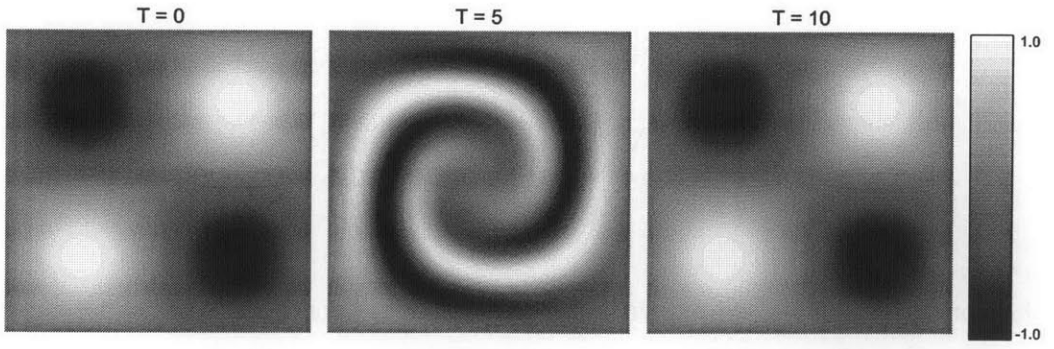


Figure 3-14: Tracer concentration at $T = [0, 5, 10]$ (left, center, and right, respectively) for the advection benchmark using p = 5, $\Delta x = \frac{1}{64}$.

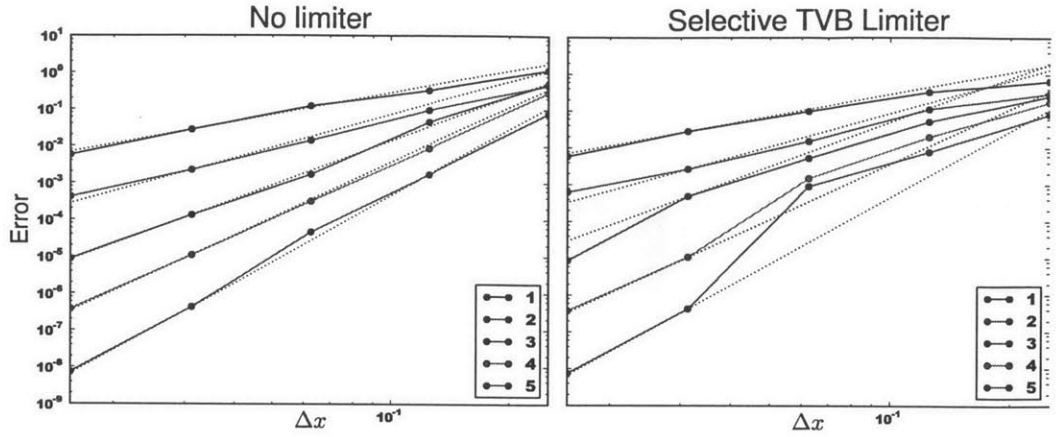


Figure 3-15: Spatial convergence of advection equation without limiter (left) and with selective nodal limiter (right). The spatial resolutions used are $\Delta x = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$.

and to have a smooth solution with positive and negative values for the convergence test, we have modified the initial condition for the tracer from Durran (1999) as

$$\phi(\mathbf{x},\, t = 0) = \sin(2\pi x)\sin(2\pi y).$$

The specified flow field causes the initial tracer concentration to swirl during the interval $T = [0, 5]$ (Fig. [3-14]). In the interval $T = [5, 10]$ the flowfield reverses direction, causing the tracer to "un-swirl." Thus, the final tracer concentration should be the same as the initial tracer concentration. Using this property, we can compute the error by comparing the initial $\phi(\mathbf{x}, t = 0)$ and final $\phi(\mathbf{x}, t = 10)$ fields.

For these simulations, we used a fixed time-step $\Delta t = 10^{-3}$, and a second-order accurate explicit RK time-integrator (with the same coefficients as the IMEX-RK integrators used later). The mesh is composed of uniform quadrilateral elements. The selectivity index, (3.25), uses $(\mathrm{p}+1)^{-3}$ and $(\mathrm{p}+1)^{-6}$ for the top and bottom reference spectra, respectively.

The simulations without the selective nodal limiter converge near-optimally Fig. [3-15]. However, for too coarse spatial discretizations, the higher order ($\mathrm{p} > 1$) simulations with the selective nodal limiter applied reduce to second-order accuracy. This is because the limiter is being fully applied at these resolutions, causing the tops of the sinusoidal tracer concentrations to be chopped off (Fig. [3-16]). Once the

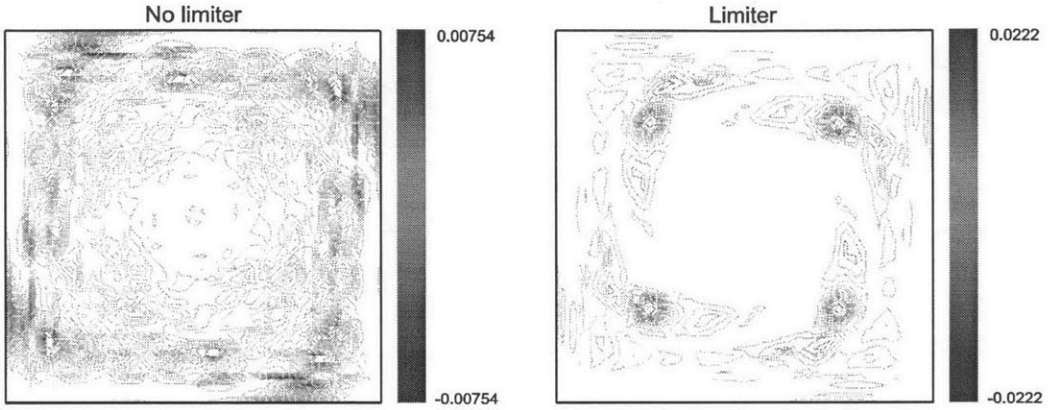


Figure 3-16: Errors of the tracer advection test-case for the intermediate resolution, p = 3, $\Delta x = \frac{1}{16}$, case in Fig. [3-15], without the limiter (left) and with the selective nodal limiter (right).

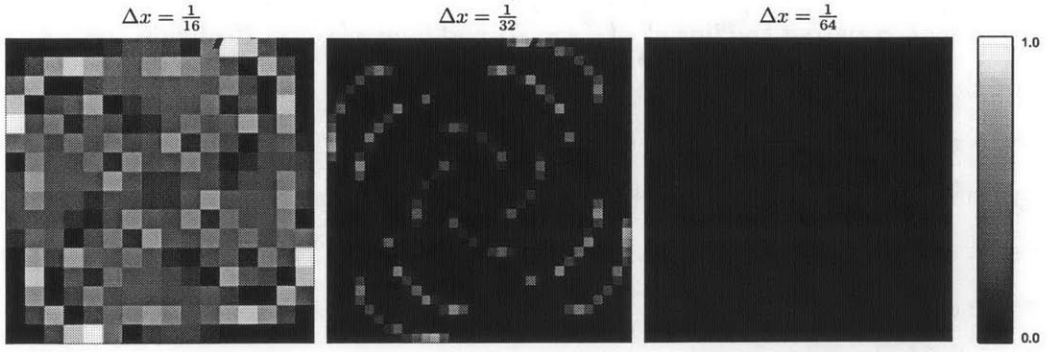


Figure 3-17: Selectivity index α (3.25) for the tracer advection test-case using p = 3 at resolutions of $\Delta x = [\frac{1}{16}, \frac{1}{32}, \frac{1}{64}]$ on the left, center, and right, respectively.

mesh is sufficiently refined, the effect of the nodal limiter is reduced by our selectivity criterion, and the higher-order convergence rate is recovered. If we examine the value of the selectivity index at $T = 5$ for the p = 3 case (Fig. [3-17]), we see that with increasing resolution the selectivity index α takes a smaller value, and is localized in space near sharp gradients. Recall, the nodal limiter is fully applied for $\alpha = 1$, and not applied at $\alpha = 0$. Therefore, at these higher resolutions, the selective nodal limiter is only active in localized regions throughout the domain. This allows the higher-order rates of convergence to be recovered at high resolutions (seeFig. [3-15]).

We have performed additional convergence tests for the advection which are not reported here, but gave similar results. Overall, these results verify both that our quadrature-free advection scheme is properly implemented, and that our selective nodal limiter can recover higher-order accuracy when the solution is sufficiently resolved.

3.8 Verification and Validation of Stokes/Navier-Stokes system of equations

In this section we verify and validate our new algorithm derived in §2.3.2. We first examine if the important theoretical properties from §2.2.2 hold for the numerical operators. Following this we perform a detailed convergence study using a manufactured solution. And finally, we solve the lock-exchange problem, comparing our density contours and Froude numbers to existing literature.

3.8.1 Definition of Analytical Benchmark

To evaluate the implementation of our new scheme, we will use the same manufactured benchmark used by Guermond et al. (2006). For this case, consider a domain

$\Omega \times [0, T]$ where $\Omega = [-1, 1] \times [-1, 1]$. The solution $[\mathbf{v}, p]$ is defined as

$$\mathbf{v}(x, y, t) = \pi \sin(t) \left[\sin(2\pi y) \sin^2(\pi x), -\sin(2\pi x) \sin^2(\pi y)\right], \tag{3.31}$$

$$p(x, y, t) = \sin(t) \cos(\pi x) \sin(\pi y). \tag{3.32}$$

From these definitions, we can calculate the forcing term $\mathbf{F}_{\partial t}$, which is

$$\begin{aligned}
\mathbf{F}_{\partial t} &= \frac{\partial \mathbf{v}}{\partial t} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla \mathbf{v} + \nabla p, \\
F^u_{\partial t} &= \pi \cos(t) \sin(2\pi y) \sin^2(\pi x) - \frac{2\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi y) \cos^2(\pi x) \\
&\quad + \frac{6\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi y) \sin^2(\pi x) - \pi \sin(t) \sin(\pi x) \sin(\pi y), \\
F^v_{\partial t} &= -\pi \cos(t) \sin(2\pi x) \sin^2(\pi y) + \frac{2\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi x) \cos^2(\pi y) \\
&\quad - \frac{6\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi x) \sin^2(\pi y) + \pi \sin(t) \cos(\pi x) \cos(\pi y).
\end{aligned} \tag{3.33}$$

This gives us a smooth analytical solution with which we can verify the spatial and temporal convergence of our scheme.

3.8.2 Numerical operator properties

In this section we evaluate whether the discrete operators maintain the two important properties introduced in §2.2.2.

To test the first property, whether $p = 0$ if $\nabla \cdot \mathbf{F}_{\partial t} = 0$, we modify the analytical benchmark by setting $p = 0$. This results in the following forcing function:

$$\begin{aligned}
F^u_{\partial t} &= \pi \cos(t) \sin(2\pi y) \sin^2(\pi x) - \frac{2\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi y) \cos^2(\pi x) \\
&\quad + \frac{6\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi y) \sin^2(\pi x), \\
F^v_{\partial t} &= -\pi \cos(t) \sin(2\pi x) \sin^2(\pi y) + \frac{2\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi x) \cos^2(\pi y) \\
&\quad - \frac{6\pi^3}{\mathrm{Re}} \sin(t) \sin(2\pi x) \sin^2(\pi y).
\end{aligned}$$

We use a fine mesh 16×16 elements with a p = 6 degree polynomial but a large time-

step of $\Delta t = 0.1$. Since we will be using a first-order time integration scheme for these tests, that means the truncation error will be of $\mathcal{O}(\Delta t) \sim 0.1$. In each case we use a zero pressure as the predictor in order to highlight the numerical errors. As such, the only difference between the pressure and the pressure-correction is the rotational term, that is $p = \delta p - \frac{1}{\mathrm{Re}} \nabla \cdot \bar{\mathbf{v}}$. We examine the effect of using different boundary conditions (see §2.2.3). We will use the standard boundary conditions proposed by Timmermans et al. (1996), as well as the new boundary conditions suggested by Shirokoff and Rosales (2011). The velocity and pressure plots in this section are scaled such that the correct solution will have a magnitude of 1. That is, we plot $\frac{u}{\pi \sin(t)}$ and $\frac{p}{\cos(t)}$. The divergence is scaled as $\frac{\nabla \cdot \bar{\mathbf{v}}}{\Delta t \cos(t)}$, which compares the magnitude of the pressure to the forcing term in the pressure Poisson equation.

For either choice of boundary condition, clearly, $p \neq 0$ (Fig. [3-18] and Fig. [3-19]). Therefore, the time-split solution does not obey the first important property. First, we note that the splitting-error in p is of the order of the truncation error $\mathcal{O}(\Delta t) = 0.1$. Even though this is re-assuring, we would like to understand the reasons why the time-split solution fails to satisfy $p = 0$ exactly. These reasons are now explained intuitively. Consider a one-dimensional slice through the u-velocity component (Fig. [3-20]). If we diffuse this solution for one time-step using the standard boundary conditions, we obtain the green solid curve. Because we fixed the u-velocity component to the boundary, the slope $\frac{\partial \mathbf{v}}{\partial \hat{\mathbf{n}}} = \frac{\partial u}{\partial x}$ is modified. Since the slope of u is coupled to the slope of v through the continuity equation, $\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}$, any modification in the slope of u requires a corresponding change in the slope of v. The decoupled Stokes equations (in this case, since $p = 0$, only diffusion and rates of change are active) solved in the first step of the projection method cannot communicate this change in slope. As such, the predictor velocities are divergent even though a non-divergent forcing function was used (Fig. [3-18]).

The boundary conditions of Shirokoff and Rosales (2011) give the solution sketched as the dashed blue curve in Fig. [3-20]. Here the v-velocity component is set to 0 at the boundary, as such the correct boundary condition for the u-component is specified as $\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y} = 0$. Thus, these boundary conditions will satisfy the continuity constraint,

and we see that the predictor velocity is divergence-free (Fig. [3-19]). However, after the diffusion step, there is no guarantee that $u = 0$ at the boundaries, and this needs to be corrected by the pressure. Thus, in this case, the pressure signal comes directly from a boundary value problem with no internal forcing. Using these boundary conditions, we know that the resulting signal should not be part of the pressure. Knowing this, we could actually obtain the correct pressure for this test.

In both cases the rotational correction increases the erroneous pressure signal (Fig. [3-18] - Fig. [3-19]). The purpose of this term is to eliminate the diffusion of the divergent parts of the right-hand side. In this case, the right-hand side is divergence-free, and should not require the rotational correction. However, since the pressure-correction is non-zero, the scheme will apply a correction, which in this case, decreases the accuracy of the pressure. The error in the pressure should be $\mathcal{O}\left(\frac{\Delta t}{\mathrm{Re}}\right)$, since we've argued that it comes from the boundary conditions of the diffusion term over one time-step. In the next test we will see that the rotational correction will improve the accuracy of the pressure instead.

Next, we test for the second property introduced in §2.2.2. if $\mathbf{F} = \nabla p$, then $\mathbf{v} = 0$. Again we modify the forcing function for the analytical stokes problem, setting $\mathbf{v} = 0$ in this case, which gives the forcing function

$$\mathbf{F}_{\partial t} = -\pi \sin(t) \sin(\pi x) \sin(\pi y),$$
$$F^v_{\partial t} = \pi \sin(t) \cos(\pi x) \cos(\pi y).$$

We use the same mesh and time-step as in the first test. Again, we use a zero pressure as the predictor and examine the effect of using different boundary conditions.

For either choice of boundary condition, clearly, $\mathbf{v} \neq 0$ (Fig. [3-21] and Fig. [3-22]). Therefore, the time-split solution does not obey the second important property either. In this case, the boundary conditions for the diffusion term transforms the irrotational forcing such that part of it is in the divergence-free space. Since the error in the velocity originates from the diffusion term, it will be of the order $\mathcal{O}\left(\frac{\Delta t}{\mathrm{Re}}\right)$. Recall, we used a zero predictor pressure to illustrate the errors; If we used the exact

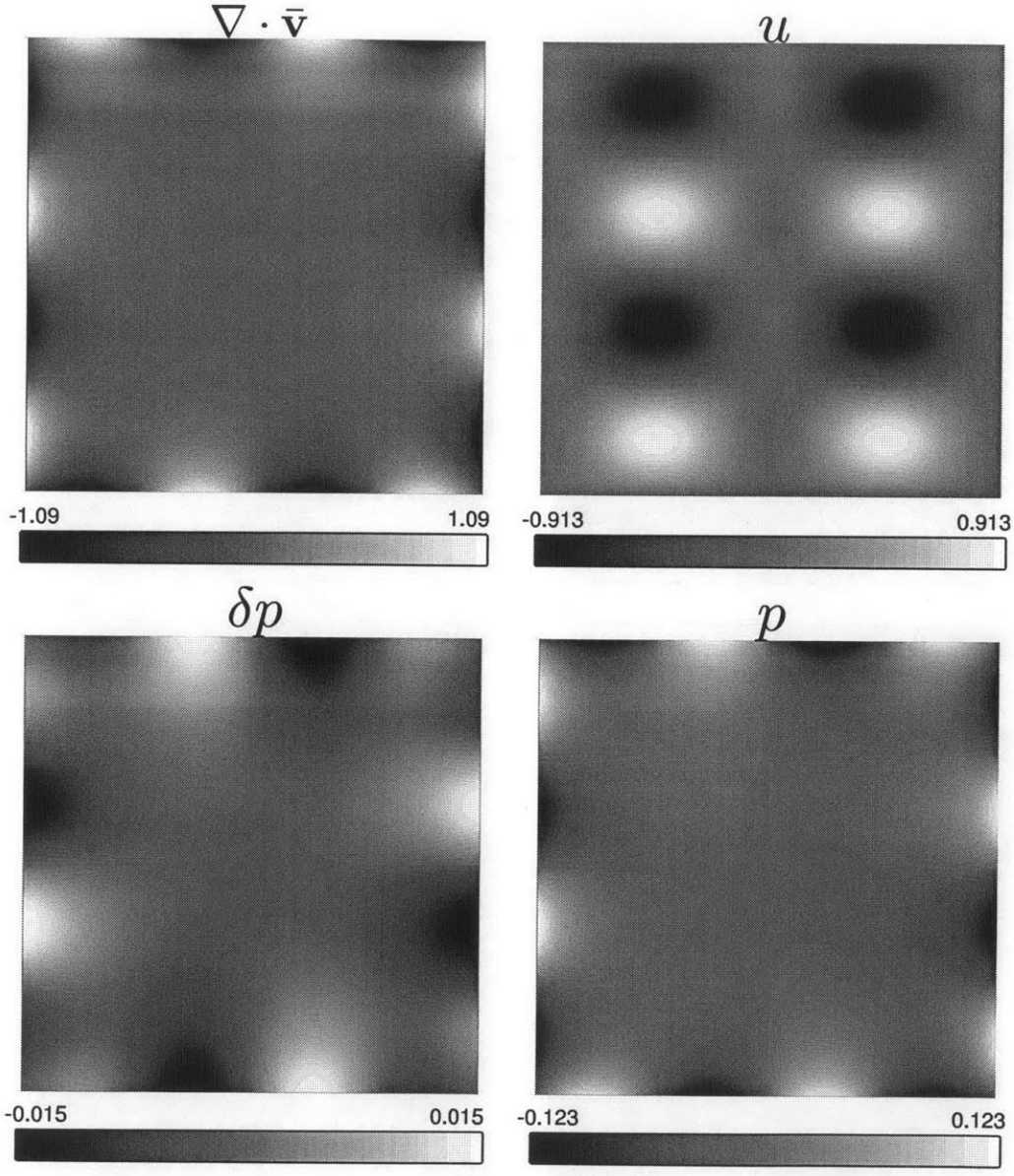


Figure 3-18: Test of the first numerical property introduced in §2.2.2 shown at $T = \frac{\pi}{4}$ after one time-step using a first-order Euler scheme. A mesh of 16×16 elements with a p = 6 degree polynomial but a large time-step of $\Delta t = 0.1$ is used. Using a divergence-free forcing function, the scaled divergence of the predictor velocity, the scaled final velocity, the scaled pressure correction, and the final scaled pressure are shown when using the boundary conditions from Timmermans et al. (1996).

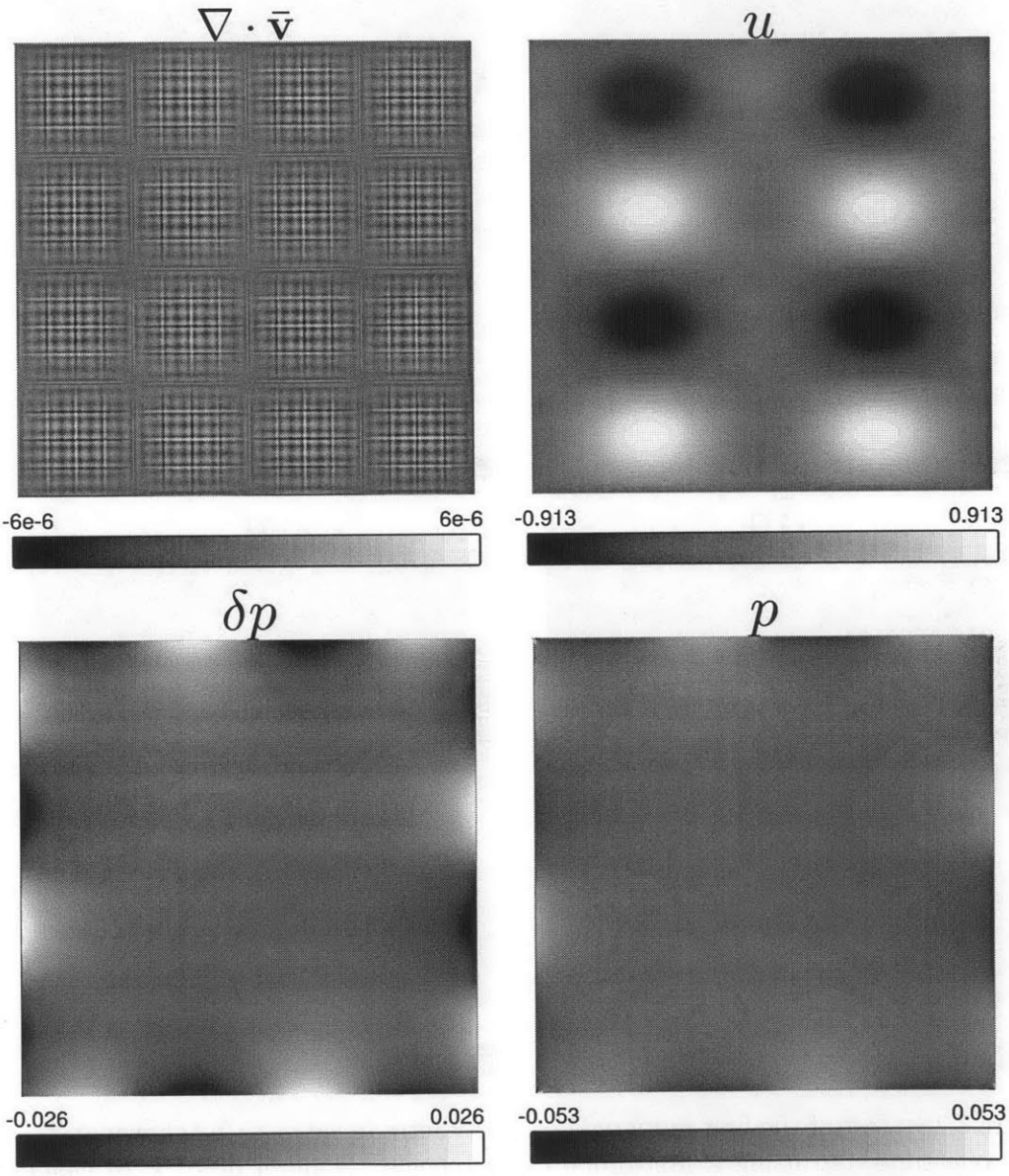


Figure 3-19: As in Fig. [3-18], but using the boundary conditions from Shirokoff and Rosales (2011).

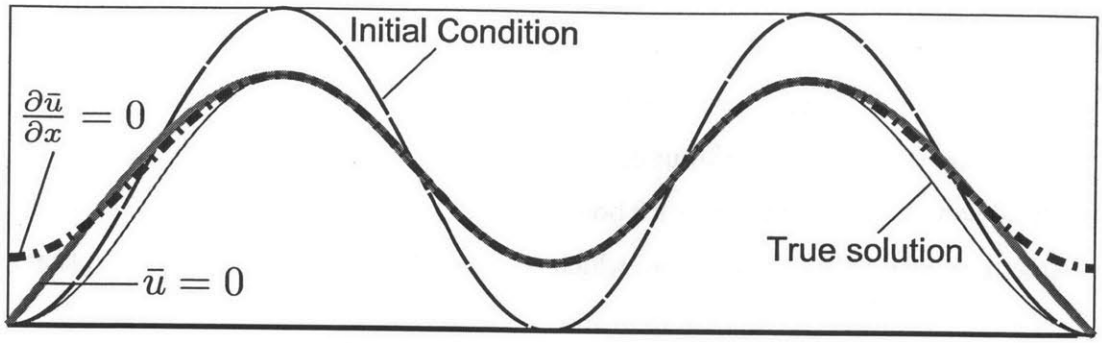


Figure 3-20: Sketch of a 1D slice through the u-velocity from Fig. [3-18] or Fig. [3-19]. With the uniform Dirichlet boundary conditions corresponding to Fig. [3-18], the solution follows the solid green line, which has a modified slope at the boundary. With the uniform Neumann boundary condition corresponding to Fig. [3-19], the solution follows the dashed blue line, which has a modified value at the boundary. The true solutions follows the thin, solid red line.

pressure, the right-hand-side forcing would have been zero, and the resulting velocity would have been correct. As such, if we can construct a good pressure predictor, this error in the velocity will be small. The pressure-correction is not a good predictor since we see that its magnitude is significantly smaller than it should be, and in Fig. [3-21] there are clear issues at the boundary. However, after the rotational correction is applied, both Fig. [3-21] and Fig. [3-22] have the correct magnitude of pressure, though some issues at the boundaries remain in Fig. [3-21]. Using this pressure as a predictor in the next time-step will nearly (up to $\mathcal{O}(\Delta t)$) set the forcing term equal to zero, but the overall accuracy of the velocity using this scheme will be limited to $\mathcal{O}\left(\frac{\Delta t^2}{\mathrm{Re}}\right)$.

These results suggest an alternative splitting algorithm. The second test shows that if the right-hand-side contains irrotational terms, part of these terms will be modified by the boundary conditions of the diffusion operator and cause an erroneous signal in the velocity. As such, the pressure should be calculated explicitly, such that the right-hand side forcing is divergence-free. This requires the solution of a pressure Poisson equation for the pressure. However, the first test showed that a divergence-free right-hand side does not give a divergence-free field, and we still require a correction. Fortunately, we know that this correction should not be a part

of the pressure. Additionally, if we use the boundary conditions of (Shirokoff and Rosales, 2011), this correction can be written as a boundary value problem. Then, what remains is to calculate the updated boundary conditions for the velocity, and potentially an update for the pressure boundary condition. Alternatively, it should be possible to solve a diffusion equation, which couples all the velocity components, that will not introduce irrotational components when the forcing is divergence-free. Here we have provided some basic guidelines, but the details of this alternative algorithm is the subject of future research. Alternatively, a possible avenue is to modify the scheme by Brown et al. (2001), where a high-order (in time) approximation of the pressure-correction boundary condition is needed.

In this section we showed that the numerical scheme does not satisfy the properties described in §2.2.2. We showed that it fails because of the time-splitting of the boundary conditions for the diffusion operator. We tested two different methods for splitting the boundary conditions, and both had similar errors. Based on our results, we have suggested a new algorithm that could eliminate these splitting errors. Next we examine the convergence of the present scheme.

3.8.3 Convergence Studies

We perform spatial and temporal convergence studies using the benchmark defined in §3.8.1 to verify that our scheme is correctly implemented and theoretically sound. For the spatial convergence study, we also test the effect of τ_p for the Appendix B scheme (Fig. [3-23]). The results for our §2.3.4 scheme are similar, and nearly identical for the consistent τ_p. We see that for $\tau_p = 1$ at mid-resolutions, there are large errors (likely numerical instability) in velocity and pressure for higher than degree 1 polynomial bases. However, as resolution increases, the errors reduce significantly, consistent with what was found in §2.3.4. As the magnitude of τ_p is increased, the errors start to converge more optimally, and when using the consistent value of τ_p, we obtain near-optimal convergence and nice straight lines.

The temporal convergence is more involved due to the additional complexity introduced by the projection method time-splitting. As such, we contrast the convergence

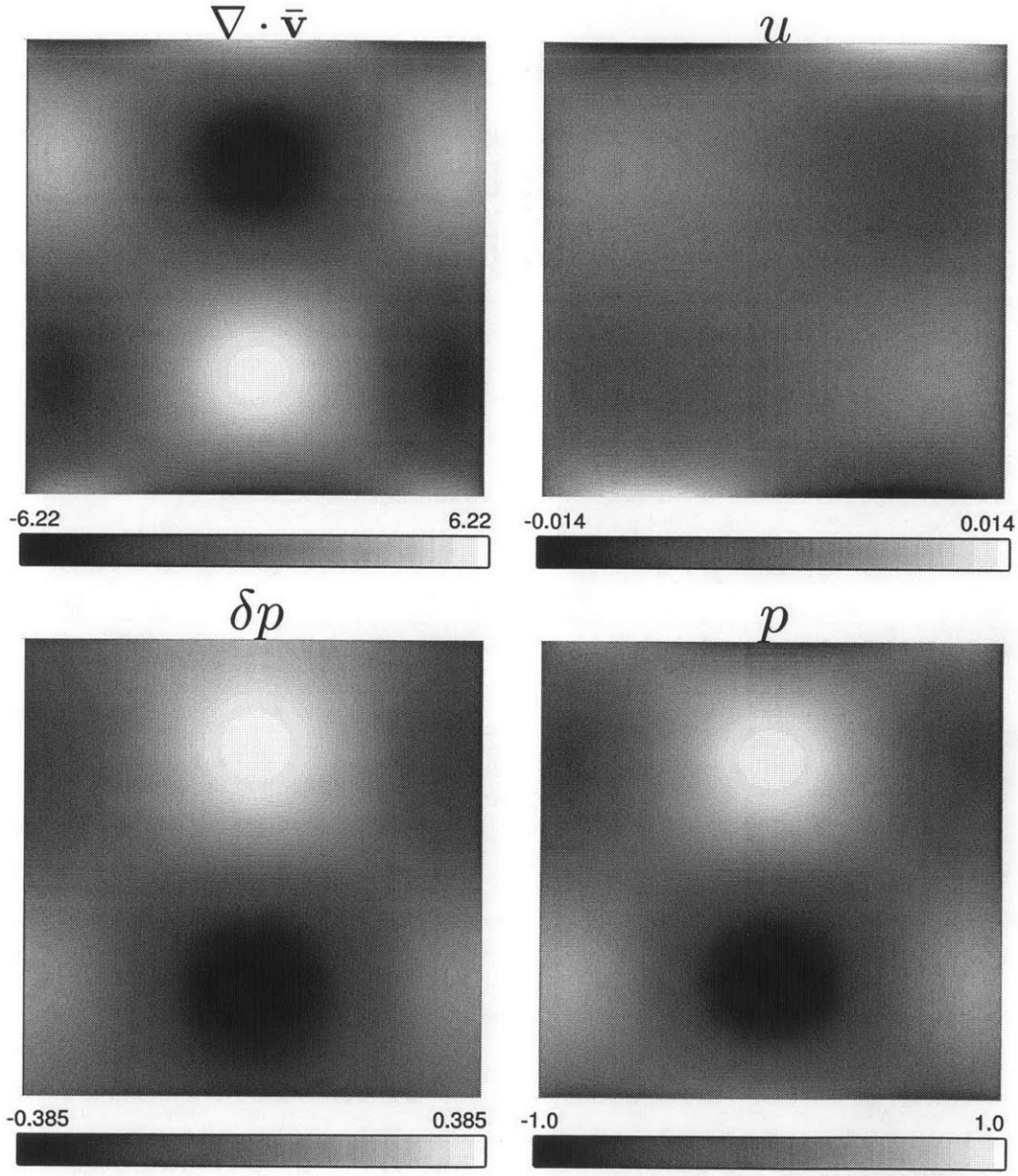


Figure 3-21: Test of the second numerical property introduced in §2.2.2 shown at $T = \frac{\pi}{4}$ after one time-step using a first-order Euler scheme. A mesh of 16×16 elements with a p = 6 degree polynomial but a large time-step of $\Delta t = 0.1$ is used. Using an irrotational forcing function, the scaled divergence of the predictor velocity, the scaled final velocity, the scaled pressure correction, and the final scaled pressure are shown when using the boundary conditions from Timmermans et al. (1996).

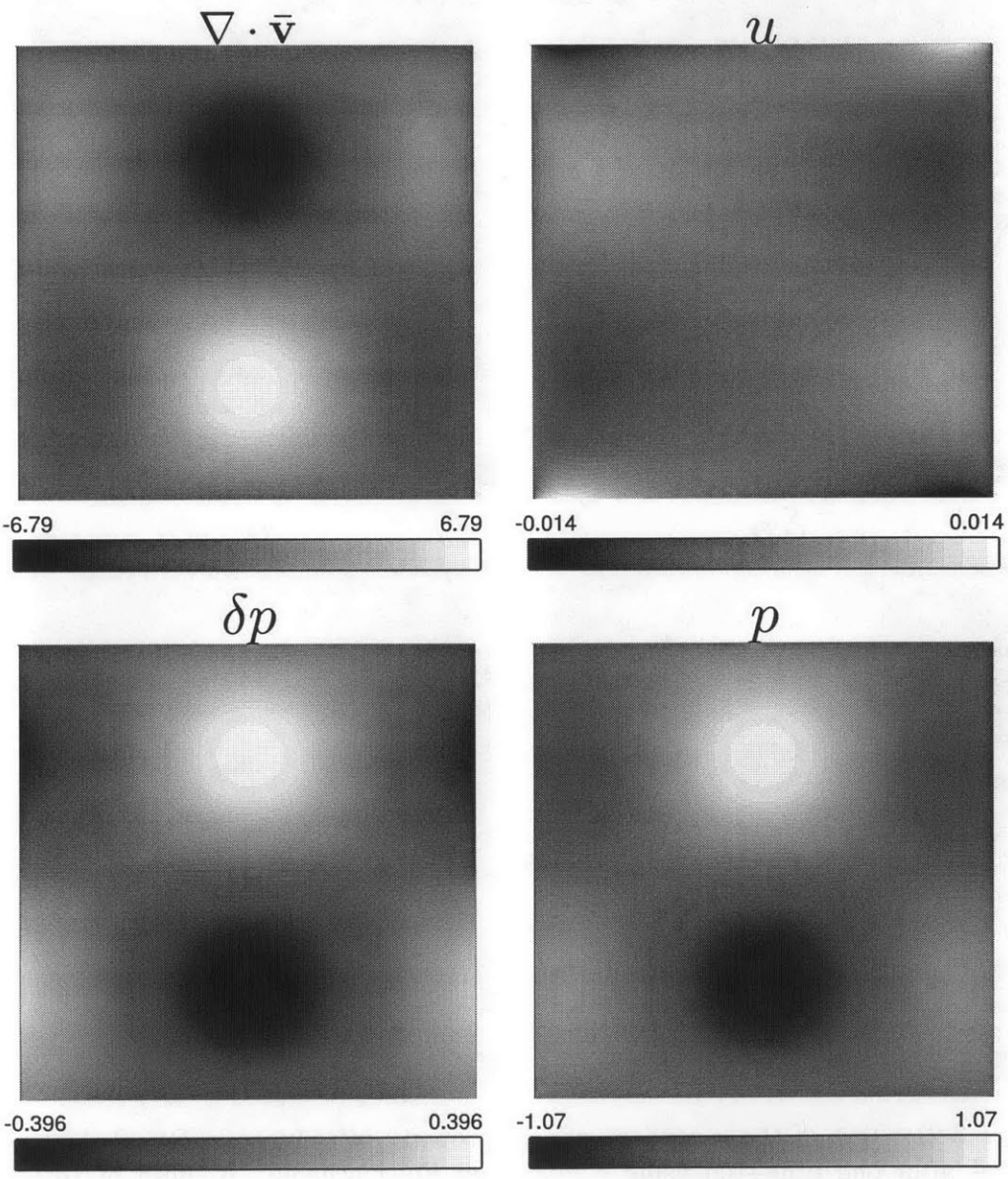


Figure 3-22: As in Fig. [3-18], but using the boundary conditions from Shirokoff and Rosales (2011).

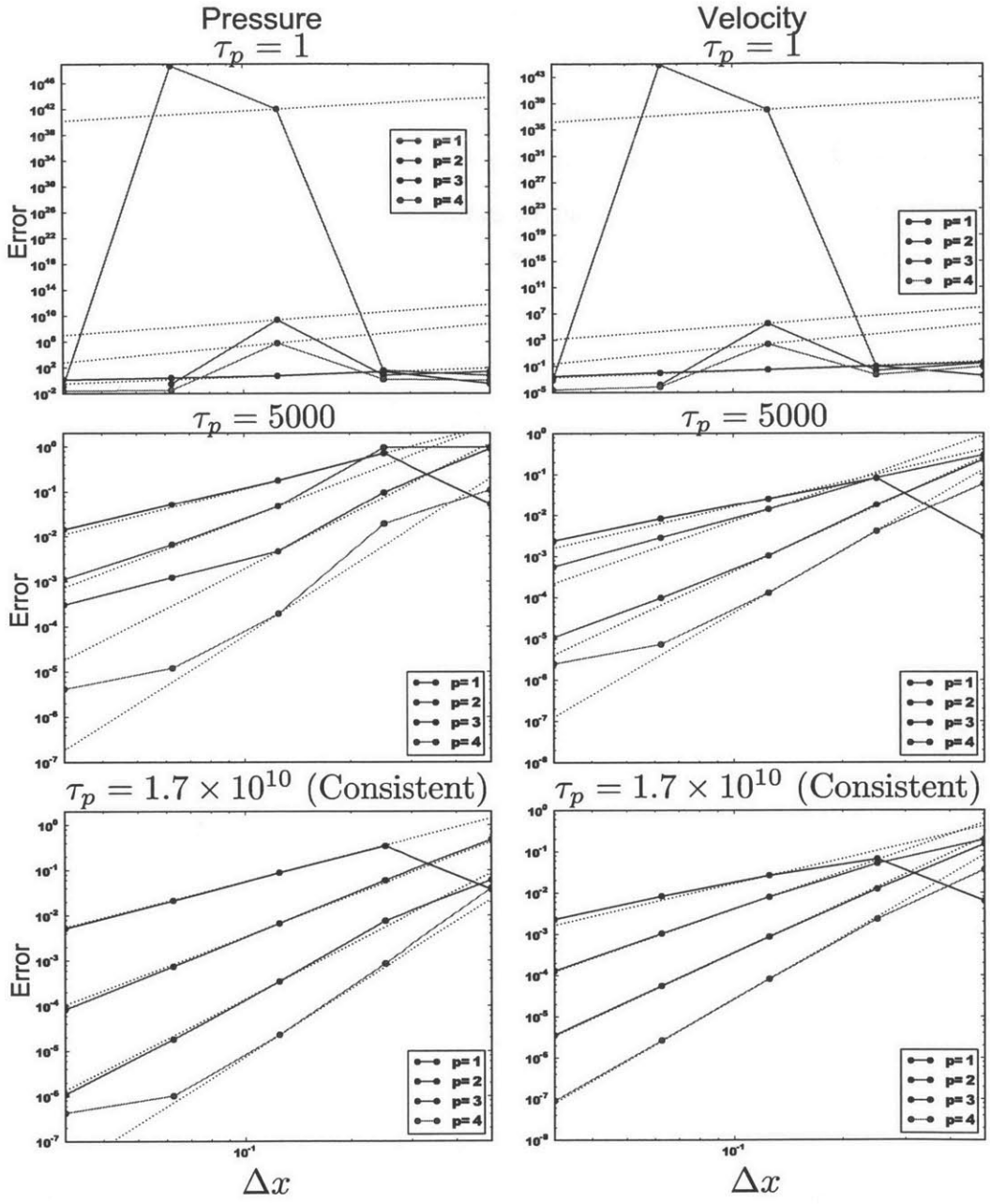


Figure 3-23: Spatial convergence of pressure (left) and velocity (right) using the analytical stokes problem with Re = 1, and $\tau_p = 1$ (top), $\tau_p = 5000$ (middle), and $\tau_p = 1.7e10 = \frac{1}{2a\tau\Delta t^2}$ (bottom), with theoretical optimal convergence in gray dashes. The Appendix B scheme is used with a second-order accurate IMEX integrator, with time-step fixed at $\Delta t = 10^{-5}$. For small values of τ_p the solution is not stable until sufficient resolution is reached. For larger values of τ_p the convergence is optimal, with excellent results for $\tau_p = \frac{1}{2a\tau\Delta t^2}$.

of our §2.3.2 scheme and the Appendix B scheme. We will test the convergence of both schemes with and without the rotational pressure-correction. The Appendix B scheme requires restarts for stability when the rotational corrections is used. A "restart" here refers to setting the pressure predictor to zero at the start of an IMEX integration stage. For these tests, it is important to evaluate the different parts of the forcing terms (3.33) at the correct time levels. The terms due to pressure should be evaluated explicitly for the Appendix B scheme, while the terms due to velocity should be evaluated implicitly for both, otherwise the convergence studies will be invalid. Also, for the temporal convergence studies of Appendix B we choose $\tau_p = \frac{1}{2\tau a \Delta t^2}$ for first and second order time integration, and $\tau_p = \frac{1}{2\tau a \Delta t^3}$ for third-order time-integration without restarts. When restarts are used, we choose $\tau_p = \frac{1}{2\tau a \Delta t}$ for first and second order time integration, and $\tau_p = \frac{1}{2\tau a \Delta t^2}$ for third-order time-integration. For our §2.3.2 scheme, we always use the consistent value, $\tau_p = \frac{1}{\tau a \Delta t}$.

The time-rate of convergence for the rotational and standard pressure-correction for our §2.3.2 scheme show that the rotational correction decreases the error in the pressure field, without as large an impact on the velocity field Fig. [3-24]. The rotational correction removes part of the pressure error near the boundary of the domain Fig. [3-26], as expected from Guermond et al. (2006).

Now examining the results of the Appendix B scheme. When using restarts, the errors for the rotational scheme is also reduced compared to the standard pressure-correction scheme (Fig. [3-25]). Again, pressure-errors near the boundary are removed Fig. [3-27]. However, the restart also decreases the order of convergence, reducing all time-integration schemes to essentially first-order accuracy. When the pressure predictor is not restarted at each stage, the Appendix B rotational pressure-correction scheme is unstable, and does not converge. Without restarts, the standard pressure-correction scheme (i.e. without rotational correction) converges optimally for first and second order time-integration schemes, and the third-order time-integration only converges at second order Fig. [3-28]. In particular, note that the pressure converges at second-order for the first order time-integration. This is likely related to our use of $\tau_p = \frac{1}{2\tau a \Delta t^2}$ for the first-order time-integration of this Appendix B scheme.

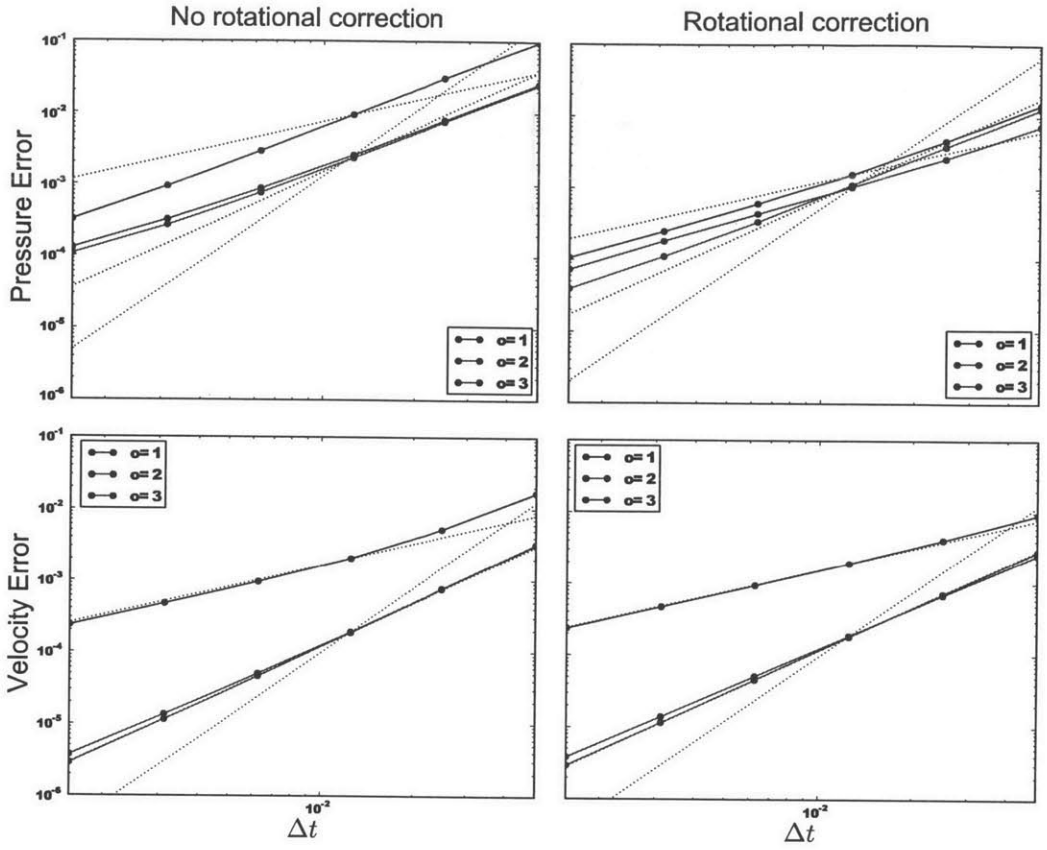


Figure 3-24: Temporal convergence of pressure (top) and velocity (bottom) using the analytical stokes problem with Re = 1 for our §2.3.2 scheme. A 64×64 square mesh with p = 6 was used for the spatial mesh, and first to third order accurate IMEX RK schemes were used. The rotational correction is applied (right), and not applied (left). The rotational correction lowers the absolute pressure-error.

Comparing the Appendix B convergence (Fig. [3-28]) to our §2.3.2 results (Fig. [3-24]), we see that our §2.3.2 scheme has smaller errors, particularly for the pressure. This demonstrates the advantage of our §2.3.2 scheme over the Appendix B scheme, and further justifies the use of our flux definition (2.78).

We would like to have a truly high-order accurate numerical scheme, both in time and space. However, the irreducible splitting error from the projection method restricts the accuracy of the solution to second order in time Fig. [3-24]. Since we argued that the origin of this error lies in the implicit diffusion term §2.2.2, §3.8.2, we can verify the correctness of our time-integration method by considering an infinite Reynolds number. When we do so, we find that pressure and velocity converge

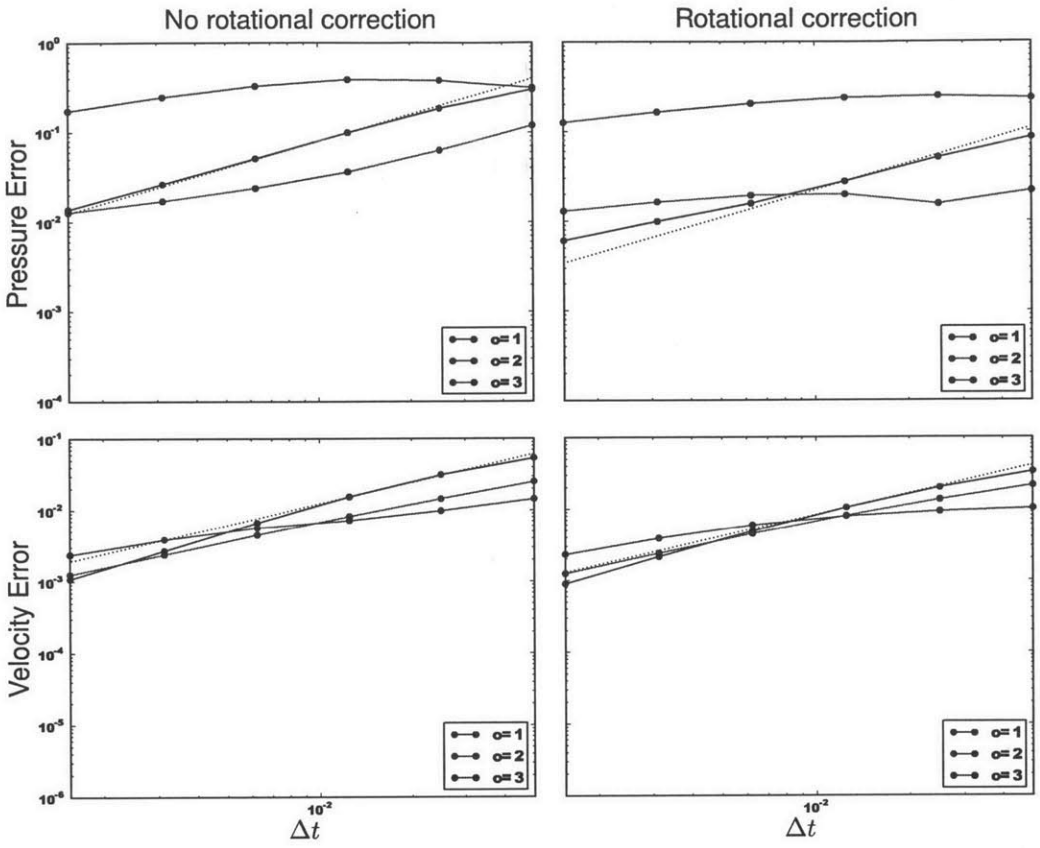


Figure 3-25: As in Fig. [3-24], but using the Appendix B scheme with restarts.

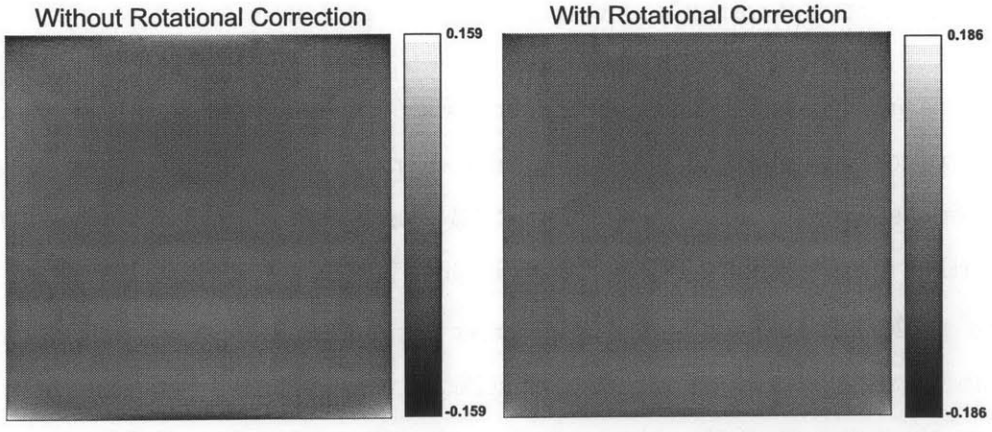


Figure 3-26: Pressure error for second order accurate IMEX-RK time integration using $\Delta t = 0.1$ for the standard (left) and rotational (right) pressure corrections of our §2.3.2 scheme. The rotational correction removes errors at the boundary of the domain, but errors at the corners remain.

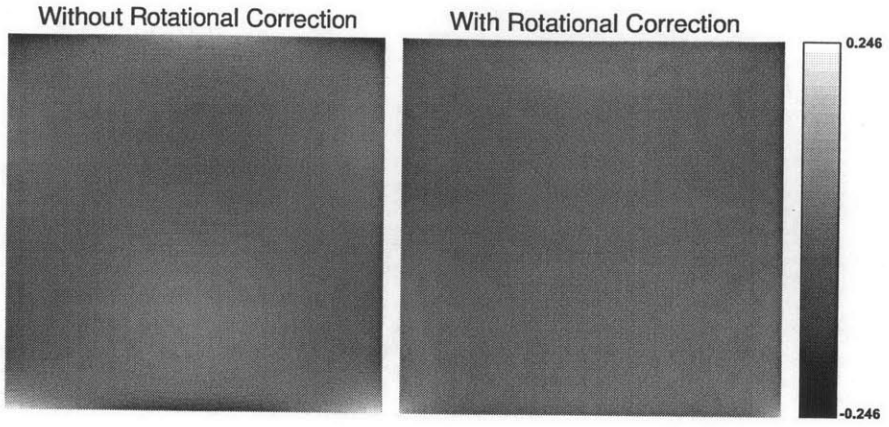


Figure 3-27: As in Fig. [3-26], but using the Appendix B scheme with restarts and $\Delta t = 0.025$.

optimally Fig. [3-29].

As the Reynolds number increases, we expect that the splitting error would have a smaller impact on the solution because it originates from the diffusion term (§3.8.2). To test this, we calculated the velocity and pressure errors for various Reynolds numbers. We note that the error in the pressure steadily decreases with increasing Reynolds number, and then saturates at $\mathrm{Re} = 10^6$ (Fig. [3-30]). The velocity error is not drastically affected for the second and third-order time-integration schemes, but the first-order scheme's error increases for increasing Reynolds number before plateauing. For the pressure, the calculated order of convergence approaches second-order accuracy, then decreases as the error plateaus. For the velocity, the calculated order of convergence is mostly unaffected. The transition from $\mathrm{Re} = 10^7$ to $\mathrm{Re} = \infty$ is not smooth, suggesting that the mere presence of the diffusion operator has an effect numerically. As such, there does not seem to be a benefit to using a time-integration scheme higher than second-order accurate when the implicit diffusion terms are present in a projection method.

However, we note that this projection-method restriction on the order of accuracy for the time integration may not be the limiting factor for the solution accuracy. First, since the time-step is restricted by the CFL condition for advection, the temporal dimension is often much more finely discretized than the spatial one, particularly when the non-dimensional physical advection-speed exceeds unity. Second, if higher-

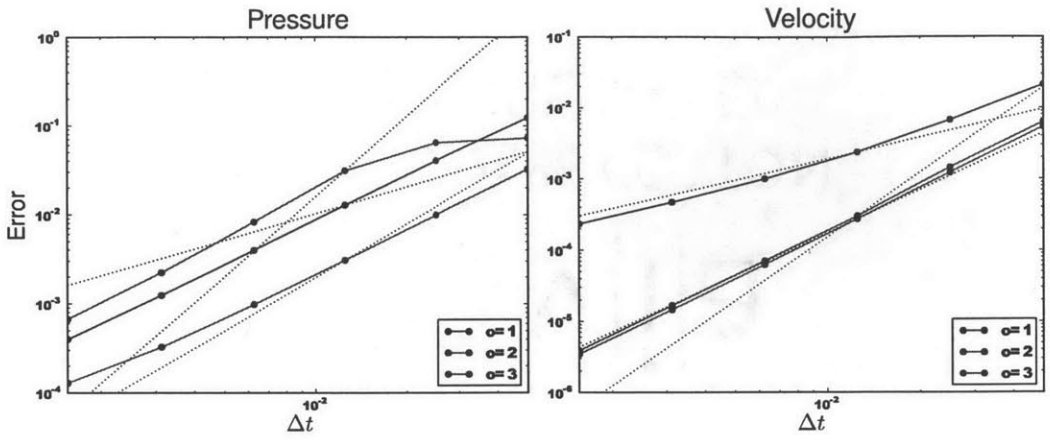


Figure 3-28: As in Fig. [3-25], using the Appendix B scheme, but without the pressure restart or the rotational correction. The second order scheme converges near-optimally for both pressure and velocity, while the third order scheme converges sub-optimally. The pressure converges at a faster rate for the first order scheme, while it converge near-optimally for the velocity.

order temporal accuracy is required, the projection method can be used as an iterative scheme, where the pressure-predictor of the second iteration is the final pressure from one full execution of the projection method (see for example Ferziger and Peric (2002)). This may still be more computationally efficient compared to solving a fully coupled system of equations. The fully coupled system requires the inversion of a matrix that is $d+1$ times larger compared to our smaller matrix inversions. Since matrix inversion often scales as the square of the number of unknowns, one full execution of the projection method is expected to be $(d+1)^2$ times more efficient (in d=3, this is a factor of 16). Thirdly, the results from a full execution of the projection method could also be used as a starting guess (or preconditioner) for the fully uncoupled problem.

Concerning the rotational correction, we argue that the rotational pressure correction is less important for higher-Reynolds number flows. Since Fig. [3-30] shows a decrease in the pressure error with increased Reynolds number and the size of the rotational correction is proportional to $\frac{1}{\mathrm{Re}}$, the rotational correction will have a decreasing effect on the pressure error. Also, if a good pressure-predictor (resulting from a fine temporal discretization for example) gives a nearly divergence-free right-hand-

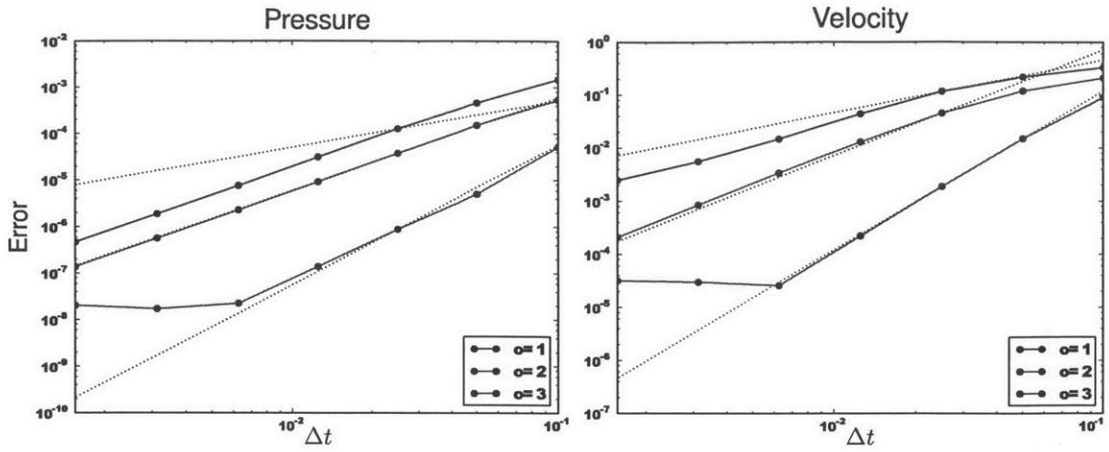


Figure 3-29: As in Fig. [3-28], but with Re $= \infty$. Now the third-order scheme also converges optimally.

side, the rotational correction may increase the error in the pressure (see Fig. [3-18]). Given the small gain in accuracy using the rotational pressure correction scheme, the use of the standard pressure correction may be justified for higher-Reynolds number flows.

In this section we have evaluated that our implementation is correct, and that our method behaves as expected. We showed that we can obtain near-optimal spatial and temporal rates of convergence when using an analytically defined problem. Next we need to validate our code against a standard benchmark case to ensure that our model gives the correct solution for an unforced case.

3.8.4 Validation

We validate our scheme by using the Lock-exchange problem with the same non-dimensional parameters as those of Härtel et al. (2000) and Fringer et al. (2006). Our simulation uses a 2D domain of size $[-8,\ 8] \times [0,\ 2]$, discretized using uniformly sized and distributed quadrilaterals of various resolutions, and we integrate for $T = [0,\ 10]$. We use the no-slip boundary condition at all boundaries, a Schmidt number of $Sc = 1$, and a Grashof number of $Gr = \frac{g'h^3}{\nu^2} = 1.25 \times 10^6$, where $g' = \frac{\Delta\rho}{\rho_0}$ is the reduced gravity

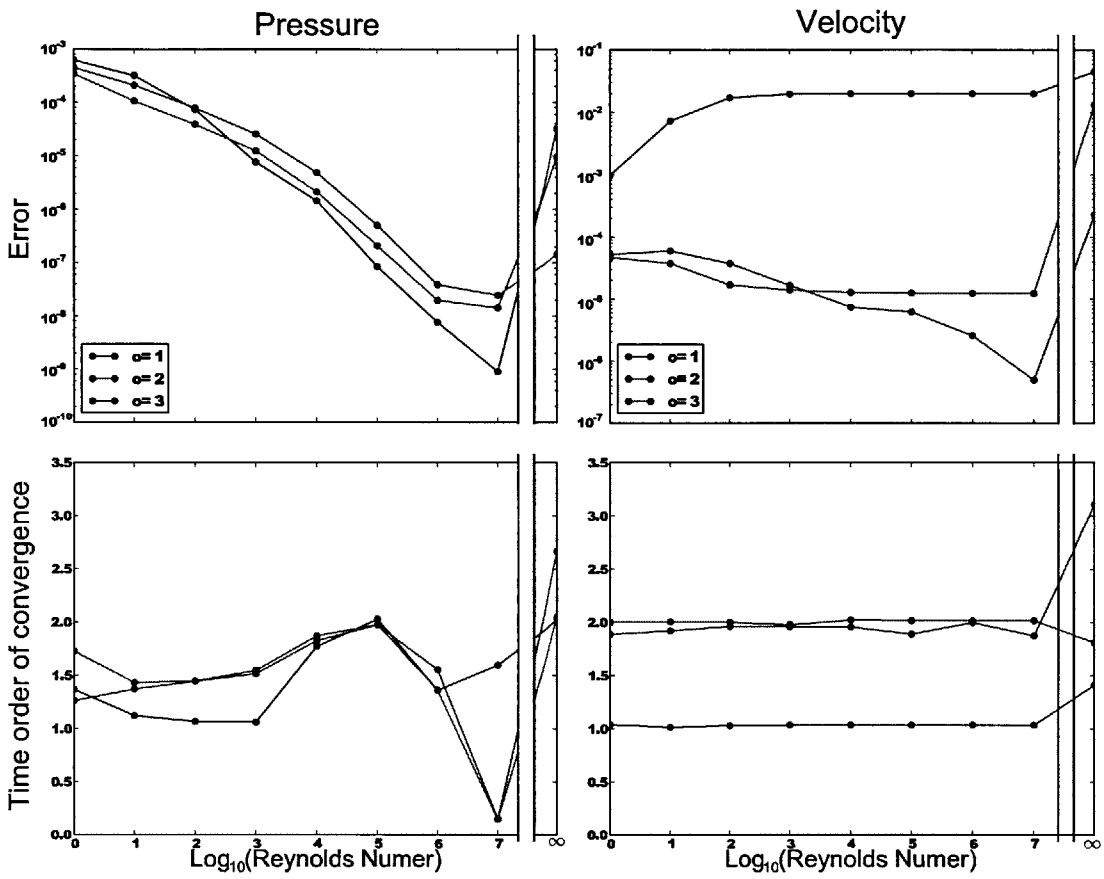


Figure 3-30: Error (top) of pressure (left) and velocity (right), and order of temporal convergence (bottom) using the analytical stokes problem with Re = 1. A 64×64 square mesh with p = 6 was used for the spatial mesh, and first to third order accurate IMEX RK schemes were used, the error is plotted for $\Delta t = 0.0125$ and the order of convergence was calculated using $\Delta t = [0.025, 0.0125]$. As the Reynolds number increases the pressure error decreases while the velocity error increases for the first-order IMEX scheme, with less effect for the higher-order schemes. The order of converge remains unaffected, but when Re= ∞, near optimal convergence is obtained for velocity.

and $h = 1$ is the half-height of the domain. The initial density profile is defined as

$$\rho = \frac{1}{2}\tanh(10^5 \mathbf{x}).$$

We use a second-order accurate time-integration scheme, with a fixed time-step of $\Delta t = 0.001$.

To compare our results to Härtel et al. (2000) and Fringer et al. (2006), we have to compute the Froude number $Fr = \frac{u_f}{u_b}$, where u_f is the speed of the front, defined as the speed at which the foremost point of the front travels, and $u_b = \sqrt{g'h}$ is the buoyancy velocity. To estimate u_f, we find the foremost point of the front at $T = 5$ and $T = 10$, then we simply use $u_f = \frac{\Delta x}{\Delta T} = \frac{\Delta x}{5}$, which gives an average front speed over that time period. Finding the foremost point of the front is non-trivial for cases using higher-order polynomial bases (Fig. [3-31]). In those cases we first identify the element that contains the foremost point of the front. Following this, we do an iterative root-find and line-search to find the foremost point. The root-find is needed to determine where in the element the density is equal to zero (since the finite element basis is a polynomial). When we find one such point, we evaluate the change of the x-coordinate at that point, and take a step in the direction that maximizes x. However, since this step is linear, we need to do a root-find again to get back onto the line where $\rho = 0$. We use an alternating direction Newton-Rhapson method to perform the root-find. That is, we do a 1D step in the x-direction, followed by a 1D step in the y-direction. At each iteration of the line-search, we decrease the step size by 2, approximating a bi-section method. While not an optimal scheme, it does find the leading edge of the front to machine precision, and it is not a computational bottleneck.

Our density contours are similar to those calculated by Härtel et al. (2000) (Fig. [3-32]). The higher-order simulations also match the second-order accurate simulation. Comparing the Froude numbers, as the spatial resolution is refined, our answer approaches the value of Härtel et al. (2000), and the spread in our results is on the same order as the difference between Härtel et al. (2000) and Fringer et al. (2006)

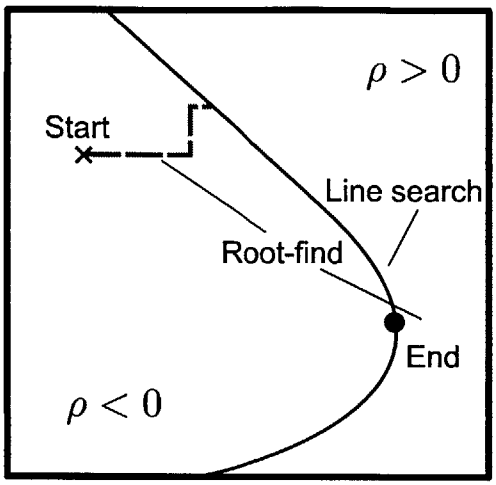


Figure 3-31: To find the foremost point of the density front where $\rho = 0$, an alternating direction Newton-Rhapson root-find (green dashes) is followed by a line-search (cyan dotted line). This procedure is applied iteratively until convergence.

(Fig. [3-33]). We note that the first-order time integration scheme performs nearly as well as the second-order time integration scheme for higher spatial resolutions. This suggests that the temporal dimension is well-resolved. Also, as the spatial resolution increases, the simulations with varying spatial order of accuracy agree more closely. For the low spatial resolution cases, there is a large spread of values between the high and low-order runs. In this case, finding the location of the foremost point of the front may play a role in the error. Nonetheless, our results agree closely with Härtel et al. (2000) and Fringer et al. (2006).

3.9 Summary

In this chapter we addressed numerical implementation issues of our new solution scheme for the Boussinesq equations (described in §2). Specifically, we justified our choice of using a nodal basis compared to a modal one. Then we discussed the computational accuracy and efficiency of quadrature-based and quadrature-free integration schemes. Based on this discussion, we selected the quadrature-free integration method for our implementation. As such, we also developed a quadrature-free scheme that is consistent with the HDG method. Since the implementation of HDG methods

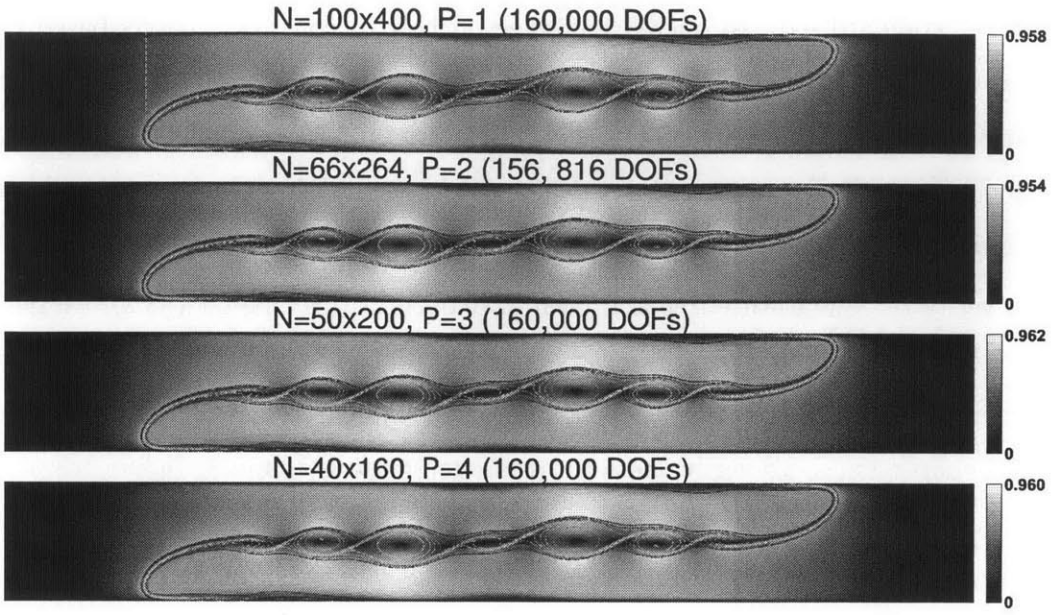


Figure 3-32: Density solution at time 10 of the Lock-Exchange problem ($Gr = 1.25 \times 10^6$) using various orders of accuracy and spatial resolution, all runs with approximately 160,000 degrees of freedom. There are some minor differences in the front propagation speed and the shape of the Kelvin-Helmholtz instabilities.

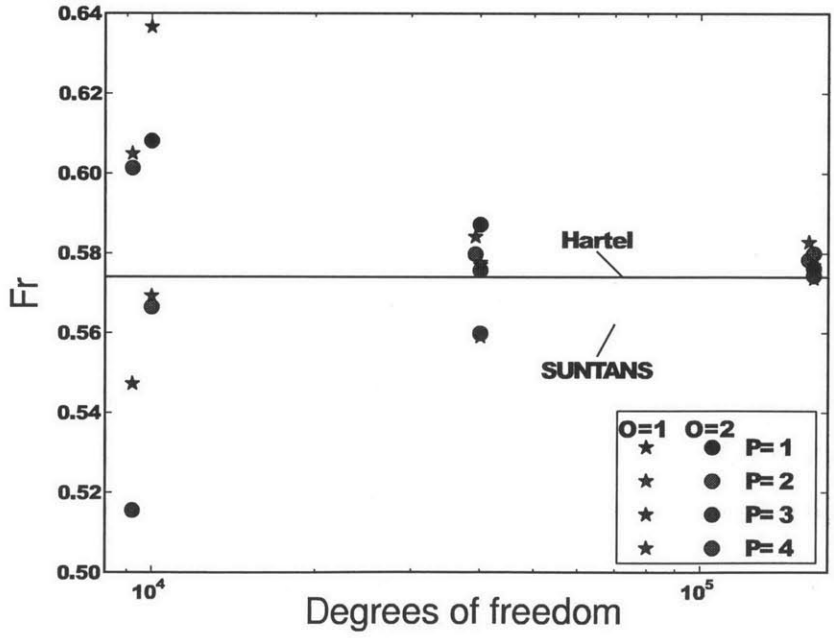


Figure 3-33: Density front propagation speed for various resolutions for the no-slip case. Solid and dashed lines indicate the solution obtained by HARTEL and SUNTANS respectively.

is not yet standardized, we offered guidance on how to construct matrix-based or matrix-free solvers. Then, we detailed our selective nodal limiting approach used to stabilize high-order schemes while retaining high-order accuracy in regions where the solution is smooth. Next we explained our code design philosophy, also describing the classes and modules relevant to new users of the framework.

To verify the quadrature-free approach developed in this chapter (§3.2), we performed a convergence study on a steady diffusion problem using straight-sided and curved meshes. We showed that both straight-sided and curved meshes achieve near-optimal convergence. To verify the selective nodal limiting approach (§3.5), we performed a convergence study on an unsteady tracer advection problem. We showed that when the selective nodal limiter is applied the rate of convergence reduces to second-order for coarse resolutions while the high-order accuracy is retained for fine resolutions.

For the class of projection methods considered, we showed that the time-splitting error comes from the implicit treatment of the diffusion term and its boundary conditions when the right-hand-side forcing is divergence-free. Then we demonstrated how the splitting-error manifests itself in the pressure and velocity fields using an analytical benchmark for this particular case. We also showed that the rotational correction to the pressure may increase the error in the pressure when the right-hand side forcing is divergence-free. Using numerical examples, we showed that it is essential to use the proper magnitude of the HDG stability parameter for the pressure when the spatial domain is coarsely resolved. Then we showed that our discretization of the rotational correction term removes pressure errors at the boundary of the domain for the analytical benchmark. We also showed that while the splitting error decreases with increasing Reynolds number, it continues to limit the time-order of accuracy of the scheme.

We performed spatial and temporal convergence studies to verify that our full Navier-Stokes solver based on the new HDG projection method scheme is properly formulated and implemented. We evaluated our approach by comparing our results for the standard lock-exchange benchmark to published literature. We found that

our solution closely matched the previously published data.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Formulation and Discretization of a Novel Non-Hydrostatic Ocean Model

With advances in Computational Fluid Dynamics (CFD) over the last twenty years, an improved understanding of ocean dynamics, and the evolution of computer architectures, it has become evident that ocean science studies can benefit from a new generation of prediction codes. We are interested in studying nonlinear, transient, and multiscale ocean dynamics over complex geometries with steep bathymetry and intricate coastlines, from sub-mesoscales to basin-scales. Our requirements for an accurate, efficient and flexible next-generation ocean model led us to develop a variable resolution, higher-order, non-hydrostatic ocean model.

Ocean dynamics and forcing can be highly nonlinear, with inhomogeneous, unstationary and anisotropic variability. With an improved understanding of ocean dynamics, it is now known that many ocean processes involve multiple scales with inherent transient effects and intense localized gradients. Depending on the phenomena being studied, important time scales can be on the order of seconds to the order of geological time scales. Similarly, important length scales can be on the order of millimeters to planetary length scales. While significant scientific progress has been made by studying only one or two scales at a time, strongly interacting dynamics

over many time and space scales need to be studied to further advance scientific understanding (Dolbow et al., 2004, Beck, 2009).

Present ocean models solve rather similar geophysical fluid dynamics equations and use similar numerical methods, essentially all based on the schemes proposed by Bryan (1969) and others in the early seventies. Since then, significant advances have been made in the parameterization of subgrid-scale processes, in data assimilation methodologies and in boundary conditions. Also, due to more efficient implementations on more advanced hardware, model resolution has substantially increased. While they have excellent efficiency per degree of freedom (DOF), present models are largely based on conservative but low-order discretizations on uniform, non-adaptive structured grids (see Griffies et al. (2000, 2010) for a review). The increased sophistication of ocean modeling systems and increased model resolution without updates to the numerics is, however, not sufficient for simulating the vast scales present in the ocean. It is the recent advances in CFD that can further increase the range of resolved scales (Deleersnijder and Lermusiaux, 2008) without a concomitant increase in computational cost.

Having recognized that present models can benefit from new numerical methods, a number of groups have started developing next-generation ocean models. For a review of next-generation models see Slingo et al. (2009), or Ueckermann (2009). While the various new models adopt largely varying approaches, a unifying element for all these models is the use of unstructured (normally triangular) meshes in the horizontal direction. Also, it is notable that some models solve non-hydrostatic equations, and some have sophisticated adaptive meshes. A whole community of modelers have been formed, with annual workshops and special publications (Deleersnijder and Lermusiaux, 2008, Deleersnijder et al., 2010). A few examples of idealized and realistic applications of these new models include two dimensional modeling of tides (Westerink et al., 1994, Wunsch et al., 1997) and storm surges (Lane and Walters, 2009, Westerink et al., 2007), and three dimensional modeling of tides (Walters et al., 2010), internal tides (Jachec et al., 2007), estuaries and rivers (Baptista and Zhang, 2008, Liu et al., 2008, de Brye et al., 2012), bays (Chen et al., 2008), and seas (Wang et al., 2009,

Ernsdorf et al., 2011). While the greater geometric flexibility of unstructured grids used in new models allow more accurate solutions (because of improved discretization of the boundaries and interior), the major drawback is a reduced efficiency per DOF. As a result, new models have primarily been adopted in regions with complex geometries (such as estuaries), but have not yet been used for real-time forecasting in regional domains. Additionally, higher-order models, due to their complexity and computational efficiency per DOF have also not yet been widely adopted.

Since our review of next-generation models (Ueckermann, 2009) there have been a number of advances in numerics by the ICOM and SLIM groups. As our work is most closely related to these two groups, it is important to frame our contributions in reference to their efforts. In particular, ICOM has researched solution methods of the Poisson equation (Kramer et al., 2010), efficient implementation methods on GPU architectures (Markall et al., 2010b,a), the selection of finite element pairs (Cotter and Ham, 2011), mesh adaptivity (Davies et al., 2011, Farrell and Maddison, 2011, Farrell et al., 2011, Farrell, 2011, Maddison et al., 2011a) and how to numerically capture geostrophic balances (Maddison et al., 2011b), to name a few areas. The SLIM group has researched finite element pairs (Comblen et al., 2010b), time-stepping schemes (Comblen et al., 2010a, Seny et al., 2013), Lagrangian schemes (Shah et al., 2011), turbulence closure (Kärnä et al., 2012b) and generally improved their baroclinic ocean model, including numerical conservation and consistency considerations (Blaise et al., 2010, Kärnä et al., 2012a). While both groups are actively developing their numerical schemes, both have applied their models to various applications. ICOM and SLIM both utilize discontinuous finite elements and have evaluated different finite element pairs, but neither have considered the novel hybrid discontinuous Galerkin (HDG) method. Also, our focus is on higher-order accurate methods. In light of their work, we develop schemes to evaluate whether high-order HDG-elements are appropriate for ocean modeling. We also consider the efficient implementation of HDG methods for various terms in the equations (including the Poisson term), semi-implicit time-stepping schemes, and the numerical stability/consistency of the HDG scheme for ocean flows.

Based on our previous work developing new computational schemes for two-dimensional unsteady biogeochemical ocean models, (Ueckermann, 2009, Ueckermann and Lermusiaux, 2010), here we derive and develop new high-order, unstructured grid, hybridized discontinuous Galerkin finite element schemes for ocean modeling. The corresponding model codes can be solved hydrostatically, non-hydrostatically, and with a free-surface or a rigid lid. First we review the derivation of the continuous ocean equations starting from the Boussinesq equations (§4.1). Then we describe our time-discretization scheme (§4.2) which uses projection methods. Following this, we perform the hybridized discontinuous Galerkin spatial discretization (§4.3). Finally, we validate our new code using idealized benchmarks (§4.4)

4.1 Formulation of continuous ocean equations

Governing equations for ocean dynamics are usually derived starting from the full Navier-Stokes equations written in the reference frame of the earth (that is, with Coriolis forces). In Cartesian coordinates, under the Boussinesq approximation (small density variations and at first order incompressible flow) and eddy-viscosity approximation (other closures are discussed later, all assume small molecular viscosities/diffusivities), one obtains the following governing equations:

$$\begin{aligned}\frac{\partial \mathbf{u}}{\partial t} - \nabla_z \cdot \nu_z \nabla_z \mathbf{u} + \frac{1}{\rho_0}\nabla_{xy} p = &-\nabla_{xy}\cdot \mathbf{u}\mathbf{u} - \nabla_z \cdot w\mathbf{u} \\ &+ \nabla_{xy}\cdot \nu_{xy}\nabla_{xy}\mathbf{u} - f\hat{\mathbf{k}}\times\mathbf{u} + \frac{1}{\rho_0}\mathbf{f}_{\mathbf{u}},\end{aligned} \tag{4.1}$$

$$\begin{aligned}\frac{\partial w}{\partial t} - \nabla_z \cdot \nu_z \nabla_z w + \frac{1}{\rho_0}\nabla_z p = &-\nabla_{xy}\cdot \mathbf{u}w - \nabla_z \cdot ww \\ &+ \nabla_{xy}\cdot \nu_{xy}\nabla_{xy} w + \frac{\rho}{\rho_0}\mathbf{g} + \frac{1}{\rho_0}\mathbf{f}_w,\end{aligned} \tag{4.2}$$

$$\nabla_{xy}\cdot\mathbf{u} + \nabla_z \cdot w = 0, \tag{4.3}$$

$$\frac{\partial T}{\partial t} - \nabla_z \cdot \kappa_z \nabla_z T = -\nabla_{xy}\cdot \mathbf{u}T - \nabla_z \cdot wT + \nabla_{xy}\cdot\kappa_{xy}\nabla_{xy}T + f_T, \tag{4.4}$$

$$\frac{\partial S}{\partial t} - \nabla_z \cdot \kappa_z \nabla_z S = -\nabla_{xy}\cdot \mathbf{u}S - \nabla_z \cdot wS + \nabla_{xy}\cdot\kappa_{xy}\nabla_{xy}S + f_S \tag{4.5}$$

$$\rho = \rho(S,T), \tag{4.6}$$

where p is the full gauge pressure, ρ is the density, and the prognostic state variables are the horizontal and vertical velocity components $\mathbf{u} = [u, v, 0]$ and $w = [0, 0, w]$, respectively, the temperature, T, and the salinity, S. The independent variables are the spatial coordinates, denoted $[x, y, z]$, and time, t. The horizontal and vertical gradient operators are denoted $\nabla_{xy} = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, 0\right]$ and $\nabla_z = \left[0, 0, \frac{\partial}{\partial z}\right]$, respectively. We denote the constant reference density ρ_0, the gravity vector $\mathbf{g} = [0, 0, -g]$, the Coriolis parameter f, the vertical and horizontal turbulent viscosities ν_{xy} and ν_z, the horizontal and vertical turbulent diffusivities κ_{xy} and κ_z, and the forcing terms for the horizontal momentum, vertical momentum, temperature, and salinity equations $\mathbf{f_u}$, $\mathbf{f}_w$, f_T, and f_S, respectively. Note, to avoid the re-arrangement of terms later, we have written the terms that will be treated implicitly and explicitly on the left and right sides of the equations, respectively.

Next, we introduce a set of approximations and derivations, and obtain the final equations that we solve. Specifically, in §4.1.1, we justify the Boussinesq approximation for our dynamics of interest and review the errors that are introduced by this approximation. In §4.1.2, we consider two different definitions of the hydrostatic pressure for ocean flows with a free air-sea surface (in short, free-surface flows), justify one of them, and provide related derivations. In §4.1.3, we derive our nonlinear free-surface equation and boundary conditions. The rigid lid formulation and turbulence closures are briefly outlined in §4.1.4 – §4.1.5. The final ocean equations combining these approximations and derivations are summarized in §4.1.6, for both the non-hydrostatic and hydrostatic dynamics.

4.1.1 Justification of Boussinesq approximation

In classical fluid mechanics, the density is often treated as a constant, where fast time-scale variations due to acoustic waves are neglected. The Boussinesq approximation relaxes this assumption, and retains the gravitational forcing in the momentum equations due to small density variations. The justification is that the gravitational acceleration constant increases the size of the small density variations in this forcing term. As such, to leading order, this is the most important variation to retain in

the equations. We briefly review how this approximation affects the Navier-Stokes equations, specifically indicating the size of the expected errors for ocean applications.

The full density field is decomposed into a constant reference, and temporally/spatially varying component:

$$\rho(\mathbf{x}, t) = \rho_0 + \rho'(\mathbf{x}, t). \tag{4.7}$$

Substituting this into the continuity equation $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$ we obtain successively

$$\begin{aligned}
&\frac{\partial(\rho_0 + \rho')}{\partial t} + \nabla \cdot ((\rho_0 + \rho')\mathbf{v}) = 0, \\
&\qquad \Rightarrow \nabla \cdot \mathbf{v} = -\frac{1}{\rho_0}\rho' \nabla \cdot \mathbf{v} - \frac{1}{\rho_0}\frac{\partial \rho'}{\partial t} - \frac{1}{\rho_0}\mathbf{v} \cdot \nabla(\rho'), \\
&\qquad \Rightarrow \nabla \cdot \mathbf{v} = -\frac{1}{\rho_0}\rho' \nabla \cdot \mathbf{v} - \frac{1}{\rho_0}\frac{\mathrm{D}\rho'}{\mathrm{D}t}, \\
&\qquad \Rightarrow \nabla \cdot \mathbf{v} \approx 0,
\end{aligned}$$

where $\mathbf{v} = \mathbf{u} + w$. In the ocean, the density variation is on the order of 1%. As such, the $\frac{\rho'}{\rho_0}\nabla \cdot \mathbf{v}$ term will be of a similar magnitude. Since the salinity is conserved and density differences due to energy dissipation is negligibly small in the interior of the ocean, the density following a fluid particle is approximately conserved, and the $\frac{1}{\rho_0}\frac{\mathrm{D}\rho'}{\mathrm{D}t}$ term will be of the order 1% or smaller (e.g. (Cushman-Roisin and Beckers, 2011)). To summarize, the following terms were neglected in the continuity equation leading to an error of order 1% in the continuity equation:

$$\frac{\rho'}{\rho_0}\nabla \cdot \mathbf{v} + \frac{1}{\rho_0}\frac{\mathrm{D}\rho'}{\mathrm{D}t} \approx 0.$$

The error of neglecting the density perturbations in the momentum equations will have a similar magnitude. Part of the error comes from dividing by the mean density (that is, neglecting the perturbation density in the division), but this error is small.

To see this, use a Taylor series expansion,

$$\frac{1}{\rho} = \frac{1}{\rho_0 + \rho'} = \frac{1}{\rho_0} - \rho' \frac{1}{\rho_0^2} + \dots$$

which shows that the leading error term for this part of the approximation will be of order 0.01%. However, when dividing only by the mean density, additional terms are left in the momentum equations that will be neglected. For the horizontal momentum equations, these terms are

$$g\frac{\rho'}{\rho_0}\nabla_{xy}\eta - \frac{\rho'}{\rho_0} f\hat{\mathbf{k}} \times \mathbf{u} - \frac{1}{\rho_0}\left\{\frac{\partial \rho' \mathbf{u}}{\partial t} + \nabla_{xy} \cdot \rho' \mathbf{u}\mathbf{u} + \nabla_z \cdot \rho' w \mathbf{u}\right\} \approx 0.$$

The first term comes from the free-surface (see §4.1.2). The second term comes from neglecting the perturbation density multiplied by the Coriolis parameter, $\frac{\rho'}{\rho_0} f\hat{\mathbf{k}} \times \mathbf{u}$. For the last term, the continuity equation was used to cast the momentum equations in the conservative form. As such, we also neglected terms that are similar to those neglected in the continuity equation, leading to the same error magnitude of order 1% in the momentum equations.

Thus, for our oceanic applications, the Boussinesq approximation introduces a 1% error in both the continuity and momentum equations.

4.1.2 Hydrostatic pressure definitions and derivations

The full pressure can be decomposed as a sum of hydrostatic and non-hydrostatic components

$$p = p_{hyd} + \rho_0 p'. \tag{4.8}$$

In this section we focus explicitly on the hydrostatic component, p_{hyd}, and while a free-surface component will also appear, it is discussed in detail in §4.1.3.

Next, to simplify the solution method, a hydrostatic pressure that balances the density forcing can be defined. This pressure can be defined in terms of the total density or the mean density. The solution method of hydrostatic ocean models is

simplified by defining the hydrostatic pressure in terms of the total density, but since we are also interested in non-hydrostatic models, we want to explore all possibilities. The hydrostatic pressure could be defined as either of the following

$$p_{hyd}(x,\, y,\, z,\, t) = \int_z^{\eta(x,y,t)} \rho(x,\, y,\, \zeta,\, t)\, g\, \mathrm{d}\zeta, \tag{4.9}$$

$$p^*_{hyd}(x,\, y,\, z,\, t) = \int_z^{\eta(x,y,t)} \rho_0\, g\, \mathrm{d}\zeta = g\, \rho_0(\eta - z) \tag{4.10}$$

where η is the height of the free surface relative to the undisturbed ocean, and ζ is a dummy integration variable. To see how the different definitions of the hydrostatic pressure appear in the momentum equations, we will calculate the horizontal and vertical gradients.

First, for (4.9) we have

$$\begin{aligned}
\nabla_{xy} p_{hyd} &= \nabla_{xy} \left\{ \int_z^{\eta(x,y,t)} \rho(x,\, y,\, \zeta,\, t)\, g\, \mathrm{d}\zeta \right\}, \\
&= \nabla_{xy} \left\{ \int_z^{\eta(x,y,t)} \rho_0\, g\, \mathrm{d}\zeta \right\} + \nabla_{xy} \left\{ \int_z^{\eta(x,y,t)} \rho'(x,\, y,\, \zeta,\, t)\, g\, \mathrm{d}\zeta \right\}, && (4.11) \\
&= g\, \rho_0 \nabla_{xy}\{\eta - z\} + g\, \rho'(x,\, y,\, \eta,\, t) \nabla_{xy}\eta - g\, \rho'(x,\, y,\, z,\, t) \nabla_{xy} z \\
&\quad + \int_z^{\eta(x,y,t)} \nabla_{xy} \rho'(x,\, y,\, \zeta,\, t)\, g\, \mathrm{d}\zeta, && (4.12) \\
&= g\, (\rho_0 + \rho') \nabla_{xy}\eta + \int_z^{\eta(x,y,t)} \nabla_{xy} \rho'(x,\, y,\, \zeta,\, t)\, g\, \mathrm{d}\zeta, && (4.13)
\end{aligned}$$

where we have used the Leibniz rule to interchange the order of the gradient and the integral. We could have kept the gradient term outside the integral sign (4.11), and this has implementation consequences. In (4.13) we have to calculate the vertical integral of a gradient as opposed to a scalar. This, however, is more accurate when terrain-following coordinates are used (Wang et al., 2008). Also, we will neglect the $g\rho'\nabla_{xy}\eta$ term, since it will be of order 1% compared to $g\rho_0\nabla_{xy}\eta$, and this is consistent

with the approximation made in §4.1.1. Now for the vertical gradient we have

$$\begin{aligned}
\nabla_z p_{hyd} &= \nabla_z \left\{ \int_z^{\eta(x,y,t)} \rho_0 \, g \, \mathrm{d}\zeta \right\} + \nabla_z \left\{ \int_z^{\eta(x,y,t)} \rho'(x,\, y,\, \zeta,\, t) \, g \, \mathrm{d}\zeta \right\}, \\
&= g\rho_0 \nabla_z \{\eta - z\} + g\rho'(x,\, y,\, \eta,\, t) \nabla_z \eta - g\rho'(x,\, y,\, z,\, t) \nabla_z z \\
&\quad + \int_z^{\eta(x,y,t)} \nabla_z \rho'(x,\, y,\, \zeta,\, t) \, g \, \mathrm{d}\zeta, \\
&= -g(\rho_0 + \rho'), \\
&= -g(\rho),
\end{aligned}$$

where we note that $\rho'(x,\, y,\, \zeta,\, t)$ is not a function of z. This vertical pressure gradient will exactly cancel the $\frac{\rho}{\rho_0}\mathbf{g}$ term in the momentum equations (recall that $\mathbf{g} = [0,\, 0,\, -g]$). This has vital consequences for a hydrostatic ocean model, as the effects of the density forcing are fully captured in the horizontal momentum equations through this hydrostatic pressure. In this case, we do not solve the vertical momentum equations (4.2), but recover w through the continuity equations. As such, the density forcing cannot be communicated through the vertical momentum equation and needs to be communicated through the horizontal hydrostatic pressure gradients.

Next, let us examine the alternate definition of the hydrostatic pressure, (4.10), which is approximate since it uses ρ_0. Calculating its gradients we have

$$\begin{aligned}
\nabla_{xy} p^*_{hyd} &= \nabla_{xy} g\rho_0 \{\eta - z\}, \\
&= g\rho_0 \nabla_{xy} \eta.
\end{aligned}$$

This form does not require the calculation of any vertical integrals, which may appear to have computational advantages. However, for the vertical gradient we have

$$\begin{aligned}
\nabla_z p^*_{hyd} &= \nabla_z g\rho_0 \{\eta - z\}, \\
&= -g\rho_0.
\end{aligned}$$

This only partially cancels the density forcing term in the vertical momentum equa-

tion. As such, the remaining density forcing due to ρ' has to be communicated to the horizontal components of velocity through the continuity equation. This means that the non-hydrostatic pressure will have larger horizontal gradients (since it will contain the hydrostatic horizontal gradients due to ρ'), which may negatively impact the computational efficiency of solving the non-hydrostatic pressure.

For oceanic applications, the hydrostatic pressure defined using the full density is preferred. The mean-density form is often used for classical engineering applications, where the globally coupled 3D continuity constraint has to be imposed at all times. In this case, the density forcing is communicated to the horizontal velocity components through the continuity equation, and the additional computational and implementation effort required to vertically integrate the density gradient is not easily justified (particularly on fully unstructured meshes).

For the hydrostatic ocean case, however, a globally coupled 3D continuity constraint does not need to be imposed. Instead, the continuity can be enforced using a globally coupled 2D equation, and 1D equations for each vertical fluid column (see for example Haley and Lermusiaux (2010a)). The 2D equation is for the free-surface (or rigid-lid surface-pressure), and the 1D equations recover the vertical velocities from the continuity equation. As such, the vertical momentum equation does not need to be solved. Therefore, the full density form of the hydrostatic pressure, which captures the full density forcing in the horizontal momentum equations, is advantageous. Additionally, for terrain-following coordinate models, the vertical integration does not present a large implementation challenge because the nodes are vertically aligned. In terms of accuracy, the full-density hydrostatic pressure should better capture the geostrophic balance.

For non-hydrostatic ocean models, the full density form also presents advantages because it more completely separates the hydrostatic and non-hydrostatic pressure terms. As such, the non-hydrostatic pressure should be closer to zero, and require fewer iterations to solve numerically. Additionally, from an implementation standpoint it is useful to have the non-hydrostatic formulation be similar to the hydrostatic one, allowing a user to choose seamlessly between the two.

Now we discuss further differences between the hydrostatic pressure definitions, as they relate to the 3D divergence of the density forcing. Recall from §2.2.2 that the full pressure balances the divergent terms in the right-hand-side forcing. Here we have split the pressure into hydrostatic and non-hydrostatic components. Now, the hydrostatic pressure was defined in terms of the forcing due to density only. Thus, we can also ask whether or not our hydrostatic pressure definitions balance the divergence from the forcing due to density variations. While this is not a required property, it is desirable because then the non-hydrostatic pressure has to balance fewer terms on the right hand side. That is, if we can define a hydrostatic pressure such that

$$\nabla \cdot \left(\frac{1}{\rho_0} \nabla p_{hyd} \right) = \nabla \cdot \left(\frac{\rho}{\rho_0} \mathbf{g} \right),$$

then the non-hydrostatic pressure only has to balance the remaining forcing terms

$$\begin{aligned} \nabla \cdot (\nabla p') = \nabla_{xy} \cdot \Big(& \nabla_z \cdot \nu_z \nabla_z \mathbf{u} - \nabla_{xy} \cdot \mathbf{u}\mathbf{u} - \nabla_z \cdot w\mathbf{u} + \nabla_{xy} \cdot \nu_{xy} \nabla_{xy} \mathbf{u} - f\hat{\mathbf{k}} \times \mathbf{u} + \frac{1}{\rho_0} \mathbf{f}_{\mathbf{u}} \Big) \\ + \nabla_z \cdot \Big(& \nabla_z \cdot \nu_z \nabla_z w - \nabla_{xy} \cdot \mathbf{u} w - \nabla_z \cdot ww + \nabla_{xy} \cdot \nu_{xy} \nabla_{xy} w + \frac{1}{\rho_0} \mathbf{f}_w \Big), \end{aligned}$$

instead of all the forcing terms

$$\begin{aligned} \nabla \cdot (\nabla p') = \nabla_{xy} \cdot \Big(& \nabla_z \cdot \nu_z \nabla_z \mathbf{u} - \nabla_{xy} \cdot \mathbf{u}\mathbf{u} - \nabla_z \cdot w\mathbf{u} + \nabla_{xy} \cdot \nu_{xy} \nabla_{xy} \mathbf{u} - f\hat{\mathbf{k}} \times \mathbf{u} + \frac{1}{\rho_0} \mathbf{f}_{\mathbf{u}} \Big) \\ + \nabla_z \cdot \Big(& \nabla_z \cdot \nu_z \nabla_z w - \nabla_{xy} \cdot \mathbf{u} w - \nabla_z \cdot ww + \nabla_{xy} \cdot \nu_{xy} \nabla_{xy} w + \frac{1}{\rho_0} \mathbf{f}_w \Big) \\ + \nabla \cdot \Big(& -\frac{1}{\rho_0} \nabla p_{hyd} + \frac{\rho}{\rho_0} \mathbf{g} \Big). \end{aligned}$$

However, with either the full-density (4.9) or mean-density (4.10) definitions, the 3D divergence of the 3D hydrostatic pressure gradient will not cancel the 3D divergence due to the density forcing. To see what terms remain, we can compute the divergence of the hydrostatic pressure summed with the density forcing. For the

full-density pressure definition (4.9) we have

$$\begin{aligned}\nabla\cdot(-\nabla p_{hyd}+\rho\mathbf{g}) &= -\nabla^2_{xy}\int_z^{\eta(x,y,t)}\rho(x,\,y,\,\zeta,\,t)g\mathrm{d}\zeta+\nabla_z\cdot\{\nabla_z p_{hyd}+g(\rho_0+\rho')\},\\ &= -g\rho(x,\,y,\,\eta,\,t)\nabla^2_{xy}\eta-\int_z^{\eta(x,y,t)}\nabla^2_{xy}\rho'(x,\,y,\,\zeta,\,t)g\mathrm{d}\zeta,\\ &\approx -g\rho_0\nabla^2_{xy}\eta-\int_z^{\eta(x,y,t)}\nabla^2_{xy}\rho'(x,\,y,\,\zeta,\,t)g\mathrm{d}\zeta, \qquad (4.14)\end{aligned}$$

where $\{\nabla_z p_{hyd}+g(\rho_0+\rho')\}=0$ as defined. For the mean-density pressure definition (4.10) we have

$$\begin{aligned}\nabla\cdot(-\nabla p^*_{hyd}+\rho\mathbf{g}) &= -g\rho_0\nabla^2_{xy}\eta+\nabla_z\cdot\{\nabla_z p^*_{hyd}+g(\rho_0+\rho')\},\\ &= -g\rho_0\nabla^2_{xy}\eta+\nabla_z\cdot\rho'\mathbf{g}, \qquad (4.15)\end{aligned}$$

where $\{\nabla_z p^*_{hyd}+g(\rho_0)\}=0$. In both cases, the free-surface term does not vary in depth while the divergence of the density perturbation does (in general). However, if we consider the case of a stably stratified ($\rho'=\rho'(z)$) ocean with an unperturbed free-surface ($\eta=0$), then we see that for (4.14), $\nabla\cdot(-\nabla p_{hyd}+\rho\mathbf{g})=0$, while for (4.15), $\nabla\cdot(-\nabla p_{hyd}+\rho\mathbf{g})=\nabla_z\cdot\rho'\mathbf{g}$. This shows that the full density (4.9) definition for the hydrostatic pressure is closer to what we desire.

In the rigid-lid case, we can solve for a surface pressure component, $p_s=p_s(x,y,t)$, so as to satisfy the depth-integrated continuity constraint. This is how the surface pressure discrete correction is obtained in §4.2.4, in direct analogy to the free-surface equation obtained in §4.1.3. The surface pressure could be further decomposed into hydrostatic p_s^{hyd} and non-hydrostatic $p_s^{'}$ components. Aiming to reduce the size of $p^{'}$ as discussed above, one could have also defined a "modified surface" pressure, $p^*_s=p^*_s(x,y,t)$, so as to set (4.14) to zero in the vertical integrated sense, i.e. set the depth-integrated 3D divergence of $(-\nabla p_{hyd}-\nabla p^*_s+\rho\mathbf{g})$ to zero. One then obtains:

$$\begin{aligned}\int_{-H}^{0}\nabla\cdot(-\nabla p_{hyd}-\nabla p^*_s+\rho\mathbf{g})\mathrm{d}\zeta &= \int_{-H}^{0}-\nabla_{xy}\cdot\{\nabla_{xy}p_{hyd}+\nabla_{xy}p^*_s\}\,\mathrm{d}\zeta\\ &\Rightarrow =0, \qquad (4.16)\end{aligned}$$

where the divergence and integral sign would commute if H is a constant (flat bottom). This latter p_s^* definition is not used.

To summarize, we will use the hydrostatic pressure as defined in (4.9), where the rigid-lid case takes $\eta = 0$.

4.1.3 Free-surface derivation

The free-surface height, η, appears in the horizontal momentum equations due to the vertical integration of the density forcing term (§4.1.2). It is essentially an unknown boundary condition describing the size of the domain, and it is related to the vertical velocity kinematic boundary condition at the free-surface through

$$\frac{\mathrm{D}\eta(x,y,t)}{\mathrm{D}t} = w(x,y,z=\eta,t). \tag{4.17}$$

Subsequently, we need to derive an expression for η which we can solve directly. This is a common exercise, which we repeat here for convenience.

With the knowledge that the free-surface height can be understood as a component of the full pressure, and that the mathematical purpose of the pressure in the Boussinesq equations is to enforce the continuity constraint, we can derive an equation for η from the continuity equation:

$$\nabla_z \cdot w + \nabla_{xy} \cdot \mathbf{u} = 0.$$

Vertical integration from the ocean bottom to surface yields:

$$\int_{-H}^{\eta} \{\nabla_z \cdot w + \nabla_{xy} \cdot \mathbf{u}\}\, \mathrm{d}z = 0,$$

$$w(z=\eta) - w(z=-H) + \int_{-H}^{\eta} \nabla_{xy} \cdot \mathbf{u} \mathrm{d}z = 0.$$

Substituting the Lagrangian equations (4.17) and $w(z=-H) = w|_{-H} = -\frac{\mathrm{D}H}{\mathrm{D}t}$, and

transforming to the Eulerian form:

$$\begin{aligned}
\frac{\mathrm{D}\eta}{\mathrm{D}t} + \frac{\mathrm{D}H}{\mathrm{D}t} + \int_{-H}^{\eta} \nabla_{xy} \cdot \mathbf{u} \mathrm{d}z = 0, \\
\left\{ \frac{\partial \eta}{\partial t} + \mathbf{u}|_{\eta} \cdot \nabla_{xy} \eta \right\} + \left\{ \frac{\partial H}{\partial t} + \mathbf{u}|_{-H} \cdot \nabla_{xy} H \right\} + \int_{-H}^{\eta} \nabla_{xy} \cdot \mathbf{u} \mathrm{d}z = 0, \\
\left\{ \frac{\partial \eta}{\partial t} + \mathbf{u}|_{\eta} \cdot \nabla_{xy} \eta \right\} + \{ \mathbf{u}|_{-H} \cdot \nabla_{xy} H \} + \int_{-H}^{\eta} \nabla_{xy} \cdot \mathbf{u} \mathrm{d}z = 0,
\end{aligned} \tag{4.18}$$

where, we have assumed that the depth H of the ocean is not changing in time $\frac{\partial H}{\partial t} = 0$. We could stop here, but further simplification is possible if we use the Leibniz rule to interchange the order of integration on the last term:

$$\left\{ \frac{\partial \eta}{\partial t} + \mathbf{u}|_{\eta} \cdot \nabla_{xy} \eta \right\} + \{ \mathbf{u}|_{-H} \cdot \nabla_{xy} H \} - \mathbf{u}|_{\eta} \cdot \nabla_{xy} \eta - \mathbf{u}|_{-H} \cdot \nabla_{xy} H + \nabla_{xy} \cdot \int_{-H}^{\eta} \mathbf{u} \mathrm{d}z = 0.$$

Finally, after cancellations, we have:

$$\frac{\partial \eta}{\partial t} + \nabla_{xy} \cdot \int_{-H}^{\eta} \mathbf{u} \mathrm{d}z = 0. \tag{4.19}$$

Apart from its simplicity, the second form (4.19) has accuracy advantages over the first (4.18). In the second form, baroclinic velocities will be numerically integrated out accurately, and a non-conservative advection operator is also not required. The only disadvantage of the second form is that it is more challenging to make it numerically consistent with the 3D continuity equation (Wang et al., 2008, White et al., 2008).

4.1.4 Rigid-lid formulation

For the rigid-lid ocean model, we need to solve for the surface-pressure p_s at the rigid lid. In this case, an equation for a modified surface-pressure, p_s^*, was derived from the depth-integrated 3D divergence of the momentum equation (4.16) in §4.1.2. In our case, we will derive a classic surface-pressure in §4.2.4, starting directly from the time-discretized form of the equations and using the depth-integrated continuity equation.

4.1.5 Turbulence Closure Scheme

A study of various turbulence closer schemes is beyond the scope of this thesis. However, the form of the equations chosen include diffusion terms, since commonly-used turbulence closure schemes depend on the second derivative of velocity. Some of these schemes model turbulence as a diffusive effect, and modify the effective viscosity of the fluid. A zero-equation turbulence model would simply select a value of ν_{xy}, ν_z, κ_{xy}, and κ_z, but more sophisticated models will solve evolution equations to determine appropriate temporally and spatially variable turbulent viscosities and diffusivities. Thus, the chosen form of our equations allows the implementation of these classical turbulence closure schemes. For a survey of turbulence models see (Umlauf and Burchard, 2003, Warner et al., 2005, Umlauf and Burchard, 2005).

In this thesis, we will be using a zero-equation turbulence closure.

4.1.6 Summary of final equations and hydrostatic approximations

Here we summarize the continuous equations after decomposing the density and pressure as described in §4.1.1 and §4.1.2, and adding the equation for the free-surface from §4.1.3. Combining (4.1)-(4.3), (4.8), (4.9) and (4.19), we have the non-hydrostatic form of the equations that will be the starting point for our discretization:

$$\begin{aligned}\frac{\partial \mathbf{u}}{\partial t} - \nabla_z \cdot \nu_z \nabla_z \mathbf{u} + \nabla_{xy} p' + g\nabla_{xy}\eta = &-\frac{1}{\rho_0}\int_z^\eta g\nabla_{xy}\rho' \mathrm{d}\zeta - \nabla_{xy}\cdot\mathbf{u}\mathbf{u} - \nabla_z \cdot w\mathbf{u} \\ &+ \nabla_{xy}\cdot\nu_{xy}\nabla_{xy}\mathbf{u} - f\hat{\mathbf{k}}\times\mathbf{u} + \frac{1}{\rho_0}\mathbf{f}_{\mathbf{u}},\end{aligned} \tag{4.20}$$

$$\begin{aligned}\frac{\partial w}{\partial t} - \nabla_z \cdot \nu_z \nabla_z w + \nabla_z p' = &-\nabla_{xy}\cdot\mathbf{u}w - \nabla_z \cdot ww \\ &+ \nabla_{xy}\cdot\nu_{xy}\nabla_{xy} w + \frac{1}{\rho_0}\mathbf{f}_w,\end{aligned} \tag{4.21}$$

$$\nabla_{xy}\cdot\mathbf{u} + \nabla_z \cdot w = 0, \tag{4.22}$$

$$\frac{\partial \eta}{\partial t} + \nabla_{xy}\cdot\int_{-H}^{\eta}\mathbf{u}\mathrm{d}z = 0, \tag{4.23}$$

along with (4.4) - (4.6)

$$\frac{\partial T}{\partial t} - \nabla_z \cdot \kappa_z \nabla_z T = -\nabla_{xy} \cdot \mathbf{u}T - \nabla_z \cdot wT + \frac{1}{\rho}\nabla_{xy} \cdot \kappa_{xy}\nabla_{xy}T + f_T,$$
$$\frac{\partial S}{\partial t} - \nabla_z \cdot \kappa_z \nabla_z S = -\nabla_{xy} \cdot \mathbf{u}S - \nabla_z \cdot wS + \frac{1}{\rho}\nabla_{xy} \cdot \kappa_{xy}\nabla_{xy}S + f_S,$$
$$\rho = \rho(S,T),$$

where: $\rho = \rho_0 + \rho'$ (4.7), $p = p_{hyd} + \rho_0 p'$ (4.8), $p_{hyd} = \int_z^\eta \rho\, g\, \mathrm{d}\zeta$ (4.9) and $g\rho'\nabla_{xy}\eta$ has been neglected compared to $g\rho_0\nabla_{xy}\eta$. We note that ρ' and p' are not directly linked: ρ' is the total density anomaly with respect to ρ_0 while p' is the non-hydrostatic pressure normalized by ρ_0. We also note that in (4.20), the non-hydrostatic pressure gradient due to the free-surface is all in $\nabla_{xy}p'$ while the hydrostatic pressure gradient due to the free-surface is all contained in the sum $g\nabla_{xy}\eta + \frac{1}{\rho_0}\int_z^\eta g\nabla_{xy}\rho'\mathrm{d}\zeta$.

The boundary conditions are

$$\mathbf{u}|_{\partial\Omega_D} = \mathbf{g}_D, \qquad w|_{\partial\Omega_D} = g_D, \tag{4.24}$$
$$\left.\frac{\partial \mathbf{u}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_N} = \mathbf{g}_N, \qquad \left.\frac{\partial w}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_N} = g_N, \tag{4.25}$$
$$T|_{\partial\Omega_{D_T}} = g_{D_T}, \qquad S|_{\partial\Omega_{D_S}} = g_{D_S}, \tag{4.26}$$
$$\left.\frac{\partial T}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_{N_T}} = g_{N_T}, \qquad \left.\frac{\partial S}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_{N_S}} = g_{N_S}, \tag{4.27}$$

on Dirichlet boundaries $\partial\Omega_D$ and Neumann boundaries $\partial\Omega_N$. Note that one component of velocity may have a different boundary condition type in the same location. For example, at an inlet, u might be Dirichlet, while v could be Neumann. For open boundary conditions, we refer for example to Lermusiaux (1997) and Haley and Lermusiaux (2010a).

The hydrostatic equations are a subset of these equations. As such, we will treat the more general, non-hydrostatic pressure formulation first. The solution method for both will be similar, with the major differences being the solution of the vertical velocity. In any case, the system of equations under the hydrostatic pressure

assumption can be found by setting $p' \approx 0$, which leads to

$$\frac{\partial \mathbf{u}}{\partial t} - \nabla_z \cdot \nu_z \nabla_z \mathbf{u} + g \nabla_{xy} \eta = -\frac{1}{\rho_0} \int_z^\eta g \nabla_{xy} \rho' \mathrm{d}\zeta - \nabla_{xy} \cdot \mathbf{u}\mathbf{u} - \nabla_z \cdot w\mathbf{u} + \nabla_{xy} \cdot \nu_{xy} \nabla_{xy} \mathbf{u} - f\hat{\mathbf{k}} \times \mathbf{u} + \frac{1}{\rho_0} \mathbf{f}_\mathbf{u}, \tag{4.28}$$

$$\nabla_{xy} \cdot \mathbf{u} + \nabla_z \cdot w = 0, \tag{4.29}$$

$$\frac{\partial \eta}{\partial t} + \nabla_{xy} \cdot \int_{-H}^\eta \mathbf{u} \mathrm{d}z = 0. \tag{4.30}$$

This approximation assumes that $w \ll \mathbf{u}$ and $\frac{\partial \mathbf{u}}{\partial z} \ll \frac{\partial \mathbf{u}}{\partial x} \sim \frac{\partial \mathbf{u}}{\partial y}$, which is a thin-fluid assumption. In essence, we assume that the vertical scales are much smaller than the horizontal scales. In the open ocean, the associated error will be on the order of 1% (Cushman-Roisin and Beckers, 2011), similar to the error in the approximations of §4.1.1 for the density forcing. In the coastal ocean and for specific 3D ocean processes where horizontal scales become smaller and closer to vertical scales, the size of the non-hydrostatic terms (e.g. pressure) increase and can become significant. Examples of processes where the non-hydrostatic pressure and corresponding velocities can be significant include short wavelength high frequency internal waves, convection plumes, overturning cells as well as varied processes over steep and shallow bathymetries.

4.2 Semi-implicit Time Discretization using Projection Methods

Herein we describe the semi-implicit time-integration scheme for (4.20) - (4.23), as well as the time-integration for the temperature and salinity equations (4.4)–(4.6). As in §2, we will be using a projection method, which is commonly used for non-hydrostatic ocean models (for example Marshall et al. (1997), Fringer et al. (2006)). The difference between §2 and the present derivation is the inclusion of the free-surface equation, the different form of the density forcing, and, for the hydrostatic case, the calculation of the vertical velocity from the continuity equation. We will describe the time-discretization of a rigid-lid or free-surface, and a non-hydrostatic

or hydrostatic solver. As such, there are four different solver options in total. While they are similar, important differences do appear and need to be handled separately.

We first provide the un-split time-discretized equations §4.2.1 and then our split projection method schemes §4.2.2. Taking differences between un-split and split equations, our projection method corrections are then derived for different dynamical equations, specifically, the first velocity corrector with a free-surface §4.2.3, first velocity corrector with a rigid-lid §4.2.4, final velocity corrector with non-hydrostatic pressure §4.2.5, final velocity corrector with hydrostatic pressure §4.2.6 and, finally, the free-surface and pressure corrector boundary conditions §4.2.7. Then, we derive splitting errors in §4.2.8. Finally we give the time-discretization of the temperature and salinity equations in §4.2.9.

4.2.1 Un-split Discretization

Following §2.2, we lump the explicit terms into a right-hand side forcing function, and deal exclusively with the implicit solution step. The un-split time-discretized equations are then:

$$\frac{\mathbf{u}^{k+1}}{a\Delta t} - \nabla_z \cdot \nu_z \nabla_z \mathbf{u}^{k+1} + \nabla_{xy} p'^{,k+1} + g\nabla_{xy}\eta^{k+1} = \mathbf{F}_{\mathbf{u}}^{k,k+1}, \tag{4.31}$$

$$\frac{w^{k+1}}{a\Delta t} - \nabla_z \cdot \nu_z \nabla_z w^{k+1} + \nabla_z p'^{,k+1} = \mathbf{F}_{w}^{k,k+1}, \tag{4.32}$$

$$\nabla_{xy} \cdot \mathbf{u}^{k+1} + \nabla_z \cdot w^{k+1} = 0, \tag{4.33}$$

$$\frac{\eta^{k+1}}{a\Delta t} + \nabla_{xy} \cdot \int_{-H}^{\eta^{k+1}} \mathbf{u}^{k+1} \mathrm{d}z = \frac{\eta^{k}}{a\Delta t}, \tag{4.34}$$

for a first-order time integration scheme, where

$$\begin{aligned}
\mathbf{F}_{\mathbf{u}}^{k,k+1} &= \frac{\mathbf{u}^k}{a\Delta t} - \frac{1}{\rho_0}\int_z^{\eta^k} g\nabla_{xy}\rho'^{,k}\mathrm{d}\zeta - \nabla_{xy}\cdot\mathbf{u}^k\mathbf{u}^k - \nabla_z\cdot w^k\mathbf{u}^k \\
&\quad + \nabla_{xy}\cdot\nu_{xy}\nabla_{xy}\mathbf{u}^k - f\hat{\mathbf{k}}\times\mathbf{u}^k + \frac{1}{\rho_0}\mathbf{f}_{\mathbf{u}}^{k,k+1}, \\
\mathbf{F}_{w}^{k,k+1} &= \frac{w^k}{a\Delta t} - \nabla_{xy}\cdot\mathbf{u}^k w^k - \nabla_z\cdot w^k w^k \\
&\quad + \nabla_{xy}\cdot\nu_{xy}\nabla_{xy}w^k + \frac{1}{\rho_0}\mathbf{f}_{w}^{k,k+1},
\end{aligned}$$

with boundary conditions

$$\mathbf{u}|_{\partial\Omega_D}^{k+1} = \mathbf{g}_D, \qquad w|_{\partial\Omega_D}^{k+1} = g_D, \tag{4.35}$$

$$\left.\frac{\partial\mathbf{u}}{\partial\hat{\mathbf{n}}}\right|_{\partial\Omega_N}^{k+1} = \mathbf{g}_N, \qquad \left.\frac{\partial w}{\partial\hat{\mathbf{n}}}\right|_{\partial\Omega_N}^{k+1} = g_N, \tag{4.36}$$

on Dirichlet boundaries $\partial\Omega_D$ and Neumann boundaries $\partial\Omega_N$.

4.2.2 Projection Method Scheme

Our projection method consists of predictor steps and correction steps. The predictor will be executed in two steps for the non-hydrostatic solver, and one step for the hydrostatic solver. After these predictor steps, the corrections can be obtained by taking the difference between the final un-split equations and the split predictor equations (as we have done in §2.2.1). The free-surface non-hydrostatic scheme is summarized first, followed by the rigid-lid and hydrostatic modification. They are derived in the following subsections.

Step 1: First Velocity predictor. Consider the first step of the projection method, which is similar for the non-hydrostatic and hydrostatic case. The previous time-step values (that is, at k) are used as predictors for the free surface height and the

non-hydrostatic pressure. This leads to the velocity predictors $\bar{\mathbf{u}}$ and $\bar{w}$:

$$\frac{\bar{\mathbf{u}}^{k+1}}{a\Delta t} - \nabla_z \cdot \nu_z \nabla_z \bar{\mathbf{u}}^{k+1} + \nabla_{xy} p'^{,k} + g\nabla_{xy}\eta^k = \mathbf{F}_{\mathbf{u}}^{k,k+1}, \tag{4.37}$$

$$\frac{\bar{w}^{k+1}}{a\Delta t} - \nabla_z \cdot \nu_z \nabla_z \bar{w}^{k+1} + \nabla_z p'^{,k} = \mathbf{F}_{w}^{k,k+1}, \tag{4.38}$$

with boundary conditions

$$\bar{\mathbf{u}}|_{\partial\Omega_D}^{k+1} = \mathbf{g}_D, \qquad \bar{w}|_{\partial\Omega_D}^{k+1} = g_D, \tag{4.39}$$

$$\left.\frac{\partial \bar{\mathbf{u}}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_N}^{k+1} = \mathbf{g}_N, \qquad \left.\frac{\partial \bar{w}}{\partial \hat{\mathbf{n}}}\right|_{\partial\Omega_N}^{k+1} = g_N. \tag{4.40}$$

The hydrostatic version will not solve (4.38), but (4.37) is the same for both.

Step 2: Free-surface corrector. To update the free-surface, we solve for the free-surface corrector (could also be referred to as "free-surface-change predictor"):

$$\frac{\delta\eta^{k+1}}{a\Delta t} - \nabla_{xy} \cdot \left[a\Delta t g(\eta^k + H)\nabla_{xy}\delta\eta^{k+1}\right] = F_\eta^{k,k+1}, \tag{4.41}$$

where $F_\eta^{k,k+1} = -\nabla_{xy} \cdot \int_{-H}^{\eta^k} \bar{\mathbf{u}}^{k+1} \mathrm{d}z$, and with boundary conditions:

$$\nabla_{xy}\delta\eta|_{\partial\Omega_N}^{k+1} \cdot \hat{\mathbf{n}}_{xy} = \frac{1}{a\Delta t g(H+\eta^k)} \int_{-H}^{\eta^k} (\bar{\mathbf{u}}^{k+1} - \mathbf{g}_D) \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z, \tag{4.42}$$

$$\delta\eta|_{\partial\Omega_O}^{k+1} = g_{O_\eta}, \tag{4.43}$$

where $\partial\Omega_N$ and $\partial\Omega_O$, are boundaries with wall and open conditions for the velocities. This step is the same for both the hydrostatic and non-hydrostatic version.

Step 3, Free-surface corrections: Second velocity predictor and free-surface updates. Once the free-surface corrector is calculated, the second predictor velocities and updated free-surface elevation are then obtained by using the algebraic corrector

equations:

$$\bar{\bar{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - a\Delta t g \nabla_{xy} \delta\eta^{k+1}, \tag{4.44}$$

$$\eta^{k+1} = \delta\eta^{k+1} + \eta^{k}. \tag{4.45}$$

These equations are derived in §4.2.3. We are primarily interested in free-surface models, but the rigid-lid version of these equations are similar (see §4.2.4).

The final steps in the split time-discretization are different between the non-hydrostatic and hydrostatic solvers, so we will summarize them separately.

Step 4 for non-hydrostatic formulation: Pressure corrector. To update the non-hydrostatic pressure, we solve for the non-hydrostatic pressure corrector:

$$\nabla^2_{xy} \delta p'^{,k+1} + \nabla^2_z \delta p'^{,k+1} = \frac{\nabla_{xy} \cdot \bar{\bar{\mathbf{u}}}^{k+1}}{a\Delta t} + \frac{\nabla_z \cdot \bar{\bar{w}}^{k+1}}{a\Delta t}, \tag{4.46}$$

with boundary conditions

$$\nabla \delta p'|^{k+1}_{\partial\Omega_{NS,S}} \cdot \hat{\mathbf{n}} = \frac{1}{a\Delta t}(\bar{\bar{\mathbf{u}}}^{k+1} - \mathbf{g}_D) \cdot \hat{\mathbf{n}}_{xy} + \frac{1}{a\Delta t}(\bar{\bar{w}}^{k+1} - g_D) \cdot \hat{\mathbf{n}}_z, \tag{4.47}$$

$$\delta p'|^{k+1}_{\partial\Omega_\eta} = 0, \tag{4.48}$$

$$\delta p'|^{k+1}_{\partial\Omega_O} = g_{O_{p'}}, \tag{4.49}$$

where $\partial\Omega_{NS,S}$, $\partial\Omega_\eta$ and $\partial\Omega_\eta$, are boundaries with no-slip, slip, free-surface, and open boundary conditions for the velocities. Usually no-slip and slip boundaries are used for lateral and bottom boundaries.

Step 5 for non-hydrostatic formulation: Final velocity and pressure corrections. The velocity and pressure are then corrected using

$$\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1} - a\Delta t \nabla_{xy} \delta p'^{,k+1}, \qquad w^{k+1} = \bar{\bar{w}}^{k+1} - a\Delta t \nabla_z \delta p'^{,k+1}. \tag{4.50}$$

$$p'^{,k+1} = p'^{,k} + \delta p'^{,k+1}. \tag{4.51}$$

Next we state steps 2 and 3 for the rigid-lid case.

Step 2: Rigid-lid corrector. To update the rigid-lid, we solve for the rigid-lid

surface pressure corrector:

$$-a\Delta t H\nabla_{xy}^2\delta p_s^{k+1} - a\Delta t\nabla_{xy}\delta p_s^{k+1}\cdot\nabla_{xy}H = F_{p_s}^{k+1}, \tag{4.52}$$

where $F_{p_s}^{k+1} = -\int_{-H}^{0}\nabla_{xy}\cdot\bar{\mathbf{u}}^{k+1}\mathrm{d}z - \bar{\mathbf{u}}|_{-H}^{k+1}\cdot\nabla_{xy}H$, and with boundary conditions:

$$\nabla_{xy}\delta p_s|_{\partial\Omega_{NS}}^{k+1}\cdot\hat{\mathbf{n}}_{xy} = 0, \tag{4.53}$$

$$\nabla_{xy}\delta p_s|_{\partial\Omega_S}^{k+1}\cdot\hat{\mathbf{n}}_{xy} = \frac{1}{a\Delta t g H}\int_{-H}^{0}\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy}\mathrm{d}z, \tag{4.54}$$

$$\delta p_s|_{\partial\Omega_O}^{k+1} = g_{O_{p_s}}, \tag{4.55}$$

where $\partial\Omega_{NS}$, $\partial\Omega_S$, and $\partial\Omega_O$, are boundaries with no-slip, slip, and open boundary conditions for the velocities. This step is the same for both the hydrostatic and non-hydrostatic version.

Step 3, rigid-lid corrections: Second velocity predictor and rigid-lid surface pressure updates. Once the rigid-lid corrector is calculated, the second predictor velocities and updated rigid-lid surface pressure are then obtained by using the algebraic corrector equations:

$$\bar{\bar{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - a\Delta t\nabla_{xy}\delta p_s^{k+1}, \tag{4.56}$$

$$p_s^{k+1} = \delta p_s^{k+1} + p_s^k. \tag{4.57}$$

These equations are derived in §4.2.4. Next we state step 4 for the hydrostatic case.

Step 4 for hydrostatic formulation: Vertical velocity. For the hydrostatic form, the final horizontal velocities are calculated in Step 3, that is:

$$\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1}. \tag{4.58}$$

The vertical velocity is recovered from the 3D continuity equation:

$$\nabla_z\cdot w^{k+1} = -\nabla_{xy}\cdot\mathbf{u}^{k+1}, \tag{4.59}$$

with boundary condition

$$w^{k+1}|_{-H} = -\mathbf{u}^{k+1} \cdot \nabla_{xy} H. \tag{4.60}$$

This completes the summary of the time-split equations. In what follows we provide derivations and notes about these time-discrete equations.

4.2.3 Derivation of first velocity corrector with a free-surface

To decouple the free-surface equation from the horizontal momentum equations, we follow an approach similar to that of Dukowicz and Smith (1994), Casulli (1999), Casulli and Zanolli (2002) and Fringer et al. (2006), but with two main differences. First, since we are interested in shallower areas, we also include the non-linear terms as in Haley and Lermusiaux (2010a), i.e. we compute the integral of velocity up to an estimate of the free surface (and not up to a zero free-surface). Second, we do not decompose the velocity into barotropic and baroclinic components, even though this is very often done in classic ocean modeling.

First, some preliminaries. We wish to treat the free-surface equation implicitly for numerical stability. As such, we ideally want to use $\mathbf{u}^{k+1}$ in the vertical integral of (4.34). Since it is not yet available (or would require an implicit solve of the larger system of equations), we will use instead $\mathbf{u}^{k+1} \approx \bar{\mathbf{u}}^{k+1}$, the second predictor velocity. Remembering that the $g\eta$ term is the contribution to pressure linked to the free surface, i.e. $p'^{k+1} + g\eta^{k+1}$, we expect that the updated predictor velocity will be similar to a classic Navier-Stokes pressure correction of velocity. To show this, we start from the un-split equations (4.31) and note that:

$$p'^{k+1} + g\eta^{k+1} = (p' + g\eta)^k + \delta p'^{k+1} + g\delta\eta^{k+1}. \tag{4.61}$$

For the first predictor velocity step ((4.37)–(4.38)) that we specified, both "δ" terms were set to be null. However, for the free-surface corrector step and thus free-surface corrections step (including the "first velocity corrector" which can also be referred to

as the "second velocity predictor"), we solve for $\delta\eta^{k+1}$ implicitly. Hence, we replace $p'^{k+1} + g\eta^{k+1}$ by the predictor $p'^{k} + g\eta^{k} + g\delta\eta^{k+1}$, and thus, by analogy to classic pressure corrections, imply a velocity correction of the type (4.44)

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

where $\delta\eta^{k+1} \approx \eta^{k+1} - \eta^{k}$. With (4.44), we can now solve for $\delta\eta^{k+1}$ implicitly from (4.34). The magnitude of the splitting error due to this approximation will be examined in §4.2.8.

As outlined above, the free-surface corrector step 2 is derived by inserting (4.44) into (4.34), which gives:

$$\frac{\eta^{k+1}}{a\Delta t} + \nabla_{xy} \cdot \int_{-H}^{\eta^{k+1}} \left\{ \bar{\mathbf{u}}^{k+1} - a\Delta t g \nabla_{xy}\delta\eta^{k+1} \right\} \mathrm{d}z = \frac{\eta^{k}}{a\Delta t}.$$

To simplify the time-integration, the vertical integral term can be re-written as:

$$\int_{-H}^{\eta^{k+1}} \bar{\bar{\mathbf{u}}}^{k+1} \mathrm{d}z = \int_{-H}^{\eta^{k}} \bar{\bar{\mathbf{u}}}^{k+1} \mathrm{d}z + \int_{\eta^{k}}^{\eta^{k+1}} \bar{\bar{\mathbf{u}}}^{k+1} \mathrm{d}z. \tag{4.62}$$

The final term in (4.62) will be of order $\mathcal{O}\left(\Delta t \frac{\eta}{H}\right)$ compared to the first. As such, the final term in (4.62) is small, i.e. of higher-order than the first, and is thus dropped. This gives:

$$\begin{aligned}
&\frac{\eta^{k+1}}{a\Delta t} + \nabla_{xy} \cdot \int_{-H}^{\eta^{k}} \left\{ \bar{\mathbf{u}}^{k+1} - a\Delta t g \nabla_{xy}\delta\eta^{k+1} \right\} \mathrm{d}z = \frac{\eta^{k}}{a\Delta t}, \\
&\qquad \frac{\eta^{k+1} - \eta^{k}}{a\Delta t} - a\Delta t g \nabla_{xy} \cdot \int_{-H}^{\eta^{k}} \nabla_{xy}\delta\eta^{k+1} \mathrm{d}z = -\nabla_{xy} \cdot \int_{-H}^{\eta^{k}} \bar{\mathbf{u}}^{k+1} \mathrm{d}z, \\
&\qquad \frac{\delta\eta^{k+1}}{a\Delta t} - a\Delta t g \nabla_{xy} \cdot \left[(\eta^{k} + H) \nabla_{xy}\delta\eta^{k+1} \right] = -\nabla_{xy} \cdot \int_{-H}^{\eta^{k}} \bar{\mathbf{u}}^{k+1} \mathrm{d}z,
\end{aligned}$$

or finally, we derive (4.41):

$$\frac{\delta\eta^{k+1}}{a\Delta t} - \nabla_{xy}\cdot\left[a\Delta t g(\eta^k + H)\nabla_{xy}\delta\eta^{k+1}\right] = F_\eta^{k,k+1}, \tag{4.63}$$

where $F_\eta^{k,k+1} = -\nabla_{xy}\cdot\int_{-H}^{\eta^k}\bar{\mathbf{u}}^{k+1}\mathrm{d}z$, and with boundary conditions (4.42)–(4.43):

$$\begin{aligned}
\nabla_{xy}\delta\eta|_{\partial\Omega_{NS}}^{k+1}\cdot\hat{\mathbf{n}}_{xy} &= 0,\\
\nabla_{xy}\delta\eta|_{\partial\Omega_S}^{k+1}\cdot\hat{\mathbf{n}}_{xy} &= \frac{1}{a\Delta t g(H+\eta^k)}\int_{-H}^{\eta}\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy}\mathrm{d}z,\\
\delta\eta|_{\partial\Omega_O}^{k+1} &= g_{O_\eta},
\end{aligned}$$

where $\partial\Omega_{NS}$, $\partial\Omega_S$, and $\partial\Omega_O$, are boundaries with no-slip, slip, and open boundary conditions for the velocities. These boundary conditions are derived from the velocity boundary conditions (see §4.2.7).

To summarize, after solving for $\bar{\mathbf{u}}$ from (4.37), we solve for $\delta\eta^{k+1}$ from (4.41) above, then we correct the velocity to obtain $\bar{\bar{\mathbf{u}}}$ from (4.44), and update the free-surface height using

$$\eta^{k+1} = \delta\eta^{k+1} + \eta^k. \tag{4.64}$$

If we had treated the Coriolis terms implicitly, this correction would have been more complicated, and we would have had a contribution similar to the rotational correction term in §2.2.1. The reason why we do not have a rotational correction term in the present scheme (step 3) is because only the vertical diffusion term is treated implicitly, and $\nabla_z\delta\eta^{k+1} = 0$ since η does not have any vertical dependence. Next we repeat this step, but for the rigid-lid formulation.

4.2.4 Derivation of first velocity corrector with a rigid-lid

For the rigid-lid formulation, we wish to impose the 2D (vertically integrated) continuity constraint. That is, after this step we want the depth-integrated continuity

equation to be zero. In this case, we again define a second velocity predictor, (4.56)

$$\bar{\bar{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - a\Delta t \nabla_{xy} \delta p_s^{k+1}, \tag{4.65}$$

where here we use p_s for the surface pressure. To derive an equation for δp_s^{k+1}, we set the depth-integrated continuity of the corrected velocity equal to zero:

$$\begin{aligned}
\int_{-H}^{0} \nabla_{xy} \cdot \bar{\bar{\mathbf{u}}}^{k+1} \mathrm{d}z + \int_{-H}^{0} \nabla_z \cdot \bar{\bar{w}}^{k+1} \mathrm{d}z = 0, \\
\int_{-H}^{0} \nabla_{xy} \cdot \left\{ \bar{\mathbf{u}}^{k+1} - a\Delta t \nabla_{xy} \delta p_s^{k+1} \right\} \mathrm{d}z + \bar{\bar{w}}^{k+1}|_0 - \bar{\bar{w}}^{k+1}|_{-H} = 0, \\
-a\Delta t H \nabla_{xy}^2 \delta p_s^{k+1} + \int_{-H}^{0} \nabla_{xy} \cdot \bar{\mathbf{u}}^{k+1} \mathrm{d}z + \bar{\bar{w}}^{k+1}|_0 - \bar{\bar{w}}^{k+1}|_{-H} = 0.
\end{aligned}$$

We want to eliminate the vertical velocity from these equations. At the rigid-lid, the vertical velocity will be perpendicular to the surface, and for no flow across this boundary, it has to be zero, $\bar{\bar{w}}^{k+1}|_0 = 0$. At the bottom, the vertical velocity will be non-zero for slip boundary conditions and sloped boundaries. The bottom vertical velocity can be written in terms of the horizontal velocity components, by using the no normal flow boundary condition, $\mathbf{u}|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_{xy}|_{-H} + w|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_z|_{-H} = 0$:

$$-\bar{\bar{w}}^{k+1}|_{-H} = \bar{\bar{\mathbf{u}}}|_{-H}^{k+1} \cdot \left. \frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z} \right|_{-H}, \tag{4.66}$$

$$= \bar{\mathbf{u}}|_{-H}^{k+1} \cdot \nabla_{xy} H - a\Delta t \nabla_{xy} \delta p_s^{k+1} \cdot \nabla_{xy} H, \tag{4.67}$$

where we have used the mathematical identity $\left. \frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z} \right|_{-H} = \nabla_{xy} H$ (see also (4.18)). Substituting and re-arranging we obtain the equation for the rigid-lid pressure, (4.52):

$$-a\Delta t H \nabla_{xy}^2 \delta p_s^{k+1} - a\Delta t \nabla_{xy} \delta p_s^{k+1} \cdot \nabla_{xy} H = F_{p_s}^{k+1}, \tag{4.68}$$

where

$$F_{p_s}^{k+1} = -\int_{-H}^{0} \nabla_{xy} \cdot \bar{\mathbf{u}}^{k+1} \mathrm{d}z - \bar{\mathbf{u}}|_{-H}^{k+1} \cdot \nabla_{xy} H,$$

with boundary conditions

$$\nabla_{xy}\delta p_s|^{k+1}_{\partial\Omega_{NS}} \cdot \hat{\mathbf{n}}_{xy} = 0, \tag{4.69}$$

$$\nabla_{xy}\delta p_s|^{k+1}_{\partial\Omega_S} \cdot \hat{\mathbf{n}}_{xy} = \frac{1}{a\Delta t g H}\int_{-H}^{0} \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z, \tag{4.70}$$

$$\delta p_s|^{k+1}_{\partial\Omega_O} = g_{O_{p_s}}, \tag{4.71}$$

where $\partial\Omega_{NS}$, $\partial\Omega_S$, and $\partial\Omega_O$, are boundaries with no-slip, slip, and open boundary conditions for the velocities. As in §4.2.3, these boundary conditions are again derived from the velocity boundary conditions (see §4.2.7).

We now provide a few notes. First, $F^{k+1}_{p_s}$ can also be written in different forms, whatever may be computationally convenient. For example, we could have solved for

$$-a\Delta t H \nabla^2_{xy}\delta p^{k+1}_s = F^{k+1}_{p_s},$$

where

$$F^{k+1}_{p_s} = -\int_{-H}^{0} \left\{\nabla_{xy}\cdot\bar{\mathbf{u}}^{k+1} + \nabla_z \cdot \bar{\bar{w}}^{k+1}\right\} \mathrm{d}z.$$

With this form for no-slip boundary conditions, it is tempting to set $\bar{\bar{w}}^{k+1}|_{-H}$ equal to zero in the equation, since with no-slip boundaries, all components of the final velocity should be zero. However, this would not be correct for the predictor velocity $\bar{\bar{w}}^{k+1}|_{-H}$, since the no-slip boundary condition cannot be satisfied at this step. This is because the rigid-lid correction does not change in depth. The quantity $a\Delta t\nabla_{xy}\delta p^{k+1}_s \cdot \nabla_{xy}H$ from (4.68) is what modifies $\bar{w}^{k+1}|_{-H}$ such that $\bar{\bar{\mathbf{u}}}|^{k+1}_{-H} \cdot \hat{\mathbf{n}}_{xy}|_{-H} + \bar{\bar{w}}|^{k+1}_{-H} \cdot \hat{\mathbf{n}}_z|_{-H}$ equals zero at the bottom. This happens because when the horizontal velocity components are corrected with a constant-in-depth correction, they will no longer satisfy the no normal flow condition at the bottom sloped boundary when using the $\bar{w}^{k+1}|_{-H}$ velocity. The $a\Delta t\nabla_{xy}\delta p^{k+1}_s \cdot \nabla_{xy}H$ quantity, then, effectively corrects the vertical velocity at the bottom boundary to satisfy the no normal flow boundary condition.

Second, note that we could have taken the vertical integration first, followed by

the divergence. While the choice does not matter mathematically, numerically, there are consequences. Since the finite element operators do not commute (White et al., 2008), the order of the integration and divergence matters. Because we want the 2D continuity operator to be numerically consistent with the 3D continuity operator, we used the chosen form, which is naturally numerically consistent.

4.2.5 Derivation of final velocity corrector with non-hydrostatic pressure

In the non-hydrostatic pressure formulation, the 3D continuity constraint, which ensures that the final velocity is divergence-free, is imposed by the non-hydrostatic pressure. At this point, we have solved for the second velocity predictor, either using a free-surface or a rigid-lid approximation. This velocity predictor satisfies a depth-integrated continuity constraint. In the spatial discretization, we will make use of this fact. For the temporal discretization, we introduce the final corrector equations (4.50) (as was done in §4.2.3 for (4.44) with (4.61)):

$$\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1} - a\Delta t \nabla_{xy} \delta p'^{,k+1}, \qquad w^{k+1} = \bar{\bar{w}}^{k+1} - a\Delta t \nabla_z \delta p'^{,k+1}.$$

Next, we provide some preliminaries before the derivation proper. First $\bar{\bar{w}}^{k+1} = \bar{w}^{k+1}$ in the interior of the domain, while at the bottom sloped boundary, $\bar{\bar{w}}^{k+1}$ can be obtained from (4.38), $\bar{\bar{w}}|_{-H}^{k+1} = -\bar{\bar{\mathbf{u}}}|_{-H}^{k+1} \cdot \nabla_{xy} H$ (see (4.66)), and at the free-surface we can recalculate the vertical velocity which should be consistent with the free-surface equation (4.17), i.e. $\bar{\bar{w}}|_{\eta}^{k+1} = w|_{\eta}^{k+1} = \frac{\delta\eta^{k+1}}{a\Delta t} + \bar{\bar{\mathbf{u}}}|_{-H}^{k+1} \nabla_{xy} \eta^{k+1}$ (see §4.1.3). In practice, the recalculated vertical velocity at the free surface may not be numerically consistent with the vertical velocity recovered from the free-surface equation: the error comes from the inconsistency of the numerical operators (White et al., 2008). Also, we never actually calculate the updated bottom boundary velocity, but impose it through the Poisson equation for the pressure correction.

Second, we could, alternatively, take $\bar{\bar{w}}^{k+1} = \bar{w}^{k+1} = 0$ if we did not want to calculate the vertical momentum equation. In this case, the vertical non-hydrostatic pres-

sure gradient predictor would also be zero, which means that $\nabla_z p'^{,k+1} = \nabla_z \delta p'^{,k+1}$, i.e it is the corrector step that adds all of the non-hydrostatic effects. With this approach, the non-hydrostatic solver will more-closely resemble a hydrostatic one. Therefore, this could be useful for adding non-hydrostatic terms to an existing hydrostatic model.

For the derivation proper, we simply use the property that the final velocity should be divergence-free, (as would be for the original equations (4.20)–(4.23), and un-split equation (4.31)–(4.34)). Hence, taking the 3D divergence of (4.50) (as in §2.2), we obtain the equation for the non-hydrostatic pressure corrector (4.46):

$$\nabla^2_{xy} \delta p'^{,k+1} + \nabla^2_z \delta p'^{,k+1} = \frac{\nabla_{xy} \cdot \bar{\bar{\mathbf{u}}}^{k+1}}{a\Delta t} + \frac{\nabla_z \cdot \bar{\bar{w}}^{k+1}}{a\Delta t},$$

with boundary conditions (4.47)–(4.49)

$$\begin{aligned} \nabla \delta p'|^{k+1}_{\partial\Omega_{NS,S}} \cdot \hat{\mathbf{n}} &= \frac{1}{a\Delta t} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + \frac{1}{a\Delta t} \bar{\bar{w}}^{k+1} \cdot \hat{\mathbf{n}}_z, \\ \delta p'|^{k+1}_{\partial\Omega_\eta} &= 0, \\ \delta p'|^{k+1}_{\partial\Omega_O} &= g_{O_{p'}}, \end{aligned}$$

where $\partial\Omega_{NS,S}$, $\partial\Omega_\eta$, and $\partial\Omega_O$, are boundaries with no-slip, slip, open boundary, and free-surface conditions for the velocities. Again, these boundary conditions are derived from the velocity boundary conditions (see §4.2.7). The non-uniform value for $\partial\Omega_{NS,S}$ ensures no normal flow at solid boundaries. Normally, a uniform Neumann condition for the pressure-correction can be used on no-slip boundaries. However, after solving for the rigid-lid or free-surface, $\bar{\bar{\mathbf{u}}}^{k+1}$ only satisfies the depth-integrated continuity equation. Since $\bar{\bar{\mathbf{u}}}^{k+1}$ is corrected using a 2D field (that is, constant in depth) the no normal flow velocity boundary condition cannot be enforced on sloped bottoms. As such, the non-hydrostatic pressure has to correct the bottom boundary velocities.

The new non-hydrostatic pressure is recovered from (4.51)

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

For ocean applications the rotational correction term is small and we will neglect it, $\mathbf{r}_{\text{cor}}^{k+1} = -\nabla_z \cdot \nu_z \nabla_z a \Delta t \nabla \delta p'^{,k+1} \approx 0$ (see §3.8.3).

We note that once the velocities are corrected using (4.50), the vertical velocity at the free-surface may not be consistent with the free-surface height calculated. The time splitting errors, as well as inconsistencies in the numerical operators can lead to differences between the vertical velocity calculated here and the free-surface height calculated in (4.41). However, for the non-hydrostatic solver this numerical inconsistency is not a major concern, since the free-surface calculation is not a required step. The free-surface calculation is a constant-depth update to the old total pressure p^k, which enforces depth-integrated continuity, but the pressure p^k still requires a depth-varying update. As such, the constant-depth update from the free-surface could be absorbed into the non-hydrostatic pressure correction. Thus, the advantage of a numerically consistent free-surface calculation is that it may give an improved pressure predictor $p^k + \rho_0 g \nabla_{xy} \delta\eta^{k+1}$, which may affect the computational efficiency of solving for the non-hydrostatic pressure corrector (that is, reduce the number of iterations until convergence for an iterative matrix solver). Additionally, the vertical displacement of the domain will be more accurate with a consistent free-surface. Nonetheless, the accuracy and stability of the algorithm should not be significantly affected.

4.2.6 Derivation of final velocity corrector with hydrostatic pressure

In the hydrostatic formulation, the second horizontal predictor velocity which satisfies the 2D depth-integrated continuity equation will not be modified by the 3D continuity constraint. As such, we directly use (4.58)

$$\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1}.$$

At this point, the vertical velocity has not been calculated. It is normally recovered from the 3D continuity equation (4.59):

$$\nabla_z \cdot w^{k+1} = -\nabla_{xy} \cdot \mathbf{u}^{k+1},$$

with boundary condition (4.60)

$$w^{k+1}|_{-H} = -\mathbf{u}^{k+1} \cdot \nabla_{xy} H.$$

Even though the above derivation is straightforward, some discussions are warranted. First, we discuss the consistency of operators and boundary conditions. Instead of using the bottom boundary condition, the top boundary condition could be used, if the depth-integrated continuity equation is numerically consistent with this 3D continuity equation. For the rigid-lid hydrostatic pressure case, numerical consistency of the operators is crucial because the velocity at the top of the domain has to be zero, and at the bottom it has to enforce the no normal flow condition. If either of these are not satisfied, mass will not be conserved. As we can only enforce either the top or bottom boundary condition, the vertical integration has to be numerically consistent. For the free-surface case, numerical consistency is not as important, because the only effect will be on the velocity of the free-surface, which is allowed to vary. If the numerical inconsistency error is small, the results will be acceptable, since the mass conservation is in that case not affected.

Second, we discuss the differences in pressure between the hydrostatic and non-hydrostatic formulations. While the hydrostatic form does not correct the vertical velocity using a pressure (as in the non-hydrostatic case), we can formulate the correction through a pressure to show this difference between the hydrostatic and non-hydrostatic approaches. Consider the non-hydrostatic velocity correction step (4.50) and the pressure correction equation (4.46), but we let $\bar{\bar{w}}^{k+1} = 0$, and

$a\Delta t\nabla_{xy}\delta p'^{,k+1} \approx 0$. Then these equations become

$$\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1} \qquad\qquad w^{k+1} = -a\Delta t\nabla_z \delta p'^{,k+1}, \tag{4.72}$$

and

$$\nabla_z^2 \delta p'^{,k+1} = \frac{\nabla_{xy} \cdot \bar{\bar{\mathbf{u}}}^{k+1}}{a\Delta t}. \tag{4.73}$$

with boundary conditions

$$\nabla_z \delta p'|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_z = -\frac{1}{a\Delta t}\bar{\bar{\mathbf{u}}}|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_{xy}, \tag{4.74}$$

$$\delta p'|_{\partial\Omega_\eta}^{k+1} = 0. \tag{4.75}$$

This form highlights the difference between the non-hydrostatic and hydrostatic formulations. We assume in the hydrostatic case that the horizontal, depth-varying pressure-correction gradients are small compared to the vertical gradients. Recall, the depth-integrated horizontal pressure gradients are calculated from the 2D depth-integrated continuity equation in the previous step. As such, we simply eliminate these gradients from the equation. Also, we use a zero vertical velocity predictor, which could also be done in the non-hydrostatic case. The only difference, then, is that we have dropped the horizontal gradient terms in the pressure-correction equation. The hydrostatic approximation drastically increases the computational efficiency of enforcing the 3D continuity constraint; if the horizontal gradients are kept, a 3D globally coupled set of equations need to be solved, whereas the hydrostatic approximation allows 1D uncoupled equations to be solved.

The correction specified through the pressure (4.73) also has numerical consequences. In the first case (4.59) we solve a first-order differential equation, whereas in the second case (4.73) we solve a second-order differential equation. Because of the numerical consistency consideration, the vertical integration scheme used to calculate the vertical velocity should match the integration scheme used to calculate the depth-integrated divergence. As such, for hydrostatic computations, we will use the

(4.73) form to calculate the vertical velocity.

4.2.7 Free-surface and pressure corrector boundary conditions

In §2.2.3 we showed that the uniform Neumann boundary condition for the pressure-corrector is consistent with the pressure boundary conditions that follow from the momentum equations. However, in §2.2.3, we were treating all components of the diffusion term implicitly, whereas here we only treat the vertical diffusion term implicitly. As a result, the boundary conditions for the pressure-corrector can be affected.

In this section we derive new boundary conditions for the free-surface and pressure-corrector equations, focusing on lateral boundary conditions ($\mathbf{u}\cdot\hat{\mathbf{n}}_{xy} = 0$ for no normal flow). Then we show that they are consistent with the boundary conditions from the un-split equations.

To explain the issue, consider the first velocity predictor (4.37) projected on to the normal of the boundary:

$$\frac{\bar{\mathbf{u}}^{k+1}}{a\Delta t}\cdot\hat{\mathbf{n}}_{xy} - \nabla_z\cdot\nu_z\nabla_z\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy} + \nabla_{xy}p'^{,k}\cdot\hat{\mathbf{n}}_{xy} + g\nabla_{xy}\eta^k\cdot\hat{\mathbf{n}}_{xy} = \mathbf{F}_{\mathbf{u}}^{k,k+1}\cdot\hat{\mathbf{n}}_{xy}, \tag{4.76}$$

re-arranging

$$\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy} - a\Delta t\nabla_z\cdot\nu_z\nabla_z\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy} = a\Delta t\left(-\nabla_{xy}p'^{,k} - g\nabla_{xy}\eta^k + \mathbf{F}_{\mathbf{u}}^{k,k+1}\right)\cdot\hat{\mathbf{n}}_{xy}. \tag{4.77}$$

Since we do not treat all of the diffusion terms implicitly, we cannot impose the final lateral no normal flow boundary conditions on $\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}}_{xy}$ using (4.37) as was done in §2.2.3. This is because the spatial discretization of the vertical diffusion term uses the z-component of the normal, which will be zero on vertical faces. As such, the boundary conditions of the first predictor velocity, $\bar{\mathbf{u}}^{k+1}\cdot\hat{\mathbf{n}} \neq 0$ on some slip and no-slip boundaries. The final no-flux boundary condition, then, has to be imposed

through the free-surface and pressure corrector equations.

Derivation of no normal flow boundary conditions: First we consider the lateral no-penetration boundary conditions for the free-surface corrections (step 3 in §4.2.2). To obtain a lateral boundary condition, we use the first first corrector equation (4.44) and project it on to the normal $\hat{\mathbf{n}}_{xy}$ of a lateral boundary:

$$\begin{aligned} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} &= \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} - a\Delta t g \nabla_{xy} \delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy}, \\ \Rightarrow \nabla_{xy} \delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} &= \frac{1}{a\Delta t g} \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} - \frac{1}{a\Delta t g} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy}. \end{aligned} \tag{4.78}$$

Since the free-surface corrections (4.44)–(4.45) do not vary in depth, the second velocity predictor can only satisfy the depth-integrated lateral no normal flow condition $\int_{-H}^{\eta^k} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z = 0$. Taking the vertical integral of (4.78):

$$\begin{aligned} \int_{-H}^{\eta^k} \nabla_{xy} \delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z &= \int_{-H}^{\eta^k} \frac{1}{a\Delta t g} \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z - \int_{-H}^{\eta^k} \frac{1}{a\Delta t g} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z \\ (H + \eta^k) \nabla_{xy} \delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} &= \int_{-H}^{\eta^k} \frac{1}{a\Delta t g} \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z, \end{aligned}$$

and re-arranging we recover (4.42)

$$\nabla_{xy} \delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} = \frac{1}{a\Delta t g (H + \eta^k)} \int_{-H}^{\eta^k} \bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} \mathrm{d}z.$$

Second, for the non-hydrostatic pressure corrector (step 4 in §4.2.2) and any no normal flow boundary condition (lateral and bottom), we start from (4.50) projected on to the 3D normal of the boundary:

$$\mathbf{u}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + w^{k+1} \cdot \hat{\mathbf{n}}_z = \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + \bar{\bar{w}}^{k+1} \cdot \hat{\mathbf{n}}_z - a\Delta t \nabla \delta p'^{,k+1} \cdot \hat{\mathbf{n}},$$

re-arranging, and setting $\mathbf{u}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + w^{k+1} \cdot \hat{\mathbf{n}}_z = 0$ (no penetration boundary condition for the final velocity), we recover (4.47)

$$\nabla \delta p'^{,k+1} \cdot \hat{\mathbf{n}} = \frac{1}{a\Delta t} \bar{\bar{\mathbf{u}}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + \frac{1}{a\Delta t} \bar{\bar{w}}^{k+1} \cdot \hat{\mathbf{n}}_z.$$

Note that both (4.42) and (4.47) reduce to the original zero Neumann boundary conditions for the corrector if the no-flux boundary condition is satisfied by the predictor velocities.

No normal flow boundary condition consistency proof: Having derived boundary conditions for the free-surface and pressure correctors, we now have to verify that the boundary conditions imposed implicitly on the free-surface and non-hydrostatic pressure are consistent. The no normal flow boundary condition from the momentum equation can be found by projecting the un-split equation (4.31)–(4.32) on to the normal of the boundary

$$\frac{\mathbf{u}^{k+1}}{a\Delta t}\cdot\hat{\mathbf{n}}_{xy}+\frac{w^{k+1}}{a\Delta t}\cdot\hat{\mathbf{n}}_z-\nabla_z\cdot\nu_z\nabla_z\mathbf{u}^{k+1}\cdot\hat{\mathbf{n}}_{xy}-\nabla_z\cdot\nu_z\nabla_z w^{k+1}\cdot\hat{\mathbf{n}}_z+(\nabla p'^{,k+1}+g\nabla_{xy}\eta^{k+1})\cdot\hat{\mathbf{n}}$$
$$=\mathbf{F}_{\mathbf{u}}^{k,k+1}\cdot\hat{\mathbf{n}}_{xy}+\mathbf{F}_{w}^{k,k+1}\cdot\hat{\mathbf{n}}_z,$$

and re-arranging:

$$\begin{aligned}\nabla p'^{,k+1}\cdot\hat{\mathbf{n}}+g\nabla_{xy}\eta^{k+1}\cdot\hat{\mathbf{n}}_{xy}=&\left(\nabla_z\cdot\nu_z\nabla_z\mathbf{u}^{k+1}+\mathbf{F}_{\mathbf{u}}^{k,k+1}\right)\cdot\hat{\mathbf{n}}_{xy}\\&+\left(\nabla_z\cdot\nu_z\nabla_z w^{k+1}\cdot\hat{\mathbf{n}}_z+\mathbf{F}_{w}^{k,k+1}\right)\cdot\hat{\mathbf{n}}_z,\end{aligned}\tag{4.79}$$

where we have enforced the desired nor normal flow condition $\mathbf{u}^{k+1}\cdot\hat{\mathbf{n}}_{xy}+w^{k+1}\cdot\hat{\mathbf{n}}_z=0$ on the final velocity. We have to show that the implicit boundary conditions imposed by (4.42) and (4.47) on the corrections have the same form as (4.79). To do so, we will substitute from the time-split equations to recover (4.79).

Starting with the non-hydrostatic pressure, take the gradient of (4.51) and project it on to the normal of the boundary

$$\nabla p'^{,k+1}\cdot\hat{\mathbf{n}}=\nabla p'^{,k}\cdot\hat{\mathbf{n}}+\nabla\delta p'^{,k+1}\cdot\hat{\mathbf{n}}.$$

Substituting for our derived boundary condition for the pressure-corrector, (4.47), we obtain

$$\nabla p'^{,k+1}\cdot\hat{\mathbf{n}}=\nabla p'^{,k}\cdot\hat{\mathbf{n}}+\frac{1}{a\Delta t}\bar{\bar{\mathbf{u}}}^{k+1}\cdot\hat{\mathbf{n}}_{xy}+\frac{1}{a\Delta t}\bar{\bar{w}}^{k+1}\cdot\hat{\mathbf{n}}_z.\tag{4.80}$$

Then substituting for the second velocity predictor from its update equation, (4.44),

we have

$$\nabla p'^{,k+1} \cdot \hat{\mathbf{n}} = \nabla p'^{,k} \cdot \hat{\mathbf{n}} + \frac{1}{a\Delta t}\bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + \frac{1}{a\Delta t}\bar{w}^{k+1} \cdot \hat{\mathbf{n}}_z - g\nabla_{xy}\delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy}. \quad (4.81)$$

To obtain the left-hand-side of (4.79) , we sum (4.81) with $g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy}$

$$\begin{aligned} \nabla p'^{,k+1} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} = \nabla p'^{,k} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} - g\nabla_{xy}\delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} \\ + \frac{1}{a\Delta t}\bar{\mathbf{u}}^{k+1} \cdot \hat{\mathbf{n}}_{xy} + \frac{1}{a\Delta t}\bar{w}^{k+1} \cdot \hat{\mathbf{n}}_z. \end{aligned} \quad (4.82)$$

Using (4.37) and (4.38), we substitute for $\frac{1}{a\Delta t}\bar{\mathbf{u}}^{k+1}$ and $\frac{1}{a\Delta t}\bar{w}^{k+1}$ in (4.82):

$$\begin{aligned} \nabla p'^{,k+1} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} = \nabla p'^{,k} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} - g\nabla_{xy}\delta\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} \\ - \nabla p'^{,k} \cdot \hat{\mathbf{n}} - g\nabla_{xy}\eta^{k} \cdot \hat{\mathbf{n}}_{xy} \\ + \left(\nabla_z \cdot \nu_z \nabla_z \bar{\mathbf{u}}^{k+1} + \mathbf{F}_{\mathbf{u}}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_{xy} \\ + \left(\nabla_z \cdot \nu_z \nabla_z \bar{w}^{k+1} + \mathbf{F}_{w}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_z. \end{aligned} \quad (4.83)$$

Finally, substituting for $\delta\eta^{k+1}$ from (4.45) and canceling terms, we have

$$\begin{aligned} \nabla p'^{,k+1} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} &= \nabla p'^{,k} \cdot \hat{\mathbf{n}} + g\nabla_{xy}\eta^{k+1} \cdot \hat{\mathbf{n}}_{xy} - g\nabla_{xy}(\eta^{k+1} - \eta^k) \cdot \hat{\mathbf{n}}_{xy} \\ &\quad - \nabla p'^{,k} \cdot \hat{\mathbf{n}} - g\nabla_{xy}\eta^{k} \cdot \hat{\mathbf{n}}_{xy} \\ &\quad + \left(\nabla_z \cdot \nu_z \nabla_z \bar{\mathbf{u}}^{k+1} + \mathbf{F}_{\mathbf{u}}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_{xy} \\ &\quad + \left(\nabla_z \cdot \nu_z \nabla_z \bar{w}^{k+1} + \mathbf{F}_{w}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_z \\ &= \left(\nabla_z \cdot \nu_z \nabla_z \bar{\mathbf{u}}^{k+1} + \mathbf{F}_{\mathbf{u}}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_{xy} \\ &\quad + \left(\nabla_z \cdot \nu_z \nabla_z \bar{w}^{k+1} + \mathbf{F}_{w}^{k,k+1}\right) \cdot \hat{\mathbf{n}}_z. \end{aligned} \quad (4.84)$$

The difference between the split and un-split boundary conditions are only due to the rotational correction term which we have neglected, since it is small for ocean applications. To see the error $\mathcal{E}_{\text{bc}}$, let us subtract (4.84) from (4.79):

$$\mathcal{E}_{\text{bc}} = \left(\nabla_z \cdot \nu_z \nabla_z (\mathbf{u}^{k+1} - \bar{\mathbf{u}}^{k+1})\right) \cdot \hat{\mathbf{n}}_{xy} + \left(\nabla_z \cdot \nu_z \nabla_z (w^{k+1} - \bar{w}^{k+1})\right) \cdot \hat{\mathbf{n}}_z. \quad (4.85)$$

This error will be of order $\mathcal{E}_{\text{bc}} \sim \mathcal{O}(\nu_z \Delta t)$, since there are only $\mathcal{O}(\Delta t)$ corrections between $\mathbf{u}^{k+1}$ and $\bar{\mathbf{u}}^{k+1}$. When ν_z is constant (i.e. no spatial variance), this error can be eliminated by applying the rotational correction to the non-hydrostatic pressure. However, for practical ocean simulations ν_z is small, and the inclusion of the rotational correction does not seem to make a noticeable difference.

Thus, we have shown that our boundary conditions for the free-surface and pressure corrector equation are consistent with the un-split equations.

4.2.8 Splitting errors

In this section, the magnitude of the various time-splitting errors are examined one-by-one. We first show the errors between the splitting of the depth-integrated 2D continuity equation, which is different between the hydrostatic and non-hydrostatic formulations. Then, we show the error of the final projection step from the non-hydrostatic case. We note that we derive the splitting error in each step assuming that it is not affected by errors in the other steps, i.e. we don't evaluate the total (nonlinear) splitting error.

Splitting error for steps 2–3 with hydrostatic pressure: From §3.8.2 we argued that the magnitude of the splitting errors for the velocity and pressure were $\mathcal{O}\left(\frac{\Delta t^2}{\text{Re}}\right)$ and $\mathcal{O}\left(\frac{\Delta t}{\text{Re}}\right)$. In that section, we also saw that for the case examined, the splitting errors were from the boundary conditions of the implicit diffusion equations when the right-hand-side forcing was divergence-free. Since in the ocean case we are only solving implicitly for the vertical diffusion, and we only impose the depth-integrated continuity constraint at steps 2-3 of the time-integration procedure (§4.2.2), we do not expect this step to introduce any additional splitting errors. That is, since the forcing term of the depth-integrated equations are completely explicit, the 2D pressure-correction will be projecting out the divergence of the explicit terms only. Therefore, the hydrostatic pressure form should not have a splitting error.

Splitting error for steps 2–4 with non-hydrostatic pressure and free-surface: Next we focus on the splitting errors for the non-hydrostatic solver, involving steps 2,3 and 4 of §4.2.2. For this, we begin by evaluating the splitting errors in step 2, i.e. the

errors introduced in the free-surface calculation when the non-hydrostatic pressure component is neglected. We should have used $\mathbf{u}^{k+1}$ instead of $\bar{\bar{\mathbf{u}}}^{k+1}$ in (4.41). To evaluate the error, we substitute for $\mathbf{u}^{k+1} = \bar{\bar{\mathbf{u}}}^{k+1} - a\Delta t \nabla_{xy}\delta p'^{,k+1}$ in the un-split equation (4.34) to derive after some manipulations (as in §4.2.3)

$$\frac{\delta\eta^{k+1}}{a\Delta t} - \nabla_{xy}\cdot\left[a\Delta t g(\eta^k + H)\nabla_{xy}\delta\eta^{k+1}\right] = -\nabla_{xy}\cdot\int_{-H}^{\eta^k}\left\{\bar{\mathbf{u}}^{k+1} - a\Delta t\nabla_{xy}\delta p'^{,k+1}\right\}\mathrm{d}z,$$

where we have neglected the integral from η^k to η^{k+1} as in (4.62). Hence, the splitting the splitting error $\mathcal{E}$ is due to the neglected term

$$\begin{aligned}\mathcal{E} &= \nabla_{xy}\cdot\int_{-H}^{\eta^k} a\Delta t\nabla_{xy}\delta p'^{,k+1}\mathrm{d}z \\ &= a\Delta t\left[\nabla_{xy}\eta^k\cdot\nabla_{xy}\delta p'^{,k+1}|_{\eta} + \nabla_{xy}H\cdot\nabla_{xy}\delta p'^{,k+1}|_{-H} + \int_{-H}^{\eta^k}\nabla^2_{xy}\delta p'^{,k+1}\mathrm{d}z\right].\end{aligned}$$

Substituting from (4.46), the last term becomes

$$\begin{aligned}\int_{-H}^{\eta^k}\nabla^2_{xy}\delta p'^{,k+1}\mathrm{d}z &= \int_{-H}^{\eta^k}\left\{-\nabla^2_z\delta p'^{,k+1} + \frac{\nabla_{xy}\cdot\bar{\bar{\mathbf{u}}}^{k+1}}{a\Delta t} + \frac{\nabla_z\cdot\bar{\bar{w}}^{k+1}}{a\Delta t}\right\}\mathrm{d}z, \\ &= -\nabla_z\delta p'^{,k+1}|_{\eta} + \nabla_z\delta p'^{,k+1}|_{-H},\end{aligned}$$

since the depth-integrated divergence of the second predictor velocity is zero. Using the mathematical identities $\nabla_{xy}\eta^k = -\left.\frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z}\right|_{\eta^k}$ and $\nabla_{xy}H = \left.\frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z}\right|_{-H}$ (for outward facing normals), the error becomes

$$\begin{aligned}\mathcal{E} &= a\Delta t\left[-\left.\frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z}\right|_{\eta^k}\cdot\nabla_{xy}\delta p'^{,k+1}|_{\eta} + \left.\frac{\hat{\mathbf{n}}_{xy}}{\hat{n}_z}\right|_{-H}\cdot\nabla_{xy}\delta p'^{,k+1}|_{-H} - \nabla_z\delta p'^{,k+1}|_{\eta} + \nabla_z\delta p'^{,k+1}|_{-H}\right], \\ &= a\Delta t\left[\left.\frac{-\hat{\mathbf{n}}\cdot\nabla\delta p'^{,k+1}}{\hat{n}_z}\right|_{\eta^k} + \left.\frac{\hat{\mathbf{n}}\cdot\nabla\delta p'^{,k+1}}{\hat{n}_z}\right|_{-H}\right]\end{aligned}$$

and using the bottom boundary conditions (4.47) we finally have

$$\mathcal{E} = -a\Delta t\left.\frac{\hat{\mathbf{n}}\cdot\nabla\delta p'^{,k+1}}{\hat{n}_z}\right|_{\eta^k} + \frac{\bar{\bar{\mathbf{u}}}|^{k+1}_{-H}\cdot\hat{\mathbf{n}}_{xy} + \bar{\bar{w}}|^{k+1}_{-H}\cdot\hat{\mathbf{n}}_z}{\hat{n}_z}.$$

Note that the second term will be of $\mathcal{O}(\Delta t)$, since the velocity at the previous time step would have satisfied the no normal flow boundary condition $\mathbf{u}|_{-H}^{k} \cdot \hat{\mathbf{n}}_{xy} + w|_{-H}^{k} \cdot \hat{\mathbf{n}}_{z} = 0$. This expression shows that the splitting error will be large for steep bottom bathymetry ($\hat{n}_z|_{-H} \ll 1$) and if the normal gradient of the non-hydrostatic pressure-correction is large at the free-surface, that is, for surface waves of high amplitude or short wavelengths. However, as argued in §4.2.5, any errors during the free-surface calculation only affect the quality of the pressure predictor, and does not impact the final accuracy or stability of the scheme. As such, the pressure predictor is expected to be poor in regions of steep bathymetry.

Splitting error for steps 2–4 with non-hydrostatic pressure and rigid-lid: For the rigid-lid formulation (§4.2.4), the derivation of the splitting error is similar to that of the free-surface. However, by construction the no normal flow boundary condition will be satisfied at the bottom for the rigid lid, and at the surface the normal only has a component in the z direction. As such, its splitting error reduces to:

$$\mathcal{E}_{\mathrm{RL}} = -a\Delta t \nabla_z \delta p'^{,k+1}|_0,$$

which is zero due to the boundary condition (4.47).

Splitting error for step 4–5 with non-hydrostatic pressure: What remains is to examine the error for the non-hydrostatic pressure split. For this analysis, we can ignore the first splitting step, which is not required. Here we can utilize the results of §3.8.2. In this case, our expected splitting error will be of order $\mathcal{O}(\nu_z \Delta t^2)$ and $\mathcal{O}(\nu_z \Delta t)$ for the velocity and pressure, respectively. The splitting error will manifest itself only at the free-surface and bottom boundary. However, as both vertical gradients and the vertical viscosity are expected to be small for the ocean, we expect this splitting error to be smaller than the leading order error terms from approximations made as part of the mathematical formulation (§4.1.1). As such, the numerical error introduced due to this splitting error will not dominate the error of the simulation.

In summary, the hydrostatic formulation does not contain any time-splitting errors, while the non-hydrostatic formulation's errors are expected to be smaller than

the approximations made when formulating the mathematical model. As such, the proposed time-integration scheme should be sufficient for our purposes. Finally, as discussed in §3.8.3, we note that the results from a full execution of the projection method could also be used as a starting guess (or preconditioner) for the fully coupled problem. Thus, for flows where the splitting errors are large, they could be eliminated with an iterative approach.

4.2.9 Time discretization of tracer equations

For the time discretization of the tracer equations (4.4)–(4.6), we use a semi-implict time discretization scheme, simlar to the discretization of the momentum equations:

$$\frac{T^{k+1}}{a\Delta t} - \nabla_z \cdot \kappa_z \nabla_z T^{k+1} = F_T^{k,k+1}, \tag{4.86}$$

$$\frac{S^{k+1}}{a\Delta t} - \nabla_z \cdot \kappa_z \nabla_z S^{k+1} = F_S^{k,k+1}, \tag{4.87}$$

where, for a first-order time integration scheme, we have $a = 1$ and

$$F_T^{k,k+1} = \frac{T^k}{a\Delta t} - \nabla_{xy} \cdot \mathbf{u}^{k+1} T^k - \nabla_z \cdot w^{k+1} T^k + \nabla_{xy} \cdot \kappa_{xy} \nabla_{xy} T^k + f_T^{k+1},$$

$$F_S^{k,k+1} = \frac{S^k}{a\Delta t} - \nabla_{xy} \cdot \mathbf{u}^{k+1} S^k - \nabla_z \cdot w^{k+1} S^k + \nabla_{xy} \cdot \kappa_{xy} \nabla_{xy} S^k + f_S^{k+1},$$

with boundary conditions

$$T|^{k+1}_{\partial\Omega_{D_T}} = g_{D_T}, \qquad S^{k+1}|_{\partial\Omega_{D_S}} = g_{D_S}, \tag{4.88}$$

$$\left.\frac{\partial T}{\partial \hat{\mathbf{n}}}\right|^{k+1}_{\partial\Omega_{N_T}} = g_{N_T}, \qquad \left.\frac{\partial S}{\partial \hat{\mathbf{n}}}\right|^{k+1}_{\partial\Omega_{N_S}} = g_{N_S}, \tag{4.89}$$

on Dirichlet boundaries $\partial\Omega_D$ and Neumann boundaries $\partial\Omega_N$. Note, while the velocity field at k may be used, we use it at time $k+1$ in the advection term. This does not require coupling since the tracer fields can be advanced after calculating the new velocities, and it marginally improves stability.

4.3 Spatial Discretization of ocean equations using HDG

In this section we give the hybrid discontinuous Galerkin discretization of the time-split equations. We begin by defining the additional notation needed, then give the discretized equations for each step in the time-split procedure.

4.3.1 Finite Element Notation

We will use the same notation as in §2.3.1, along with a few additions. The top surface of the domain $\partial\Omega_n \equiv \Omega^\eta$ is discretized by a finite collection of non-overlapping elements $\partial\mathcal{T}_h^\eta = \cup K^\eta$, where $K^\eta \equiv \partial K \cap \Omega^\eta \neq \emptyset$. That is, for a 3D (or $d = 3$) domain discretized using prisms, the top surface is a 2D (or $d-1 = 2$) domain discretized using triangles. For these 2D elements, we will use the same notation as for the 3D elements, but we usually add a superscript $\bullet^\eta$, or sometimes a subscript $\bullet_\eta$. In particular, we define the 2D finite element spaces for tracers and vectors as

$$\left\{\theta^\eta \in L^2(\Omega^\eta) : \theta^\eta\,|_{K^\eta} \in \mathcal{P}^{\mathrm{p}}(K^\eta), \forall K^\eta \in \mathcal{T}_h^\eta\right\},$$
$$\left\{\boldsymbol{\theta}^\eta \in (L^2(\Omega^\eta))^{d-1} : \boldsymbol{\theta}^\eta\,|_{K^\eta} \in (\mathcal{P}^{\mathrm{p}}(K^\eta))^{d-1}, \forall K^\eta \in \mathcal{T}_h^\eta\right\},$$

respectively. Note that the vectors on this space have one fewer component, $d-1$. To use the HDG method, we will also require the traced finite element spaces existing on the interfaces ε^η of the $d-1$ finite elements:

$$\left\{\theta_\varepsilon^\eta \in L^2(\Omega^\eta) : \theta_\varepsilon^\eta\,|_{e^\eta} \in \mathcal{P}^{\mathrm{p}}(e^\eta), \forall e^\eta \in \varepsilon^\eta\right\},$$
$$\left\{\boldsymbol{\theta}_\varepsilon^\eta \in (L^2(\Omega^\eta))^{d-1} : \boldsymbol{\theta}_\varepsilon^\eta\,|_{e^\eta} \in (\mathcal{P}^{\mathrm{p}}(e^\eta))^{d-1}, \forall e^\eta \in \varepsilon^\eta\right\}.$$

We also set $\{\theta_\varepsilon^\eta = \mathsf{P}\mathbf{g}_D \text{ on } \partial\Omega^\eta\}$, where P is the L^2 projection of the boundary condition $\mathbf{g}_D$ into the same space as θ_ε^η. Note that θ_ε^η is continuous on the interface, e^η, shared by $K^{\eta,+}$ and $K^{\eta,-}$, but discontinuous at the borders between different interfaces.

4.3.2 Discrete equations, remarks and derivations

Here we state and derive our HDG finite element spatial discretization of the time-discrete equations obtained in §4.2.2, and the tracer equations for temperature and salinity (4.86)–(4.87). In what follows, we first obtain in (i) the complete time-discretization of the §4.2.2 scheme, outlining its derivation. In (ii), we then complete the details of the derivations and provide additional discretization remarks. Finally, in (iii), we report the HDG discretization of the tracer equations for temperature and salinity (4.4)–(4.6).

(i) Discrete equations

Our projection method is executed in 5 steps for the non-hydrostatic equations, and 4 steps for the hydrostatic ones. These steps follow the same order as in §4.2.2, and are composed of predictor and corrector steps. The various steps can be combined in four different ways for either a rigid-lid or free-surface model, with either a hydrostatic or non-hydrostatic pressure. The full discretization for a non-hydrostatic free-surface model is summarized in Fig. [4-1], and for a hydrostatic free-surface model in Fig. [4-2]. For both cases, Steps 2 and 3 can be replaced by the rigid-lid as given next also.

Step 1: First velocity predictor. The first step in the time-split equations is to solve for the first velocity predictor, $\bar{\mathbf{u}}^{k+1}$, from the horizontal momentum equations (4.37). The discretizations of the explicit forcing terms in our time-integration schemes are covered in §5, as part of the description of our implementation and of the consistency and algorithmic properties of our schemes. Here we assume that the right-hand-side forcing terms have been appropriately calculated using standard discontinuous Galerkin finite element approaches (see §5). The discretization of (4.37) then leads

to the element-local set of equations

$$\begin{aligned}
&\left(\frac{\bar{\mathbf{q}}_z^{k+1}}{\nu_z}, \boldsymbol{\theta}\right)_K - \left(\nabla_z \bar{\mathbf{u}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \bar{\mathbf{u}}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} = \left\langle \bar{\boldsymbol{\lambda}}_{xy}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K}, \\
&\left(\frac{\bar{\mathbf{u}}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla_z \cdot \bar{\mathbf{q}}_z^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau \bar{\mathbf{u}}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} = \left\langle \tau \bar{\boldsymbol{\lambda}}_{xy}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} \\
&\qquad\qquad - \left(g\mathbf{q}_\eta^k + \nabla_{xy} p'^k, \boldsymbol{\theta}\right)_K + \left(\mathbf{F}_{\mathbf{u}}^{k,k+1}, \boldsymbol{\theta}\right)_K,
\end{aligned} \tag{4.90}$$

where $\tau = 1$ is the stability parameter (see §4.3.3), $\bar{\mathbf{q}}_z^{k+1} = \nu_z \nabla_z \bar{\mathbf{u}}^{k+1}$ is the vertical derivative of the velocity predictor, and $\mathbf{q}_\eta^k = \nabla_{xy} \eta^k$ is the horizontal gradient of the free-surface at time-step k (constant in depth). The global flux-conservation equations for $\bar{\boldsymbol{\lambda}}_{xy}^{k+1}$ are

$$\begin{aligned}
\left\langle \left[\!\left[\bar{\mathbf{q}}_z^{k+1} \cdot \hat{\mathbf{n}}_z - \tau \left(\bar{\mathbf{u}}^{k+1} - \bar{\boldsymbol{\lambda}}_{xy}^{k+1} \right) \right]\!\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \langle g_N, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon + \left\langle \left[\!\left[p'^k \hat{\mathbf{n}}_{xy} \right]\!\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon, \\
\bar{\boldsymbol{\lambda}}_{xy}^{k+1}|_{\partial\Omega_D} &= \mathbf{g}_D,
\end{aligned} \tag{4.91}$$

which is solved $\forall e$ where $\hat{n}_z > \epsilon$, that is, only on non-vertical faces. The derivation of these equations are similar to those in §2.3.2, and are therefore not repeated here. In the case of meshes where the nodes are vertically aligned (sigma-layer models), $\bar{\boldsymbol{\lambda}}_{xy}^{k+1}$ will not be defined on the vertical faces. As such, for these faces we could calculate an edge value using

$$\bar{\boldsymbol{\lambda}}_{xy} = C_1 \{\{\bar{\mathbf{u}}\}\} + C_2 [\![\bar{\mathbf{u}}]\!], \forall e \text{ where } \hat{n}_z \leq \epsilon,$$

but this edge value may not be needed by the discrete algorithm. Where this edge value is needed, the particular values of C_1 and C_2 will depend on the particular operator. For example, an edge value is required to correct the divergence of the predictor velocity, and this edge value should mimic the discretization of the advection operator.

For the hydrostatic solver, the non-hydrostatic pressure $p' = 0$, and the next step is the free-surface or rigid-lid corrector (step 2). For the non-hydrostatic solver, first the vertical velocity momentum equation (4.38) is solved, which is discretized

similarly as:

$$\begin{aligned}
&\left(\frac{\bar{q}_z^{k+1}}{\nu_z},\theta\right)_K - \left(\nabla_z \bar{w}^{k+1},\theta\right)_K + \left\langle \bar{w}^{k+1}, \hat{\mathbf{n}}_z \cdot \theta\right\rangle_{\partial K} = \left\langle \bar{\lambda}_z^{k+1}, \hat{\mathbf{n}}_z \cdot \theta\right\rangle_{\partial K}, \\
&\left(\frac{\bar{w}^{k+1}}{a\Delta t},\theta\right)_K - \left(\nabla_z \cdot \bar{q}_z^{k+1},\theta\right)_K + \left\langle \tau \bar{w}^{k+1},\theta\right\rangle_{\partial K} = \left\langle \tau \bar{\lambda}_z^{k+1},\theta\right\rangle_{\partial K} \\
&\qquad\qquad - \left(\nabla_z p'^k,\theta\right)_K + \left(\mathbf{F}_w^{k,k+1},\theta\right)_K .
\end{aligned} \tag{4.92}$$

The global flux-conservation equation for $\bar{\lambda}_z^{k+1}$ is

$$\begin{aligned}
\left\langle \left[\left[\bar{q}_z^{k+1}\hat{n}_z - \tau\left(\bar{w}^{k+1} - \bar{\lambda}_z^{k+1}\right)\right]\right], \theta_\varepsilon\right\rangle_\varepsilon &= \left\langle g_N, \theta_\varepsilon\right\rangle_\varepsilon + \left\langle \left[\left[p'^k \hat{n}_z\right]\right], \boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon , \\
\bar{\lambda}_w^{k+1}|_{\partial\Omega_D} &= g_D .
\end{aligned} \tag{4.93}$$

We note first that the equations (4.91) and (4.93) are defined for a non-zero $\hat{n}_z$, i.e. for fluxes with a vertical component, which is in accord with the vertical-only implicit diffusion. We also note that in (4.90), the $\mathbf{q}$ variable is used for η while the gradient is used for p'. These properties have implications which are discussed later in the remarks sub-section (ii).

After obtaining the first horizontal velocity predictors, $\bar{\mathbf{u}}^{k+1}$, we have to correct these velocities to obey the depth-integrated continuity equation. This continuity equation can either allow for a free-surface taking the form of (4.41), or have a rigid-lid taking the form of (4.52). Both discretizations are described next.

Step 2 for free-surface model: Free-surface corrector In this step, the free-surface corrector is calculated. It is calculated such that the corrected velocity (from step 3) will satisfy the depth-integrated continuity equation. The discretization of (4.41)

leads to the element-local equations

$$\begin{aligned}\left(\frac{\mathbf{q}_{\delta\eta}^{k+1}}{a\Delta t g(\eta^k+H)},\boldsymbol{\theta}^\eta\right)_{K^\eta}-\left(\nabla_{xy}\delta\eta^{k+1},\boldsymbol{\theta}^\eta\right)_{K^\eta}+\left\langle\delta\eta^{k+1},\hat{\mathbf{n}}_{xy}^\eta\cdot\boldsymbol{\theta}^\eta\right\rangle_{\partial K^\eta}&=\left\langle\lambda_{\delta\eta}^{k+1},\hat{\mathbf{n}}_{xy}^\eta\cdot\boldsymbol{\theta}^\eta\right\rangle_{\partial K^\eta},\\ \left(\frac{\delta\eta^{k+1}}{a\Delta t},\theta^\eta\right)_{K^\eta}-\left(\nabla_{xy}\cdot\mathbf{q}_{\delta\eta}^{k+1},\theta^\eta\right)_{K^\eta}+\left\langle\tau_\eta\delta\eta^{k+1},\theta^\eta\right\rangle_{\partial K^\eta}&=\left\langle\tau_\eta\lambda_{\delta\eta}^{k+1},\theta^\eta\right\rangle_{\partial K^\eta}\\ -\left(\frac{\nabla_{xy}\cdot\bar{\mathbf{U}}^{k+1}}{a\Delta t},\theta^\eta\right)_{K^\eta}+\left\langle\frac{\widehat{\bar{\mathbf{U}}}_\star^{k+1}-\bar{\mathbf{U}}^{k+1}}{a\Delta t},\hat{\mathbf{n}}_{xy}^\eta\theta^\eta\right\rangle_{\partial K^\eta},&\end{aligned}\tag{4.94}$$

where $\tau_\eta=\frac{g(H+\eta^k)}{\tau}$, $\delta\eta^{k+1}=a\Delta t g(\eta^k+H)\nabla_{xy}\delta\eta^{k+1}$ is a function of the horizontal gradient of the free-surface corrector, $\bar{\mathbf{U}}^{k+1}=\int_{-H}^{\eta^k}\bar{\mathbf{u}}^{k+1}\mathrm{d}z$ is the numerically depth-integrated horizontal velocity (see §5.2.1), and $\widehat{\bar{\mathbf{U}}}_\star^{k}=f(\bar{\mathbf{U}}^+,\bar{\mathbf{U}}^-)$ is some function of the solution from the elements bordering a particular edge. We note that the form of this function should be consistent with the discretization of the advection operator (see §5.1.1 or §2.3.2), and the derivation of τ_η is given in §4.3.3.

The global flux-conservation equation used to calculate $\lambda_{\delta\eta}^{k+1}$ is

$$\begin{aligned}\left\langle\left[\left[\mathbf{q}_{\delta\eta}^{k+1}\cdot\hat{\mathbf{n}}_{xy}^\eta-\tau_\eta\left(\delta\eta^{k+1}-\lambda_{\delta\eta}^{k+1}\right)\right]\right],\theta_\varepsilon^\eta\right\rangle_{\varepsilon^\eta}&=\left\langle g_N,\theta_\varepsilon^\eta\right\rangle_{\varepsilon^\eta},\\ \lambda_{\delta\eta}^{k+1}|_{\partial\Omega_D^\eta}&=g_D.\end{aligned}\tag{4.95}$$

The details of the derivation for (4.94)–(4.95) are similar to those in §2.3.2.

Step 3 Free-surface corrections: Second velocity predictor and free-surface updates. The free-surface corrections ((4.44)–(4.45)) are algebraic equations and obtaining their discretizations is straightforward, up to the edge-space relations. Specifically, using the free-surface corrector, the second velocity predictor is obtained from (4.44) on the element and non-vertical HDG edge-space as

$$\bar{\bar{\mathbf{u}}}^{k+1}=\bar{\mathbf{u}}^{k+1}-\frac{\mathbf{q}_{\delta\eta}^{k+1}}{\eta^k+H},\tag{4.96}$$

(which is the discrete form of $\bar{\bar{\mathbf{u}}}^{k+1}=\bar{\mathbf{u}}^{k+1}-a\Delta t g\nabla_{xy}\delta\eta^{k+1}$) and on the vertical HDG

edge-space as

$$\hat{\bar{\mathbf{u}}}_\star^{k+1} = \hat{\bar{\mathbf{u}}}_\star^{k+1} - \frac{\mathbf{q}_{\delta\eta}^{k+1}}{\eta^k + H} + \tau_\eta \left(\frac{\delta\eta^{k+1} - \lambda_{\delta\eta}^{k+1}}{\eta^k + H} \right) \hat{\mathbf{n}}, \tag{4.97}$$

where $\hat{\bar{\mathbf{u}}}_\star^{k+1} = f(\bar{\mathbf{u}}^+, \bar{\mathbf{u}}^-)$, using the same function as for $\hat{\bar{\mathbf{U}}}_\star^{k+1}$ above, consistent with the advection operator.

Additionally the free-surface can be obtained directly from (4.45)

$$\eta^{k+1} = \eta^k + \delta\eta^{k+1}, \tag{4.98}$$

on the element local 2D space, and

$$\lambda_\eta^{k+1} = \lambda_\eta^k + \lambda_{\delta\eta}^{k+1}, \tag{4.99}$$

on the 2D HDG edge-space.

Details and remarks on the above derivations, especially for the edge-space relations, are provided in (ii).

Step 4 for non-hydrostatic model: Pressure corrector. After calculating the second predictor velocities $\bar{\bar{\mathbf{u}}}^{k+1}$ and $\bar{\bar{w}}^{k+1}$, which satisfy the depth-integrated continuity equation (either using the free-surface or rigid-lid formulation), we wish to enforce the full continuity constraint.

The discretization of the non-hydrostatic pressure corrector $\delta p'^{,k+1}$ (4.46) leads to the element local set of equations

$$\begin{aligned}
\left(\mathbf{q}_{\delta p'}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla \delta p'^{,k+1}, \boldsymbol{\theta}\right)_K + \left\langle \delta p'^{,k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K} &= \left\langle \lambda_{\delta p'}{}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K}, \\
- \left(\nabla \cdot \mathbf{q}_{\delta p'}^{k+1}, \theta\right)_K + \left\langle \tau_p \delta p'^{,k+1}, \theta \right\rangle_{\partial K} &= \left\langle \tau_p \lambda_{\delta p'}{}^{k+1}, \theta \right\rangle_{\partial K} - \left(\frac{\nabla \cdot \bar{\bar{\mathbf{v}}}^{k+1}}{a\Delta t}, \theta \right)_K \\
&\quad - \left\langle \frac{(\hat{\bar{\bar{\mathbf{v}}}}_\star^{k+1} - \bar{\bar{\mathbf{v}}}^{k+1}) \cdot \hat{\mathbf{n}}}{a\Delta t}, \theta \right\rangle_{\partial K},
\end{aligned} \tag{4.100}$$

where $\mathbf{q}_{\delta p'}^{k+1} = \nabla \delta p'^{,k+1}$ is the gradient of the pressure corrector, $\tau_p = \frac{1}{a\Delta t \tau}$, $\bar{\bar{\mathbf{v}}}^{k+1} = \left[\bar{\bar{u}}^{k+1}, \bar{\bar{v}}^{k+1}, \bar{\bar{w}}^{k+1}\right]$, and $\hat{\bar{\bar{\mathbf{v}}}}_\star^{k+1} = f(\bar{\bar{\mathbf{v}}}_\star^+, \bar{\bar{\mathbf{v}}}_\star^-)$ should match the discretization of the advection term (see §2.3.2, (2.81)-(2.83)). The global flux-conservation equations used to solve for $\lambda_{\delta p'}^{k+1}$ are then

$$\begin{aligned} \left\langle \left[\left[\mathbf{q}_{\delta p'}^{k+1} \cdot \hat{\mathbf{n}} - \tau_p \left(\delta p'^{,k+1} - \lambda_{\delta p'}^{k+1}\right)\right]\right], \theta_\varepsilon \right\rangle_\varepsilon &= \left\langle g_{N_p}, \theta_\varepsilon \right\rangle_\varepsilon \\ {\lambda_{\delta p'}}^{k+1}|_{\partial\Omega_{D_p}} &= \mathbf{g}_{D_{\delta p}}. \end{aligned} \tag{4.101}$$

Step 5 for non-hydrostatic model: Final velocity and pressure corrections. Once we know the non-hydrostatic pressure corrector, we can correct the second velocity predictor and the non-hydrostatic pressure on the element using the algebraic equations (4.50) and (4.51), whose discretizations are straightforward:

$$\begin{aligned} \mathbf{v}^{k+1} &= \bar{\mathbf{v}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}^{k+1}, \\ p'^{k+1} &= p'^k + \delta p'^{k+1}. \end{aligned}$$

The velocity is also corrected on the HDG edge-space using

$$\hat{\mathbf{v}}_\star^{k+1} = \hat{\bar{\bar{\mathbf{v}}}}_\star^{k+1} - a\Delta t {\mathbf{q}_{\delta p'}}^{k+1} + a\Delta t \tau_p \left(\delta p'^{,k+1} - \lambda_{\delta p'}^{k+1}\right) \hat{\mathbf{n}}. \tag{4.102}$$

Step 2 for rigid-lid model: Rigid-lid corrector The discretization of the rigid-lid corrector equation (4.52) leads to the element-local equations

$$\left(\mathbf{q}_{\delta p_s}^{k+1}, \boldsymbol{\theta}^\eta\right)_{K^\eta} - \left(\nabla_{xy} \delta p_s^{k+1}, \boldsymbol{\theta}^\eta\right)_{K^\eta} + \left\langle \delta p_s^{k+1}, \hat{\mathbf{n}}_{xy}^\eta \cdot \boldsymbol{\theta}^\eta \right\rangle_{\partial K^\eta} = \left\langle \lambda_{\delta p_s}^{k+1}, \hat{\mathbf{n}}_{xy}^\eta \cdot \boldsymbol{\theta}^\eta \right\rangle_{\partial K^\eta}, \tag{4.103}$$

$$\begin{aligned} -a\Delta t H \left(\nabla_{xy} \cdot \mathbf{q}_{\delta p_s}^{k+1}, \boldsymbol{\theta}^\eta\right)_{K^\eta} + a\Delta t H \left\langle \tau_{p_s} {\delta p_s}^{k+1}, \boldsymbol{\theta}^\eta \right\rangle_{\partial K^\eta} - \left(a\Delta t \nabla_{xy} H \cdot \mathbf{q}_{\delta p_s}^{k+1}, \boldsymbol{\theta}^\eta\right)_{K^\eta} \\ = a\Delta t H \left\langle \tau_{p_s} \lambda_\delta^{k+1}, \boldsymbol{\theta}^\eta \right\rangle_{\partial K^\eta} + \left(F_{p_s}^{k+1}, \boldsymbol{\theta}^\eta\right)_{K^\eta}, \end{aligned} \tag{4.104}$$

where $\tau_{p_s} = \frac{1}{a\Delta t \tau}$ (see §4.3.3). Here, the order of the discrete operators are important for numerical consistency. The discretization of the explicit terms, $F_{p_s}^{k+1}$, are detailed in §5.2.1.

These element local equations are subject to the global equation for $\lambda_{\delta p_s}^{k+1}$,

$$\begin{aligned} \left\langle \left[\left[\mathbf{q}_{\delta p_s}^{k+1} \cdot \hat{\mathbf{n}}_{xy}^{\eta} - \tau_{p_s} \left({p_s}^{k+1} - \lambda_{\delta p_s}^{k+1} \right) \right]\right], \theta_{\varepsilon}^{\eta} \right\rangle_{\varepsilon^{\eta}} &= \left\langle g_N, \theta_{\varepsilon}^{\eta} \right\rangle_{\varepsilon^{\eta}}, \\ \lambda_{\delta p_s}^{k+1}|_{\partial\Omega_D^{\eta}} &= g_D. \end{aligned} \tag{4.105}$$

Step 3 for rigid-lid model: Second velocity predictor and rigid-lid surface pres- sure updates.. The rigid-lid corrections (4.56)–(4.57) are algebraic equations and obtaining their discretizations is straightforward, up to the edge-space relations. Specifically, using the rigid-lid corrector, the second velocity predictor is obtained from (4.56) on the element and non-vertical HDG edge-space as

$$\bar{\tilde{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - a\Delta t \mathbf{q}_{\delta p_s}^{k+1}, \tag{4.106}$$

(which is the discretized form of $\bar{\tilde{\mathbf{u}}}^{k+1} = \bar{\mathbf{u}}^{k+1} - a\Delta t \nabla_{xy} \delta p_s^{k+1}$) and on the vertical hybrid discontinuous Galerkin edge-space as

$$\hat{\bar{\tilde{\mathbf{u}}}}_{\star}^{k+1} = \hat{\bar{\mathbf{u}}}_{\star}^{k+1} - a\Delta t (\mathbf{q}_{\delta p_s}^{k+1} + \tau_{p_s} \left(\delta p_s^{k+1} - \lambda_{\delta p_s}^{k+1} \right) \hat{\mathbf{n}}), \tag{4.107}$$

where the 2D field is copied as in Fig. [4-3]. Also the surface pressure at the rigid lid can be recovered from

$$p_s^{k+1} = p_s^k + \delta p_s^{k+1}. \tag{4.108}$$

Details and remarks on the above derivations, especially for the edge-space relations, are provided in (ii).

Step 4 for hydrostatic model: Vertical velocity. For the hydrostatic model, the horizontal velocities satisfying the depth-integrated continuity equation are the final velocities, and require no further modification, $\mathbf{u}^{k+1} = \bar{\tilde{\mathbf{u}}}^{k+1}$. However, the vertical velocity still needs to be calculated such that the 3D continuity constraint is satisfied. This is done by depth-integrating the continuity equation. However, for finite elements, it is simpler to formulate the integration as a derivative equation. Addition-

ally, numerical consistency considerations dictate the form of these equations. The different options are described in §5.2.1, and here we give the form that we use, since it is discretely consistent with the rigid-lid formulation.

We slightly re-write the pressure-correction form of the vertical integration equation (4.72)-(4.75) by letting $W^{k+1} = -a\Delta t \delta p'^{,k+1} = \int_z^{\eta^k} w^{k+1} \mathrm{d}\zeta$. This gives the follow set of equations to be discretized:

$$\begin{aligned} w^{k+1} &= \nabla_z W^{k+1}, \\ -\nabla_z^2 W^{k+1} &= \nabla_{xy} \cdot \bar{\bar{\mathbf{u}}}^{k+1}. \end{aligned}$$

with boundary conditions

$$\begin{aligned} \nabla_z W|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_z = w|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_z = -\bar{\bar{\mathbf{u}}}|_{-H}^{k+1} \cdot \hat{\mathbf{n}}_{xy}, \\ W|_{\partial\Omega_\eta}^{k+1} = 0. \end{aligned}$$

Note, the direction of integration depends on the choice of boundary conditions. For example, for $W^{k+1} = \int_{-H}^{z} w^{k+1} \mathrm{d}\zeta$, we would use a Dirichlet condition on W^{k+1} at the bottom boundary instead.

The HDG discretization of the above equations leads to the element-local equations

$$\begin{aligned} \left(w^{k+1}, \theta\right)_K - \left(\nabla_z W^{k+1}, \theta\right)_K + \left\langle W^{k+1}, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K} = \left\langle \boldsymbol{\lambda}_W^{k+1}, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K} \\ - \left(\nabla_z w^{k+1}, \theta\right)_K + \left\langle \tau_W W^{k+1} \cdot \hat{\mathbf{n}}_z, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K} = \left\langle \tau_W \boldsymbol{\lambda}_W^{k+1} \cdot \hat{\mathbf{n}}_z, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K} \\ + \left(\nabla_{xy} \cdot \mathbf{u}^{k+1}, \theta\right)_K + \left\langle \hat{\mathbf{u}}_\star^{k+1} - \mathbf{u}^{k+1}, \hat{\mathbf{n}}_{xy} \theta \right\rangle_{\partial K}. \end{aligned} \tag{4.109}$$

The globally-couple flux-conservation equation used to solve for $\boldsymbol{\lambda}_W$ is

$$\begin{aligned} \left\langle \left[\!\left[w^{k+1} \hat{n}_z - \tau_W \left(W^{k+1} - \boldsymbol{\lambda}_W^{k+1} \right) \right]\!\right], \theta_\varepsilon \right\rangle_\varepsilon = \left\langle g_N, \theta_\varepsilon \right\rangle_\varepsilon \\ \boldsymbol{\lambda}_W^{k+1}|_{\partial\Omega_\eta} = 0. \end{aligned} \tag{4.110}$$

This solves directly for $w^{k+1} = \nabla_z W^{k+1}$ on the element, and on the non-vertical edges

we have

$$\lambda_z^{k+1} = \{\{w^{k+1}\}\} - \tau_W \left(\{\{W^{k+1}\}\} - \lambda_W^{k+1}\right). \tag{4.111}$$

On the vertical edges, $w \cdot \hat{\mathbf{n}}_z = 0$, and we do not need to calculate an edge value, but for completeness, it could be calculated as:

$$\lambda_z^{k+1} = \{\{w^{k+1}\}\}. \tag{4.112}$$

The major advantage of the hydrostatic approximation is that the above equation is only coupled vertically, that is, in columns of fluid. As such, instead of having to solve a globally coupled 3D pressure equation, the columns of the fluid can be solved independently, or even in parallel.

(ii) Remarks and derivations

Here we provide a number of remarks and derivations concerning the discrete HDG equations. First we explain why the explicit gradients of the free-surface and pressure are discretized differently. Second we note how to properly implement the rigid-lid corrector equations. Third we note how to implement the 2D corrections of the 3D fields. Finally, we explain how to derive the edge-space corrections for the velocity.

First, in step 1, for (4.91), we note that the $\mathbf{q}$ variable (which explicitly includes an integration over an edge) is used for η while the gradient is used for p' (which does not explicitly include an integration over an edge). Because η has a time-derivative in its equation, it should to be treated differently from p'. To solve the original system of equations (4.31)–(4.34), we need to know the initial free-surface height, and the "instantaneous" average pressure (such that the final velocity is divergence free). In the time-integration, then, we have to calculate the change in free-surface height $\delta\eta^{k+1}$ and the pressure that gives a divergence-free velocity. As such, we can use the known edge-flux for the free-surface λ_η to calculate the numerical gradient of the free-surface

1. First velocity predictor(momentum equations)

Element-Local equations:

$$\left(\frac{\bar{\mathbf{q}}_z^{k+1}}{\nu_z},\boldsymbol{\theta}\right)_K - \left(\nabla_z\bar{\mathbf{u}}^{k+1},\boldsymbol{\theta}\right)_K + \left\langle\bar{\mathbf{u}}^{k+1},\mathbf{n}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\bar{\boldsymbol{\lambda}}_{xy}^{k+1},\hat{\mathbf{n}}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K}$$

$$\left(\frac{\bar{\mathbf{u}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K - \left(\nabla_z\cdot\bar{\mathbf{q}}_z^{k+1},\boldsymbol{\theta}\right)_K + \left\langle\tau\bar{\mathbf{u}}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\tau\bar{\boldsymbol{\lambda}}_{xy}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} - \left(g\mathbf{q}_\eta^k + \nabla_{xy}p'^k,\boldsymbol{\theta}\right)_K + \left(\mathbf{F}_{\mathbf{u}}^{k,k+1},\boldsymbol{\theta}\right)_K$$

$$\left(\frac{\bar{q}_z^{k+1}}{\nu_z},\theta\right)_K - \left(\nabla_z\bar{w}^{k+1},\theta\right)_K + \left\langle\bar{w}^{k+1},\hat{\mathbf{n}}_z\cdot\theta\right\rangle_{\partial K} = \left\langle\bar{\lambda}_z^{k+1},\hat{\mathbf{n}}_z\cdot\theta\right\rangle_{\partial K}$$

$$\left(\frac{\bar{w}^{k+1}}{a\Delta t},\theta\right)_K - \left(\nabla_z\cdot\bar{q}_z^{k+1},\theta\right)_K + \left\langle\tau\bar{w}^{k+1},\theta\right\rangle_{\partial K} = \left\langle\tau\bar{\lambda}_z^{k+1},\theta\right\rangle_{\partial K} - \left(\nabla_z p'^k,\theta\right)_K + \left(\mathbf{F}_w^{k,k+1},\theta\right)_K$$

Edge-space global flux conservation equations:

$$\left\langle\left[\left[\bar{\mathbf{q}}_z^{k+1}\cdot\hat{\mathbf{n}}_z - \tau\left(\bar{\mathbf{u}}^{k+1} - \bar{\boldsymbol{\lambda}}_{xy}^{k+1}\right)\right]\right],\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon = \left\langle g_N,\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon + \left\langle\left[\left[p'^k\hat{\mathbf{n}}_{xy}\right]\right],\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon$$

$$\bar{\boldsymbol{\lambda}}_{xy}^{k+1}|_{\partial\Omega_D} = \mathbf{g}_D$$

$$\left\langle\left[\left[\bar{q}_z^{k+1}\hat{n}_z - \tau\left(\bar{w}^{k+1} - \bar{\lambda}_z^{k+1}\right)\right]\right],\theta_\varepsilon\right\rangle_\varepsilon = \left\langle g_N,\theta_\varepsilon\right\rangle_\varepsilon + \left\langle\left[\left[p'^k\hat{n}_z\right]\right],\boldsymbol{\theta}_\varepsilon\right\rangle_\varepsilon$$

$$\bar{\lambda}_w^{k+1}|_{\partial\Omega_D} = g_D$$

2. Free-surface corrector (to enforce 2D continuity)

Element-Local equations:

$$\left(\frac{\mathbf{q}_{\delta\eta}^{k+1}}{a\Delta t g(\eta^k+H)},\boldsymbol{\theta}^\eta\right)_{K^\eta} - \left(\nabla_{xy}\delta\eta^{k+1},\boldsymbol{\theta}^\eta\right)_{K^\eta} + \left\langle\delta\eta^{k+1},\mathbf{n}_{xy}^\eta\cdot\boldsymbol{\theta}^\eta\right\rangle_{\partial K^\eta} = \left\langle\lambda_{\delta\eta}^{k+1},\hat{\mathbf{n}}_{xy}^\eta\cdot\boldsymbol{\theta}^\eta\right\rangle_{\partial K^\eta}$$

$$\left(\frac{\delta\eta^{k+1}}{a\Delta t},\theta^\eta\right)_{K^\eta} - \left(\nabla_{xy}\cdot\mathbf{q}_{\delta\eta}^{k+1},\theta^\eta\right)_{K^\eta} + \left\langle\tau_\eta\delta\eta^{k+1},\theta^\eta\right\rangle_{\partial K^\eta} = \left\langle\tau_\eta\lambda_{\delta\eta}^{k+1},\theta^\eta\right\rangle_{\partial K^\eta} - \left(\frac{\nabla_{xy}\cdot\bar{\mathbf{U}}^{k+1}}{a\Delta t},\theta^\eta\right)_{K^\eta} + \left\langle\frac{\hat{\mathbf{U}}_\star^{k+1} - \bar{\mathbf{U}}^{k+1}}{a\Delta t},\hat{\mathbf{n}}_{xy}^\eta\theta^\eta\right\rangle_{\partial K^\eta}$$

Edge-space global flux conservation equations:

$$\left\langle\left[\left[\mathbf{q}_{\delta\eta}^{k+1}\cdot\hat{\mathbf{n}}_{xy}^\eta - \tau_\eta\left(\delta\eta^{k+1} - \lambda_{\delta\eta}^{k+1}\right)\right]\right],\theta_\varepsilon^\eta\right\rangle_{\varepsilon^\eta} = \left\langle g_N,\theta_\varepsilon^\eta\right\rangle_{\varepsilon^\eta}$$

$$\lambda_{\delta\eta}^{k+1}|_{\partial\Omega_D^\eta} = g_D$$

Consistent HDG stability parameter:

$$\tau_\eta = \frac{g(H+\eta^k)}{\tau}$$

3. First velocity and free-surface corrections

Element-Local corrections:

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

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

Edge-space corrections:

$$\hat{\bar{\mathbf{u}}}_\star^{k+1} = \hat{\bar{\mathbf{u}}}_\star^{k+1} - \frac{\mathbf{q}_{\delta\eta}^{k+1}}{\eta^k+H} + \tau_\eta\left(\frac{\delta\eta^{k+1} - \lambda_{\delta\eta}^{k+1}}{\eta^k+H}\right)\hat{\mathbf{n}}$$

$$\lambda_\eta^{k+1} = \lambda_\eta^k + \lambda_{\delta\eta}^{k+1}$$

4. Pressure corrector (to enforce 3D continuity)

Element-Local equations:

$$\left(\mathbf{q}_{\delta p'}^{k+1},\boldsymbol{\theta}\right)_K - \left(\nabla\delta p'^{,k+1},\boldsymbol{\theta}\right)_K + \left\langle\delta p'^{,k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle\lambda_{\delta p'}{}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\theta}\right\rangle_{\partial K}$$

$$-\left(\nabla\cdot\mathbf{q}_{\delta p'}^{k+1},\theta\right)_K + \left\langle\tau_p\delta p'^{,k+1},\theta\right\rangle_{\partial K} = \left\langle\tau_p\lambda_{\delta p'}{}^{k+1},\theta\right\rangle_{\partial K} - \left(\frac{\nabla\cdot\bar{\mathbf{v}}^{k+1}}{a\Delta t},\theta\right)_K - \left\langle\frac{(\hat{\bar{\mathbf{v}}}_\star^{k+1} - \bar{\mathbf{v}}^{k+1})\cdot\hat{\mathbf{n}}}{a\Delta t},\theta\right\rangle_{\partial K}$$

Edge-space global flux conservation equations:

$$\left\langle\left[\left[\mathbf{q}_{\delta p'}^{k+1}\cdot\hat{\mathbf{n}} - \tau_p\left(\delta p'^{,k+1} - \lambda_{\delta p'}^{k+1}\right)\right]\right],\theta_\varepsilon\right\rangle_\varepsilon = \left\langle g_{N_p},\theta_\varepsilon\right\rangle_\varepsilon$$

$$\lambda_{\delta p'}{}^{k+1}|_{\partial\Omega_{D_p}} = \mathbf{g}_{D_{\delta p}}$$

Consistent HDG stability parameter:

$$\tau_p = \frac{1}{\tau a\Delta t}$$

5. Final velocity and pressure corrections

Element-Local equations:

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t\mathbf{q}_{\delta p}^{k+1}$$

$$p'^{k+1} = p'^k + \delta p'^{k+1}$$

Edge-space corrections:

$$\hat{\mathbf{v}}_\star^{k+1} = \hat{\bar{\mathbf{v}}}_\star^{k+1} - a\Delta t\mathbf{q}_{\delta p'}{}^{k+1} + a\Delta t\tau_p\left(\delta p'^{,k+1} - \lambda_{\delta p'}^{k+1}\right)\hat{\mathbf{n}}$$

Figure 4-1: New HDG projection method scheme for a non-hydrostatic free-surface ocean model.

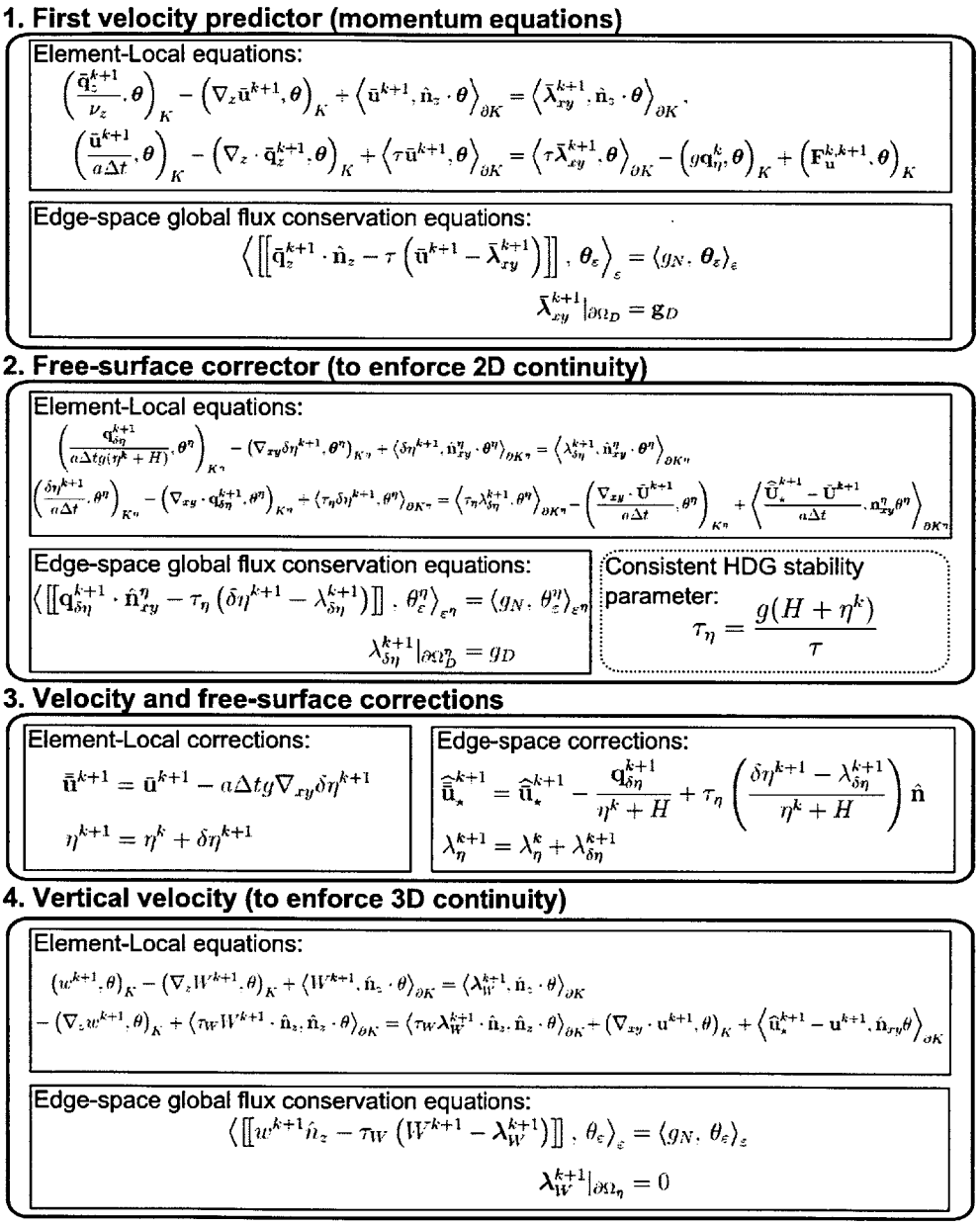


Figure 4-2: New HDG projection method scheme for a hydrostatic free-surface ocean model.

at time k ($\mathbf{q}_\eta^k$), but the edge-flux for the pressure needs to calculated for a consistent flux, as was done for the scheme in §2.3.2. We note that it is not necessary to use this approach, since a scheme similar to the one presented in Appendix B could be used. However, with the present approach, we can find consistent values for the HDG stability parameters that do not depend on the type of time-integration scheme used (see §4.3.3). Hence, we have slightly different discretizations for the gradients of the explicit free-surface and non-hydrostatic pressure contributions.

Second, in step 2 of the rigid-lid calculation, we note that the H term appears outside of the element-local integrals. This is because the finite element discretization of each term will be integrated vertically to preserve numerical consistency between the 2D and 3D continuity equations. We note that, depending on the implementation, a mass-matrix inverse may pre-multiply the operators. In this case, the H term is still in front of all the operators (as opposed to sandwiched between the mass-matrix inverse and the finite element operators). The numerical stability of rigid-lid hydrostatic ocean models depends on this numerical consistency. As such, any successful implementation will pay careful attention to the order of operations (see §5.2.1).

Third, for step 3 of the free-surface and rigid-lid updates, the 2D correction needs to be applied to a 3D field. As such, the 2D correction field is copied down to the vertically aligned nodes in the 3D field. Note that only the vertical 3D HDG edge-space (corresponding to the 2D free-surface edge-space) requires the HDG edge-space update (Fig. [4-3]).

Last, for step 3 of the free-surface and rigid-lid updates, the form of the HDG edge-space updates for the velocity ((4.97) and (4.107)) are consistent with the 2D continuity equation. The consistency can be proven by substituting these fluxes into the discrete 2D continuity equations ((4.94) and (4.104)) as was done in §2.3.3. As the proof is similar, we do not repeat it here.

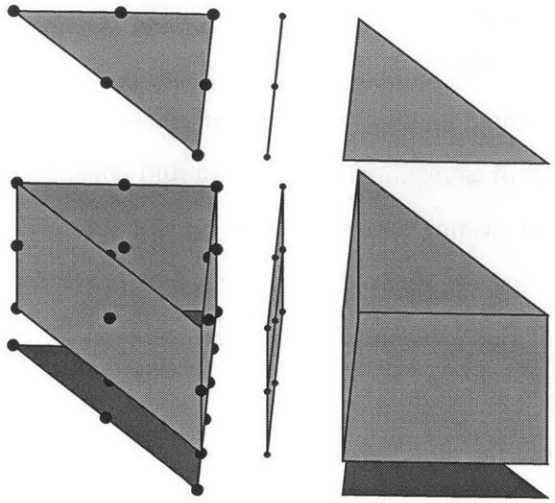


Figure 4-3: The 2D free-surface gradient correction on the 2D elements are copied down to the 3D element nodes, and non-vertical hybrid discontinuous Galerkin edge nodes (red dash-dot). The correction on the 2D hybrid discontinuous Galerkin edge-space is copied down to the 3D vertical hybrid discontinuous Galerkin edge nodes (blue dashes).

(iii) Discrete tracer equations

The temperature and salinty tracer equations discretized using HDG is described next. The derivation of these equations is similar to those in §2.3.2, so we do not repeat it here but only report the result.

The HDG discretization of the temperature and salinity equations (4.86)–(4.87) are:

$$\begin{aligned}
&\left(\frac{\mathbf{q}_{z,T}^{k+1}}{\kappa_z},\boldsymbol{\theta}\right)_K - \left(\nabla_z T^{k+1},\boldsymbol{\theta}\right)_K + \left\langle T^{k+1},\hat{\mathbf{n}}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle \lambda_T^{k+1},\hat{\mathbf{n}}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K},\\
&\left(\frac{T^{k+1}}{a\Delta t},\theta\right)_K - \left(\nabla_z\cdot\mathbf{q}_{z,T}^{k+1},\theta\right)_K + \left\langle \tau T^{k+1},\theta\right\rangle_{\partial K} = \left\langle \tau\lambda_T^{k+1},\theta\right\rangle_{\partial K} + \left(\mathbf{F}_T^{k,k+1},\theta\right)_K,
\end{aligned} \tag{4.113}$$

$$\begin{aligned}
&\left(\frac{\mathbf{q}_{z,S}^{k+1}}{\kappa_z},\boldsymbol{\theta}\right)_K - \left(\nabla_z S^{k+1},\boldsymbol{\theta}\right)_K + \left\langle S^{k+1},\hat{\mathbf{n}}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K} = \left\langle \lambda_S^{k+1},\hat{\mathbf{n}}_z\cdot\boldsymbol{\theta}\right\rangle_{\partial K},\\
&\left(\frac{S^{k+1}}{a\Delta t},\theta\right)_K - \left(\nabla_z\cdot\mathbf{q}_{z,S}^{k+1},\theta\right)_K + \left\langle \tau S^{k+1},\theta\right\rangle_{\partial K} = \left\langle \tau\lambda_S^{k+1},\theta\right\rangle_{\partial K} + \left(\mathbf{F}_S^{k,k+1},\theta\right)_K,
\end{aligned} \tag{4.114}$$

where $\tau = 1$, $\mathbf{q}_{z,T}^{k+1} = \nabla_z T^{k+1}$, $\mathbf{q}_{z,S}^{k+1} = \nabla_z S^{k+1}$, λ_T^{k+1} is the temperature on the HDG edge space, λ_S^{k+1} is the salinity on the HDG edge space, and $F_T^{k,k+1}$, $F_S^{k,k+1}$ are the combination of explicit, known, and forcing functions for temperature and salinity, respectively. The discretization of the explicit terms are described in §5.1.1 for the advection, and §5.2.3 for the diffusion.

4.3.3 Derivation of consistent HDG stability parameters

We have yet to justify the choices of the stability parameters τ, τ_p, and τ_η. As stated in §2.3.4, the magnitude of the stability parameter for the velocity τ has been well-studied, and suggests that $\tau = 1$ is a good choice. What remains is to derive the consistent values of τ_p and τ_η. We utilize the free-surface, non-hydrostatic pressure model for the proof, since the derivation is similar for all other cases.

Following the same procedure as in §2.3.4, we can construct the un-split velocity on the HDG edge-space in terms of the element-local values. For the non-hydrostatic velocity, the edge-term has nearly the same form as (2.84), but with the addition of the 2D free-surface (or surface pressure) component. Then we can compare this flux of the un-split equation to the final edge-space flux of the split equation.

We begin by obtaining the element-local and globally-coupled HDG discretization for the un-split equations. Then, we solve for the un-split $\boldsymbol{\lambda}$ in terms of element-local quantities. To compare the un-split fluxes to the split fluxes, we also have to express the edge-space variables $\bar{\boldsymbol{\lambda}}$, $\lambda_{\delta p}$, and $\lambda_{\delta\eta}$ in terms of element-local quantities, then from the final edge-space velocity using (2.73).

The un-split equations discretized using the HDG method are as follows. The

element-local set of equations with HDG fluxes substituted are:

$$\left(\frac{\mathbf{q}_z^{k+1}}{\nu_z}, \boldsymbol{\theta}\right)_K - \left(\nabla_z \mathbf{u}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \mathbf{u}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} = \left\langle \boldsymbol{\lambda}_{xy}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K},$$

$$\left(\frac{\mathbf{u}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K + \left(-\nabla_z \cdot \mathbf{q}_z^{k+1} + g\nabla\delta\eta^{k+1} + \nabla p'^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau \mathbf{u}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} + = \left\langle \tau \boldsymbol{\lambda}_{xy}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K}$$
$$- \left(g\mathbf{q}_\eta^k, \boldsymbol{\theta}\right)_K + \left(\mathbf{F}_{\mathbf{u}}^{k,k+1}, \boldsymbol{\theta}\right)_K, \tag{4.115}$$

$$\left(\frac{q_z^{k+1}}{\nu_z}, \theta\right)_K - \left(\nabla_z w^{k+1}, \theta\right)_K + \left\langle w^{k+1}, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K} = \left\langle \lambda_z^{k+1}, \hat{\mathbf{n}}_z \cdot \theta \right\rangle_{\partial K},$$

$$\left(\frac{w^{k+1}}{a\Delta t}, \theta\right)_K - \left(\nabla_z \cdot q_z^{k+1}, \theta\right)_K + \left\langle \tau w^{k+1}, \theta \right\rangle_{\partial K} + \left(\nabla_z p'^{k+1}, \theta\right)_K = \left\langle \tau \lambda_z^{k+1}, \theta \right\rangle_{\partial K} + \left(\mathbf{F}_w^{k,k+1}, \theta\right)_K, \tag{4.116}$$

$$\left(\frac{\delta\eta^{k+1}}{a\Delta t}, \theta^\eta\right)_{K^\eta} + \left(\frac{\nabla_{xy} \cdot \mathbf{U}^{k+1}}{a\Delta t}, \theta^\eta\right)_{K^\eta} + \left\langle \frac{\mathbf{U}^{k+1}}{a\Delta t}, \hat{\mathbf{n}}_{xy}^\eta \theta^\eta \right\rangle_{\partial K^\eta} = \left\langle \frac{\widehat{\mathbf{U}}_\star^{k+1}}{a\Delta t}, \hat{\mathbf{n}}_{xy}^\eta \theta^\eta \right\rangle_{\partial K^\eta}, \tag{4.117}$$

$$\left(\nabla \cdot \mathbf{v}^{k+1}, \theta\right)_K - \left\langle \mathbf{v}^{k+1} \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K} = -\left\langle \boldsymbol{\lambda}^{k+1} \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K}, \tag{4.118}$$

$$\left(p'^{k+1}, \frac{1}{|K|}\right)_K = |p'|^{k+1}, \tag{4.119}$$

where $\mathbf{U}^{k+1} = \int_{-H}^{\eta^k} \mathbf{u}^{k+1} \mathrm{d}z$ is the numerically depth-integrated horizontal velocity (see §5.2.1), and $\widehat{\mathbf{U}}_\star = f(\bar{\mathbf{U}}^+, \bar{\mathbf{U}}^-)$ is some function of the solution from the elements bordering a particular edge. Note that this system of coupled element-local equations is solvable once the velocity boundary conditions, average pressure, initial free-surface gradient, and forcing terms are specified (note the forcing terms also contain velocity initial conditions). The globally coupled equations for $\boldsymbol{\lambda}_{xy}^{k+1}$, λ_z^{k+1}, and $|p'|^{k+1}$ are

$$\left\langle \left[\left[\mathbf{q}_z^{k+1} \cdot \hat{\mathbf{n}}_z - p'^{k+1} \hat{\mathbf{n}}_{xy} - g\delta\eta^{k+1} \hat{\mathbf{n}}_{xy} - \tau \left(\mathbf{u}^{k+1} - \boldsymbol{\lambda}_{xy}^{k+1} \right) \right]\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon = \langle g_N, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon ,$$
$$\boldsymbol{\lambda}_{xy}^{k+1}|_{\partial\Omega_D} = \mathbf{g}_D, \tag{4.120}$$

$$\left\langle \left[\left[q_z^{k+1} \hat{n}_z - p'^{k+1} \hat{n}_z - \tau \left(w^{k+1} - \lambda_z^{k+1} \right) \right]\right], \theta_\varepsilon \right\rangle_\varepsilon = \langle g_N, \theta_\varepsilon \rangle_\varepsilon ,$$
$$\lambda_w^{k+1}|_{\partial\Omega_D} = g_D. \tag{4.121}$$

$$\left\langle \boldsymbol{\lambda}^{k+1} \cdot \hat{\mathbf{n}}, \frac{1}{|\partial K|} \right\rangle_{\partial\mathcal{T}_h} = 0, \tag{4.122}$$
$$\boldsymbol{\lambda}^{k+1}|_{\varepsilon_D^\partial} = \mathbf{g}_D.$$

To solve for $\boldsymbol{\lambda}_{xy}^{k+1}$ and λ_z^{k+1} in the interior of the domain, we proceed as in §2.3.4. Using (4.120) and (4.121) we expand the "jump" operator in terms of element-local quantities on either side of the edge, and finally we recombine terms using the "jump" and "mean" operators. The result gives the following fluxes:

$$\boldsymbol{\lambda}_{xy}^{k+1} = \{\{\mathbf{u}^{k+1}\}\} - \frac{1}{2\tau} [\![\mathbf{q}_z^{k+1} \hat{n}_z]\!] + \frac{g}{2\tau} [\![\delta\eta^{k+1} \hat{\mathbf{n}}_{xy}]\!] + \frac{1}{2\tau} \left[\!\left[p'^{k+1} \hat{\mathbf{n}}_{xy} \right]\!\right], \tag{4.123}$$

$$\lambda_z^{k+1} = \{\{w^{k+1}\}\} - \frac{1}{2\tau} [\![q_z^{k+1} \hat{n}_z]\!] + \frac{1}{2\tau} \left[\!\left[p'^{k+1} \hat{n}_z \right]\!\right] \tag{4.124}$$

Note, the hydrostatic pressure contribution does not show up here, since it was treated explicitly.

Now we want to compare this to the $\boldsymbol{\lambda}_{xy}$ and λ_z obtained from the split equations. Again proceeding as in §2.3.4, and starting with $\boldsymbol{\lambda}_{xy}$, we construct the horizontal velocity from the split equation. This was done by first constructing the flux for $\bar{\boldsymbol{\lambda}}_{xy}^{k+1}$ (using (4.91)) in terms of element-local quantities, then applying the edge-space corrections for the free-surface (4.97) and non-hydrostatic pressure (4.102).

The derivation is similar to that in §2.3.4, and thus we only give the resulting fluxes

$$\boldsymbol{\lambda}_{xy}^{k+1} = \{\{\mathbf{u}^{k+1}\}\} - \frac{1}{2\tau}\left[\!\left[\bar{\mathbf{q}}_z{}^{k+1}\hat{n}_z\right]\!\right] + \frac{\tau_\eta}{2(\eta^k + H)}\left[\!\left[\delta\eta^{k+1}\hat{\mathbf{n}}_{xy}\right]\!\right] + \frac{1}{2\tau}\left[\!\left[p'^k\hat{\mathbf{n}}_{xy}\right]\!\right] + a\Delta t\frac{\tau_p}{2}\left[\!\left[\delta p'^{k+1}\hat{\mathbf{n}}_{xy}\right]\!\right], \tag{4.125}$$

$$\lambda_z^{k+1} = \{\{w^{k+1}\}\} - \frac{1}{2\tau}\left[\!\left[\bar{q}_z{}^{k+1}\hat{n}_z\right]\!\right] + \frac{1}{2\tau}\left[\!\left[p'^k\hat{n}_z\right]\!\right] + a\Delta t\frac{\tau_p}{2}\left[\!\left[\delta p'^{k+1}\hat{n}_z\right]\!\right]. \tag{4.126}$$

Comparing the horizontal edge-velocities for the un-split and split equations, we find

$$\tau_\eta = \frac{g(H+\eta^k)}{\tau}, \tag{4.127}$$

$$\tau_p \approx \frac{1}{\tau a\Delta t}, \tag{4.128}$$

which is similar to the result in §2.3.4. Note, the approximation sign in (4.128) comes from assuming that $\left[\!\left[(\mathbf{q}_z{}^{k+1} - \bar{\mathbf{q}}_z{}^{k+1})\hat{n}_z\right]\!\right] \approx 0$ (as formally justified in §2.3.4). Also, if we compare the λ_z's from the vertical equation, we obtain the same result for τ_p. Hence, we don't provide this derivation here. Thus, for a consistent scheme, the stability parameters τ_p and τ_η should be specified using (4.127) and (4.128), respectively.

4.4 Verification and validation benchmarks

In this section we show that the free-surface and vertical integration terms are correctly formulated and implemented. Verification of a new code is a necessary to ensure that it solves the intended equations (Oreskes et al., 1994, Roache, 1998). To evaluate our extension of the schemes from §2.3.2 to §4.3.2, we use an analytical tidal flow in a channel benchmark followed by a 3D version of the lock-exchange problem.

4.4.1 Tidal flow benchmark

Setup: We use the the rectangular channel benchmark from Chen et al. (2007). The geometrical setup is shown in Fig. [4-4], where the depth of the channel is $H(x) = \frac{xH_0}{L}$, the length and width of the computational domain is $L - L_1$ and B,

respectively. The boundary conditions are as indicated: the coastline is a rigid, no-slip wall; the channel walls and bottom are free-slip boundaries; the top is a free-surface; and the open boundary is Dirichlet for velocity and Neumann for the free-surface. For this benchmark, we consider a reduced set of equations, that is:

$$\frac{\partial u}{\partial t} + g\frac{\partial \eta}{\partial x} = 0$$
$$\frac{\partial \eta}{\partial t} + \frac{\partial uH}{\partial x} = 0,$$

with boundary conditions

$$u = 0 \text{ on } \partial\Omega|_{x=L_1},$$
$$u = u_0(L)\mathrm{e}^{-\sigma t} \text{ on } \partial\Omega|_{x=L},$$
$$\frac{\partial \eta}{\partial x} = 0 \text{ on } \partial\Omega|_{x=L_1, x=L},$$

where u_0 is defined below. In our model, we turn off the nonlinear advection term, the Coriolis forcing and the density forcing. We set the turbulent eddy viscosity to a small value $\nu = \epsilon$, such that it will be negligible. The solution to the above equations are given in Chen et al. (2007) as:

$$\eta(x,t) = \eta_0(x)\mathrm{e}^{-\sigma t}, \tag{4.129}$$
$$u(x,t) = u_0(x)\mathrm{e}^{-\sigma t}, \tag{4.130}$$

where

$$\eta_0(x) = \frac{A}{F(L,L_1,k)}\left[Y_0'\left(2k\sqrt{L_1}\right)J_0\left(2k\sqrt{x}\right) - J_0'\left(2k\sqrt{L_1}\right)Y_0\left(2k\sqrt{x}\right)\right],$$
$$F(L,L_1,k) = Y_0'\left(2k\sqrt{L_1}\right)J_0\left(2k\sqrt{L}\right) - J_0'\left(2k\sqrt{L_1}\right)Y_0\left(2k\sqrt{L}\right),$$
$$k = \sqrt{\frac{\sigma^2 L}{gH_0}},$$

Parameter	Non-resonant	Near-resonant
L	580 km	300 km
L_1	290 km	19 km
H_0	10 m	0.67 m

Table 4.1: Parameter values for the tidal flow benchmark

Y_0 and J_0 are the zeroth-order Bessel functions of the first and second kinds, and with some manipulation we can find

$$u_0(x) = \frac{igkA}{\sigma F(L, L_1, k)} \left[Y_0' \left(2k\sqrt{L_1} \right) J_0' \left(2k\sqrt{x} \right) - J_0' \left(2k\sqrt{L_1} \right) Y_0' \left(2k\sqrt{x} \right) \right].$$

The problem is forced using the M_2 tidal frequency, $\sigma = \frac{2\pi}{12.42 \times 3600s}$ with an amplitude $A = 0.01$[m], and the parameter values for the non-resonant and near-resonant cases are given in Table [4.1], while the channel width $B = 5$[km] does not affect the solution, but allows us to test our 3D code. The analytical solutions are used to initialize the fields.

For these 2D simulations we use p = 2 basis functions, with 100 elements in the x-direction, and 1 element in the y- and z-directions. For the time-step, we use 80 time-steps per tidal cycle, or $\Delta t = \frac{2\pi}{\sigma 80} = 558.9s$. We perform the simulations using first, second, and third-order-accurate time integrators.

Results: The relative errors are calculated by dividing the absolute errors by the maximum amplitude of the analytical solution. We observe excellent agreement for this benchmark with the second and third-order time integration schemes for both cases (Fig. [4-5]). The first-order time integration scheme has significant errors, up to 60%, while the second-order scheme has errors up to 3.1% for the first case and 0.5% for the second case. The third-order time integration scheme performs best with errors up to 2.1% for the first case and 0.31% for the second case.

These results demonstrate that our free-surface time-stepping algorithm is correctly formulated and implemented, since the higher-order schemes have lower error levels than the first-order accurate scheme.

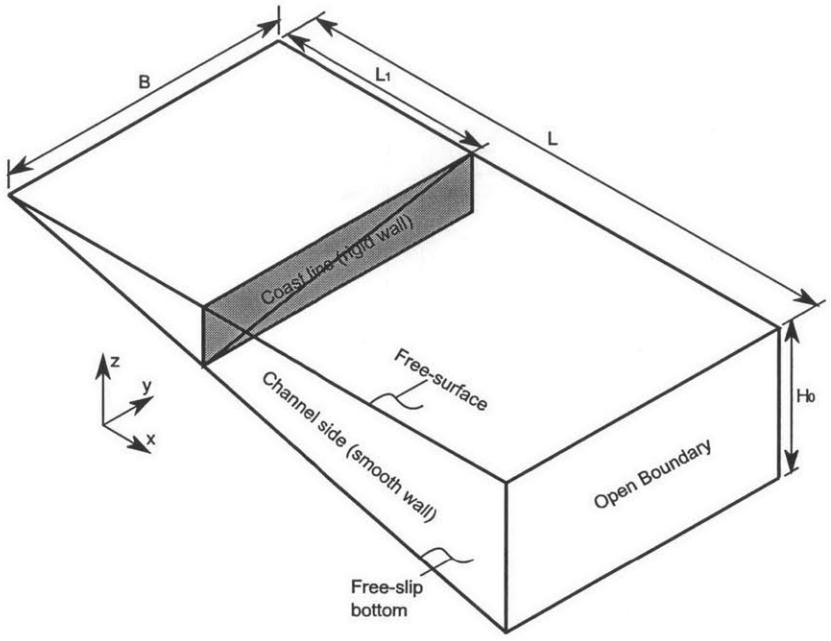


Figure 4-4: Sketch of domain for tidal flow benchmark.

4.4.2 Lock exchange benchmark

Setup: The setup of this benchmark is the same as that reported in §3.8.4. However, here we use a 3D domain, with a domain length (L), height (H), and width (B) of $L \times H \times B = 8 \times 2 \times 10^{-4}$, where the width is discretized using 1 element, and free-slip boundary conditions are used on the side-walls of the channel.

We perform this simulation for both the non-hydrostatic and hydrostatic versions of the code to compare our results to that of Fringer et al. (2006). The simulation uses a rigid-lid with the hydrostatic pressure contribution calculated as described in §5.2.2 (also see §4.1.2). Therefore, the numerical nature of the Boussinesq forcing is much different from the 2D simulations in §3.8.4. In particular, since we are now integrating the density variation in the vertical, an asymmetry is introduced due to the direction of integration (from the free-surface down to the bottom).

Results: The results at $T = [5, 10]$ are shown in Fig. [4-6], and the Froude numbers from various sources are reported in Table [4.2]. First, we find that our non-hydrostatic simulation approximately reproduces the results of Härtel et al. (2000) and Fringer et al. (2006). Nonetheless, when compared to Fringer et al. (2006), our

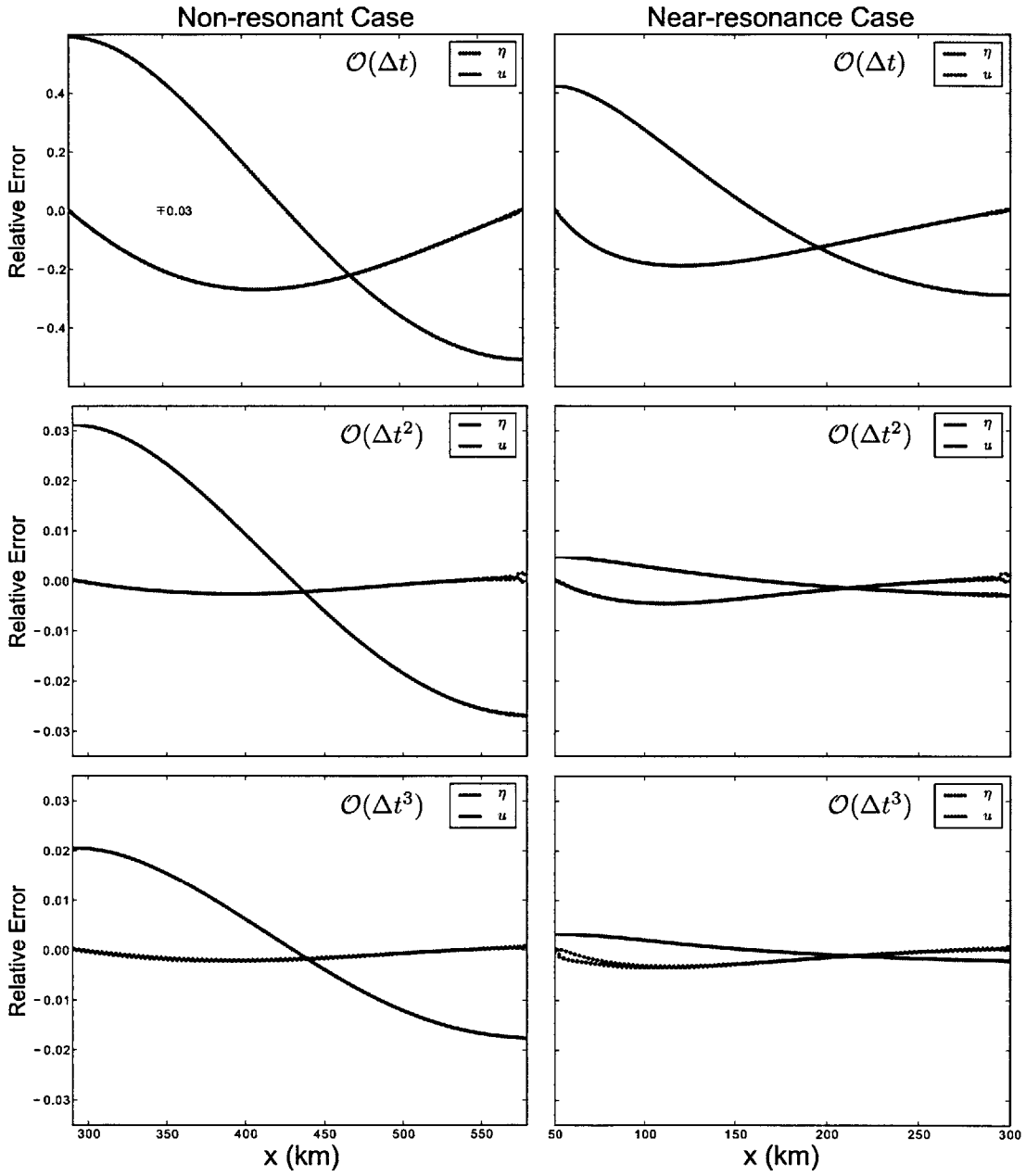


Figure 4-5: Relative errors (normalized by the maximum absolute value of the analytical solution) for the tidal flow benchmark at tidal cycle 10.125, $T = 10.125$ (12.42×3600). The errors for the non-resonant case (left) and near-resonant case (right) are plotted for the first (top), second (middle), and third (bottom) order-accurate time-integration schemes.

Source	Hydrostatic	Non-hydrostatic
Härtel et al. (2000)	–	0.574
Fringer et al. (2006)	0.470	0.562
HDG	0.544	0.576

Table 4.2: Froude numbers $Fr = \frac{u_f}{u_b}$ from various sources for no-slip lock exchange benchmark.

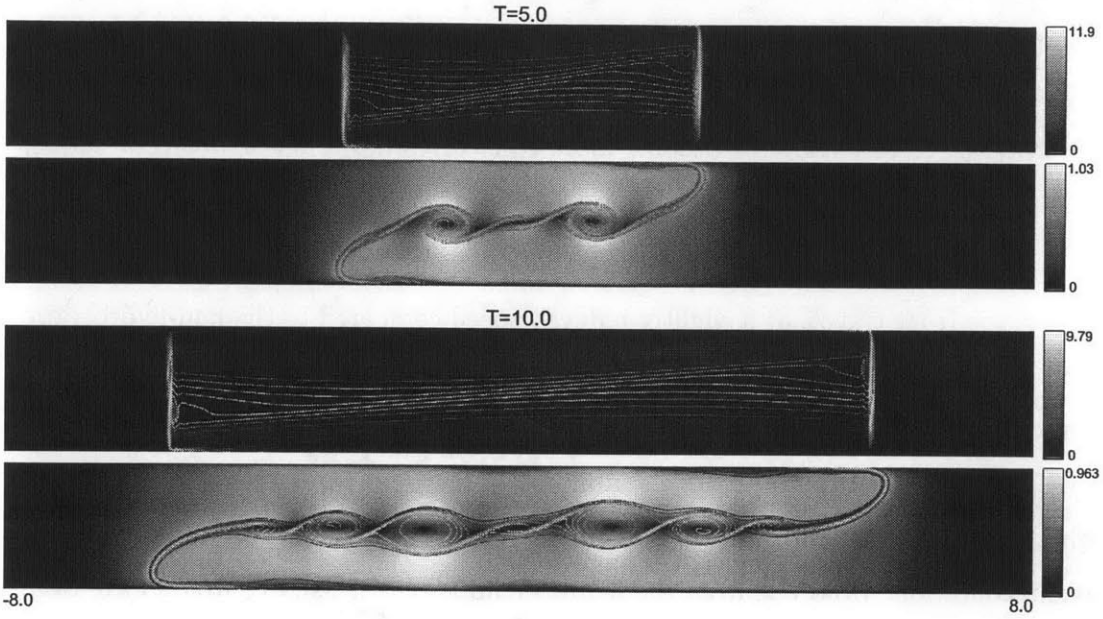


Figure 4-6: Lock exchange benchmark over domain $L \times H \times B = 8 \times 2 \times 10^{-4}$ at $T = 5$ (top two plots) and $T = 10$ (bottom two plots) for the hydrostatic (first and third plots) and non-hydrostatic (second and last plots) cases at $Gr = 1.25 \times 10^6$. Both use $N = 100 \times 400$ elements, p = 1, $\Delta t = 0.001$, and a second-order accurate time-integrator. Density contours are plotted over velocity magnitude.

density contours are less oscillatory and our front moves faster (see Table [4.2]). This may be because Fringer et al. (2006) used a diffusive upwind advection scheme and no diffusion for their tracer field, or because they used an asymmetric free-slip boundary condition on the top boundary. Secondly, we find that our 3D non-hydrostatic results correspond to our 2D results from §3.8.4, that is, we find the same Froude number for both setups, very close to those of Härtel et al. (2000).

Similar to Fringer et al. (2006), we find that the maximum magnitude of to total flow velocity is approximately an order of magnitude larger for the hydrostatic case ($\mathcal{O}(10)$) compared to the non-hydrostatic case ($\mathcal{O}(1)$), see Fig. [4-6]. This is primarily

due to a large vertical velocity at the front. In addition to being incorrect physically, this implies a smaller time-step for the hydrostatic simulation to satisfy the advection CFL condition, which would make the computational times comparable, depending on the implementation. Additionally, we observed that as the resolution of the discretization increases, the maximum velocity increases for the hydrostatic case. For example, using 50 elements in the y direction yields a maximum velocity at $T = 10$ of 5.68 compared to the 9.76 velocity when using 100 elements. Therefore, as the grid resolution increases, it might become more efficient to use a non-hydrostatic model due to the CFL restriction when treating the advection terms explicitly.

Comparing our non-hydrostatic and hydrostatic simulations, we first note that the hydrostatic front travels at a slightly reduced speed compared to the non-hydrostatic front. This is even though the hydrostatic simulation cannot exactly satisfy the no-slip boundary condition, which can be observed from the density contours in Fig. [4-6]. If we assume there are no Taylor instabilities in the boundaries, then the heavy (or light) fluid at the top (or bottom) boundary should remain at $x = 0, y = 0$ due to the no-slip condition. However, from the figure we notice the density contours have been displaced from $x = 0$. The same is true for the non-hydrostatic simulation, but its contours are closer to $x = 0$, but in this case the discrepancy is only due to numerical discretization errors (and possible mixing due to Taylor instabilities in the boundary layer). That is, the thin boundary near the top (or bottom) surface where heavy (or light) fluid should remain cannot be resolved with the present discretization. Additionally, the density contours for the hydrostatic simulation are significantly diffused compared to the non-hydrostatic runs. Additional benefits of the non-hydrostatic solver therefore include: improved front-speed evolution, improved no-slip boundary layer modeling, and improved mixing characteristics.

Finally, we observe excellent symmetry of the solution, even though an asymmetry is introduced by the vertical integration of the Boussinesq term. Only at coarse resolution (not shown) were we able to observe asymmetries in the density field. This suggests that our discretization of the hydrostatic pressure term is sufficiently accurate.

4.5 Summary

In this chapter we formulated our new HDG finite element ocean modeling schemes. We began by reviewing the derivation of our continuous equations, drawing attention to the errors from various approximations, and to the numerical consequences of various choices. Following this, we described the temporal and spatial discretization of free-surface or rigid-lid and non-hydrostatic or hydrostatic formulations. We showed that the hydrostatic formulation does not suffer from time-splitting errors due to the projection method used, while the non-hydrostatic form will have negligibly small splitting errors ($\mathcal{O}(\nu_z \Delta t)$ for the pressure). We again derived consistent expressions for the HDG stability parameters for the free-surface elevation and non-hydrostatic pressure.

We used a tidal flow benchmark with an analytical solution to show that our free-surface is correctly formulated and implemented. Then, using the lock exchange benchmark, we verified that our rigid-lid hydrostatic and non-hydrostatic ocean solvers are correctly formulated and implemented. Specifically it tested the various vertical integration terms that were not present in the 2D simulations of §3.8.4. We have discussed that when non-hydrostatic terms become significant, the velocities computed by a hydrostatic code can become much larger than those computed by the non-hydrostatic code, such that the cost of stable non-hydrostatic and hydrostatic simulations at these resolutions can become similar. Then, we argued that for the lock-exchange benchmark, the non-hydrostatic version improves the calculation of the front-speed, the no-slip boundary layer, and the mixing.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Implementation of Novel Non-hydrostatic Ocean Modeling Schemes, Consistency and Algorithmic Properties

In this chapter, we cover implementation issues for the formulation described in §4. Specifically, we detail our numerical flux choices for the advection terms, and we describe our implementation of the moving free-surface that yields a consistent and conservative discretization. For the moving free-surface, we use an arbitrary Lagrangian-Eulerian (ALE) formulation, and we combine this with an IMEX-RK time-stepper. Then, for completeness, we describe the discretization of the explicit terms in our time-integration scheme. That is, we detail the implementation of vertical integrals, the horizontal diffusion, and the Coriolis forcing.

5.1 Consistency and conservation

For schemes to be conservative it is important for the numerical operators to be consistent. This issue has been dealt with by various authors recently as new finite element ocean models are being developed (Wang et al., 2008, White et al.,

2008). However, to our knowledge, the specific issues involving hybrid discontinuous Galerkin (HDG) methods have not been examined. Drawing from what has been found by previous authors, we explain the modifications required for HDG elements. The issue of conservation is directly related to the numerical divergence operator, which appears in the advection operator, whose discretization is discussed next.

5.1.1 Advection

Consider a generic collection of tracer fields $\boldsymbol{\phi} = [\phi_0, \phi_1, \ldots]$ being advected by a velocity field $\mathbf{v}$. The pure advection problem, in this case, is:

$$\frac{\partial \boldsymbol{\phi}}{\partial t} + \nabla \cdot (\mathbf{v}\boldsymbol{\phi}) = 0.$$

It is discretized using standard discontinuous Galerkin methods as:

$$\left(\frac{\partial \boldsymbol{\phi}}{\partial t}, \boldsymbol{\theta}\right)_K + (\nabla \cdot (\mathbf{v}\boldsymbol{\phi}), \boldsymbol{\theta})_K + \left\langle \widehat{\mathbf{v} \cdot \hat{\mathbf{n}}\boldsymbol{\phi}} - \mathbf{v} \cdot \hat{\mathbf{n}}\boldsymbol{\phi}, \boldsymbol{\theta} \right\rangle_{\partial K} = 0. \tag{5.1}$$

What remains, and what differentiates various discontinuous Galerkin methods, is the choice for the inter-element flux quantity $\widehat{\mathbf{v} \cdot \hat{\mathbf{n}}\boldsymbol{\phi}}$. Now, for our projection method described in §4, the choice of the inter-element velocity flux is related to the discrete divergence of the velocity field. To see this, consider the advection of constant tracer fields, where the constant is $\boldsymbol{\phi} = [1, 1, \ldots]$. In that case, (5.1) reduces to

$$(\nabla \cdot \mathbf{v}, \boldsymbol{\theta})_K + \left\langle \widehat{\mathbf{v} \cdot \hat{\mathbf{n}}} - \mathbf{v} \cdot \hat{\mathbf{n}}, \boldsymbol{\theta} \right\rangle_{\partial K} = 0, \tag{5.2}$$

which is the discrete divergence equation. Now, for traditional finite element methods, $\widehat{\mathbf{v} \cdot \hat{\mathbf{n}}}$ is chosen as a combination between the mean and jump in velocity across elements

$$\widehat{\mathbf{v} \cdot \hat{\mathbf{n}}} = \{\{\mathbf{v}\}\} \cdot \hat{\mathbf{n}} + a [\![\mathbf{v} \cdot \hat{\mathbf{n}}]\!],$$

for some spatially and temporally variable a. As in (2.81), a central scheme is recovered for $a = 0$, and as in (2.82), an upwind scheme is recovered for $a = \frac{1}{2}\text{sign}(\mathbf{v} \cdot \hat{\mathbf{n}})$, but other values of a are also possible, and may be based on Riemann solvers. With the HDG method, a third choice is available, (2.83),

$$\widehat{\mathbf{v} \cdot \hat{\mathbf{n}}} = \boldsymbol{\lambda} \cdot \hat{\mathbf{n}}.$$

While any of these fluxes are appropriate when calculating the discrete divergence of the first velocity predictor $\bar{\mathbf{v}}$, numerical experiments indicate that $\widehat{\bar{\mathbf{v}} \cdot \hat{\mathbf{n}}} = \bar{\boldsymbol{\lambda}} \cdot \hat{\mathbf{n}}$ gives accurate results. However, only the hybridized discontinuous Galerkin flux may be used for the second velocity predictor $\bar{\bar{\mathbf{v}}}$ and when advecting tracers with the final divergence-free velocity, $\mathbf{v}$. This is because the second velocity predictor and divergence-free velocity require unique corrections on the HDG edge-space, see (4.97), (4.107), (4.102), and/or (4.111). Therefore, without using the HDG flux on the edges, the final velocity will not be numerically divergence-free.

What remains is to define the value for $\widehat{\boldsymbol{\phi}}$. Again, the same choices exist,

$$\widehat{\boldsymbol{\phi}} = \{\{\boldsymbol{\phi}\}\} + a \left[\!\left[\boldsymbol{\phi} \hat{\mathbf{n}}\right]\!\right] \cdot \hat{\mathbf{n}},$$
$$\widehat{\boldsymbol{\phi}} = \boldsymbol{\lambda}_{\boldsymbol{\phi}}.$$

but we are not restricted as we were for the velocity, even in the case for $\boldsymbol{\phi} = \mathbf{v}$. This is because the velocity should be discretely divergence free, while there is no such restriction on $\boldsymbol{\phi}$. While we expect the HDG flux to give the most accurate answer, we use an upwind flux for stability. Therefore, in our case we use

$$\widehat{\mathbf{v} \cdot \hat{\mathbf{n}} \boldsymbol{\phi}} = \boldsymbol{\lambda} \cdot \hat{\mathbf{n}} \left[\{\{\boldsymbol{\phi}\}\} + \frac{1}{2}\text{sign}(\mathbf{v} \cdot \hat{\mathbf{n}}) \left[\!\left[\boldsymbol{\phi} \hat{\mathbf{n}}\right]\!\right] \cdot \hat{\mathbf{n}} \right].$$

Next, we address the conservation and consistency issues related to the free-surface and the time-varying movable domain.

5.1.2 Free-surface and moving and stationary meshes

To deal with the movement of the free-surface, we use an arbitrary Lagrangian-Eulerian (ALE) method, following Persson et al. (2009). As described in §3.3, because of efficiency and convenience, we implement our finite element method by performing calculations on a reference element, using coordinate transformation factors to obtain the correct results. To calculate the transformation factors, we define a time-variable mapping function in terms of our polynomial basis $\mathbf{x}(\boldsymbol{\xi}, t) \approx \mathbf{x}_i(t)\theta(\boldsymbol{\xi})$, where in §3.3 this mapping was constant in time. We now describe the general ALE method, summarizing the results of Persson et al. (2009), and following this we detail our particular solution method.

Consider the general system of conservation laws (of which the Boussinesq equation is a subset):

$$\frac{\partial \boldsymbol{\phi}(\mathbf{x}, t)}{\partial t} + \nabla_{\mathbf{x}} \cdot \mathbf{F}\left(\boldsymbol{\phi}(\mathbf{x}, t), \mathbf{x}, t\right) = 0, \tag{5.3}$$

where $\nabla_{\mathbf{x}}$ is the gradient taken in the physical space. This can be re-written in the stationary reference domain as

$$\frac{\partial \mathcal{J}\boldsymbol{\phi}(\boldsymbol{\xi}, t)}{\partial t} + \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J}\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}\mathbf{F}\left(\boldsymbol{\phi}(\boldsymbol{\xi}, t), \boldsymbol{\xi}, t\right) - \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J}\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}\boldsymbol{\phi}(\boldsymbol{\xi}, t)\mathbf{v}_m(\boldsymbol{\xi}, t) = 0, \tag{5.4}$$

where $\mathcal{J} = \det\left[\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right]$ is the Jacobian, $\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}$ is the Jacobian matrix of size $d \times d$, $\nabla_{\boldsymbol{\xi}}$ is the gradient taken in the reference space, and $\mathbf{v}_m(\boldsymbol{\xi}, t) = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ is the mesh velocity. This formulation puts a restriction on the Jacobian which is often referred to as the Geometric Conservation Law (GCL) (see Thomas (1979)). It can be seen by considering the evolution of a constant field. For example, for $\boldsymbol{\phi} = 1$, (5.4) becomes:

$$\frac{\partial \mathcal{J}}{\partial t} - \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J}\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}}\mathbf{v}_m(\boldsymbol{\xi}, t) \neq 0,$$

which will not be zero discretely, because the Jacobian is not exactly integrated by the numerical scheme. The error will be small for high-order discretizations, but it can

be corrected by slightly modifying the scheme (Persson et al., 2009). First, defining a predictor Jacobian, one needs to solve

$$\frac{\partial \bar{\mathcal{J}}}{\partial t} - \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}} \mathbf{v}_m(\boldsymbol{\xi}, t) = 0. \tag{5.5}$$

Second, modifying (5.4), and simplifying notation we have

$$\frac{\partial (\bar{\mathcal{J}} \mathcal{J}^{-1}) \mathcal{J} \phi}{\partial t} + \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}} \mathbf{F} - \nabla_{\boldsymbol{\xi}} \cdot \mathcal{J} \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}} \phi \mathbf{v}_m = 0. \tag{5.6}$$

The main difference between (5.6) and (5.4) is the Jacobian correction ($\bar{\mathcal{J}} \mathcal{J}^{-1}$), which ensures numerical conservation of tracers.

This completes our brief summary of the approach by Persson et al. (2009), which guarantees the numerical conservation of quantities. What remains is to explain the order of operations when using the time split Projection method and an IMEX-RK time integration scheme.

Note that the discrete free-surface equation is is treated completely implicitly. Consider this equation (4.41) discretized using the IMEX scheme, also substitute for $F^{\eta,k+1}_{\Delta t} = -\nabla_{xy} \cdot \int_{-H}^{\eta^k} \bar{\mathbf{u}}^{k+1} \mathrm{d}z$:

$$\begin{aligned} \delta\eta^i - a_{ii} \Delta t \left(\Delta t \nabla_{xy} \cdot \left[g(\eta^{i-1} + H) \nabla_{xy} \delta\eta^i \right] \right) = \\ \sum_{j=1}^{j=i-1} a_{ij} \Delta t^2 \nabla_{xy} \cdot \left[g(\eta^{j-1} + H) \nabla_{xy} \delta\eta^j \right] + \sum_{j=1}^{j=i} \Delta t a_{ij} \nabla_{xy} \cdot \int_{-H}^{\eta^{j-1}} \bar{\mathbf{u}}^j \mathrm{d}z \end{aligned} \tag{5.7}$$

Starting with the time-integration from k to $k+1$ at stage $i-1$ of the IMEX-RK scheme, with the domain at Ω^{i-1}, the time-integration procedure is as follows.

1. Using Ω^{i-1}, calculate the explicit terms at stage i. This does not include a predictor for the free surface.

2. Update the mesh using the most-recent free-surface height, which gives $\Omega^{i-1} \mapsto \Omega^i$. Also advance the predictor Jacobian explicitly to $\bar{\mathcal{J}}^i$ (5.5).

3. Re-calculate Jacobians and transformation factors on the new mesh, and rebuild

implicit operators for the new domain.

4. Apply the Jacobian correction to the right-hand-side vectors of the predictor velocities and tracers. For example, for the tracers, the right-hand-side RHS^i_ϕ becomes:

$$\mathrm{RHS}^i_\phi = \frac{1}{\left(\bar{\mathcal{J}}\mathcal{J}^{-1}\right)^i}\mathbf{F}^{\phi,i}_{\Delta t}.$$

5. Set $\bar{\mathcal{J}}^i = \mathcal{J}^i$ for the next time-step or IMEX-RK stage.

6. Using Ω^i, calculate the implicit terms, to obtain the predictor velocities and stage-final tracers. Note, the free-surface predictor, η^k (the free-surface height at the start of the stage) is included with the implicit terms on the right-hand side of the velocity predictor .

7. Using the predictor velocities, calculate $\delta\eta^i$ as in (5.7).

8. Correct the predictor velocities with $\delta\eta^i$ (4.96)–(4.97), and correct the free-surface (4.98).

9. Perform additional calculations for non-hydrostatic or hydrostatic solver.

10. Calculate the implicit terms for the momentum equations at stage i as in (2.97). Also, calculate the implicit terms for the free-surface equation at stage i as

$$\mathbf{F}^\eta_i = \frac{\eta_i - \eta^k}{\Delta t} - \frac{1}{a_{ii}}\left\{\sum_{j=0}^{i-1} a^{im}_{ij}\mathbf{F}^\eta_j\right\}.$$

11. Repeat steps 1–10 for every IMEX-RK stage.

Step 4 comes from the discretization of (5.6). To see this, consider a first-order time

discretization of (5.6)

$$\begin{aligned}
\frac{(\bar{\mathcal{J}}\mathcal{J}^{-1})^{k+1}\mathcal{J}^{k+1}\boldsymbol{\phi}^{k+1}}{\Delta t} &= \frac{(\bar{\mathcal{J}}\mathcal{J}^{-1})^{k}\mathcal{J}^{k}\boldsymbol{\phi}^{k}}{\Delta t} - \nabla_{\boldsymbol{\xi}}\cdot\mathcal{J}^{k}\left[\frac{\partial\boldsymbol{\xi}}{\partial\mathbf{x}}\right]^{k}\mathbf{F}^{k} + \nabla_{\boldsymbol{\xi}}\cdot\mathcal{J}^{k}\left[\frac{\partial\boldsymbol{\xi}}{\partial\mathbf{x}}\right]^{k}\boldsymbol{\phi}^{k}\mathbf{v}_{m}^{k}, \\
&= \frac{\mathcal{J}^{k}\boldsymbol{\phi}^{k}}{\Delta t} - \nabla_{\boldsymbol{\xi}}\cdot\mathcal{J}^{k}\left[\frac{\partial\boldsymbol{\xi}}{\partial\mathbf{x}}\right]^{k}\mathbf{F}^{k} + \nabla_{\boldsymbol{\xi}}\cdot\mathcal{J}^{k}\left[\frac{\partial\boldsymbol{\xi}}{\partial\mathbf{x}}\right]^{k}\boldsymbol{\phi}^{k}\mathbf{v}_{m}^{k}, \\
&= \mathbf{F}_{\Delta t}^{\boldsymbol{\phi},k}, \\
\Rightarrow \frac{\mathcal{J}^{k+1}\boldsymbol{\phi}^{k+1}}{\Delta t} &= \frac{1}{(\bar{\mathcal{J}}\mathcal{J}^{-1})^{k+1}}\mathbf{F}_{\Delta t}^{\boldsymbol{\phi},k}
\end{aligned}$$

where the second equality follows from setting $\bar{\mathcal{J}}^{k+1} = \mathcal{J}^{k+1}$ in step 5. These steps are followed for each stage in an IMEX-RK integrator.

Using this method, the system of equations can be advanced with a moving-mesh free-surface, while maintaining conservation numerically. While most of the modifications to the integration scheme are minor, step 3 requires the re-calculation of matrices for the implicit terms (when a matrix-based solver is used), which may be expensive. For small free-surface displacements, this step can be omitted without introducing significant error, thereby increasing the efficiency of the scheme. For a matrix-free implementation, however, little modification should be required, and it is likely that preconditioners calculated for the mean free-surface (undisturbed ocean) would be sufficient. As such, a matrix-free solver is recommended when using a moving-mesh free-surface.

As a final note, since the free-surface field is discontinuous, the domain will also be discontinuous. Fortunately, this is not an issue, since the calculations are made on the reference domain, which is time-invariant and remains continuous. Some care is required when calculating the transformation factors for quadrature-free implementations, but this is discussed in §3.3 (see (3.7)).

5.2 Discretization of explicit terms

In this section we detail the numerical discretization of the explicit terms, which were not described in §4. The advection term was described in §5.1.1, so what re-

mains is to detail the vertical integration required in the free-surface, rigid-lid, and hydrostatic pressure equations, the horizontal diffusion terms, and the remaining source-terms.

5.2.1 Vertical integrals

Here, we discuss the discretization of the vertical integration terms on the right-hand-side of the free-surface or rigid-lid equations. Also, while we have already discussed the HDG discretization of the vertical integration to compute the vertical velocity in §4.3.2, we will also describe alternative formulations, explaining their advantages and disadvantages.

2D total integrals $\int_{-H}\eta$: In the free-surface or rigid-lid equations, we need to integrate the velocity or divergence of velocity. Unlike the calculation of the vertical velocity or hydrostatic pressure, we only need the 2D, total integral field $\int_{-H}^{\eta}\phi \mathrm{d}\zeta$, as opposed to the 3D field $\int_{z}^{\eta}\phi \mathrm{d}\zeta$. As such, the single-derivative discontinuous Galerkin formulation is appropriate. That is, to calculate the integral $\Phi = \int_{-H}^{\eta}\phi \mathrm{d}\zeta$, we let $\nabla_z \Phi = \phi$, and discretize this as

$$(\nabla_z \Phi, \theta)_K + \left\langle \widehat{\Phi} - \Phi, \hat{n}_z \theta \right\rangle_{\partial K} = (\phi, \theta)_K , \tag{5.8}$$

with boundary condition

$$\widehat{\Phi}|_{\partial\Omega_\star} = 0, \tag{5.9}$$

where the direction of integration depends on whether the boundary condition is applied at the top $\partial\Omega_\star = \partial\Omega_\eta$ or bottom $\partial\Omega_\star = \partial\Omega_{-H}$ of the domain. The one-sided flux $\widehat{\Phi} = \Phi^\star$ is used, where the top face is used if integration is from the top to bottom, and vice versa. This HDG integration happens over the 3D domain and yields a 3D field, so the final result is found by extracting the value of the field at the bottom if integration is from top to bottom.

This approach gives an efficient solution method because the one-sided flux allows

a marching scheme. That is, if the boundary condition is applied at the top, the top elements may be solved completely without knowledge of the element below it. Following this, the element below the top elements may be solved, and so forth.

3D partial integrals $\int_z^\eta$ and second derivatives: It is still possible to use a second-derivative formulation discretized using the HDG approach as in §4.3.2. However, for those schemes the solution method is coupled in the vertical direction, and a marching approach can no longer be used. Also, care needs to be taken because the final solution is extracted from the HDG flux on the edge-space, and not from the elements. Since the single-derivative and second-derivative schemes give numerically equivalent results for the total vertical integral, the single-derivative form is preferred because of its computational efficiency.

Finally, an alternative second-derivative formulation can be obtained by simply taking an additional derivative of the first-derivative form:

$$\nabla_z^2 \Phi = \nabla_z \phi. \tag{5.10}$$

This can also be discretized using HDG, and it can be used for all three vertical integrals. The advantage of this scheme is that information from the HDG edge-space will be utilized, which may improve accuracy. However, it has a number of disadvantages over the other formulations. First, it is not appropriate for finding the vertical velocity since it is not consistent with the numerical divergence equation; using this scheme, the vertical derivative of the velocity will be numerically divergence free, but due to numerical errors the actual velocity will not be. Next, for the hydrostatic pressure, this form is actually more consistent with the continuity equation, since the vertical derivative of the density will be proportional to the second vertical derivative of pressure. As such, it may be useful to calculate the hydrostatic pressure in the free-surface formulation. However, it is more difficult to solve for the horizontal gradients of the hydrostatic pressure, and the scheme used in §5.2.2 has computational advantages for the rigid-lid formulation. Last, the single-derivative form has computational advantages for the total vertical integral from this section, with no apparent

loss of accuracy. As such, the second-derivative scheme from this section (5.10) is not used for our implementation.

Thus, we have detailed the discretization of the total vertical integration required for the right-hand-sides of the free-surface and rigid-lid equations. We also briefly discussed an alternate second-derivative formulation which we do not use, but could be employed for the hydrostatic pressure in the free-surface formulation.

5.2.2 Explicit hydrostatic pressure

Here we describe the HDG discretization of the hydrostatic pressure term. There are some slight differences between the free-surface and rigid-lid calculations for this term, but both are based on a second derivative form, similar to that used to calculate the vertical velocity in §4.3.2. The major difference between the free-surface and rigid-lid is in the boundary conditions, and the rigid-lid form includes an easily-calculated 2D divergence update (see §4.1.2).

We could formulate the integration using only a single derivative, however we choose to use two derivatives. The main advantage of the single-derivative approach is that it can be solved efficiently using a marching approach. The advantages of the two derivative approach is that it is consistent with our continuity equation discretization, we can efficiently apply a correction in the rigid-lid case, and the accuracy is improved. Since the efficiency is not drastically impacted by this choice, we prefer the two-derivative form. Thus, to solve for the hydrostatic pressure, we introduce the depth-integrated hydrostatic pressure contribution P_{hyd} (see §4.1.2), which gives

$$-\nabla_z^2 \nabla_{xy} P_{hyd}^{k+1} = -g \nabla_{xy} \rho'^{,k}.$$

with boundary conditions for the free-surface case as

$$(\nabla_z \nabla_{xy} P_{hyd}^{k+1}|_{\partial\Omega_\eta}) \cdot \hat{\mathbf{n}}_z = \int_{\eta^k}^{\eta^k} g \nabla_{xy} \rho'^{,k+1} \mathrm{d}z \cdot \hat{\mathbf{n}}_z = 0,$$

$$\nabla_{xy} P_{hyd}^{k+1}|_{\partial\Omega_{-H}} = \int_{-H}^{-H} \left\{ \int_{-H}^{\eta^k} g \nabla_{xy} \rho'^{,k+1} \mathrm{d}z \right\} \mathrm{d}z = 0,$$

and for the rigid-lid case as

$$(\nabla_z \nabla_{xy} P_{hyd}^{k+1}) \cdot \hat{\mathbf{n}}_z = \nabla_{xy} p_{hyd}^{k+1} \cdot \hat{\mathbf{n}}_z = 0.$$

Note that we are solving for the gradient ($\nabla_{xy} P_{hyd}$) directly. We could have solved for the integrated pressure, then taken its gradient, but as discussed in §4.1.2, the present form has advantages for sigma-coordinate models. Next, these equations are discretized using HDG fluxes locally as

$$\begin{aligned}
\left(\nabla_{xy} p_{hyd}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla_z \nabla_{xy} P_{hyd}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \nabla_{xy} P_{hyd}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} &= \left\langle \boldsymbol{\lambda}_{\nabla_{xy} P_{hyd}}^{k+1}, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} \\
- \left(\nabla_z \nabla_{xy} p_{hyd}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau_{P_{hyd}} \nabla_{xy} P_{hyd}^{k+1} \cdot \hat{\mathbf{n}}_z, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} &= \left\langle \tau_{P_{hyd}} \boldsymbol{\lambda}_{\nabla_{xy} P_{hyd}}^{k+1} \cdot \hat{\mathbf{n}}_z, \hat{\mathbf{n}}_z \cdot \boldsymbol{\theta} \right\rangle_{\partial K} \\
&\quad - \left(g \nabla_{xy} \rho'^{,k+1}, \boldsymbol{\theta}\right)_K - \left\langle g \lambda_{\rho'}^{k+1} - g \rho'^{k+1}, \hat{\mathbf{n}}_{xy} \cdot \boldsymbol{\theta} \right\rangle_{\partial K},
\end{aligned} \tag{5.11}$$

and with the globally coupled equation

$$\begin{aligned}
\left\langle \left[\left[\nabla_{xy} p_{hyd}^{k+1} \hat{n}_z - \tau_{P_{hyd}} \left(\nabla_{xy} P_{hyd}^{k+1} - \boldsymbol{\lambda}_{\nabla_{xy} P_{hyd}}^{k+1} \right) \right]\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \left\langle g_N, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon \\
\boldsymbol{\lambda}_{\nabla_{xy} P_{hyd}}^{k+1} |_{\partial \Omega_\eta} &= 0.
\end{aligned} \tag{5.12}$$

This solves directly for $\nabla_{xy} p_{hyd}^{k+1} = \nabla_{xy} \nabla_z P_{hyd}^{k+1}$ on the element, and we do not need to calculate a value for the hydrostatic pressure on the HDG edge-space. This completes the procedure for the free-surface case, but the rigid-lid case requires one additional step.

In §4.1.2, we derived a 2D correction (4.16) for the rigid-lid flat bottom case, such that the density forcing and hydrostatic pressure would balance to give a depth-integrated divergence-free contribution. This correction can easily be calculated using available quantities when H is constant

$$\begin{aligned}
\nabla_{xy} p_s^{hyd,k+1} &= -\frac{\int_{-H}^{0} \nabla_{xy} p_{hyd}^{k+1} \mathrm{d}z}{H}, \\
&= -\frac{\nabla_{xy} P_{hyd}^{k+1}|_{-H}}{H}.
\end{aligned}$$

This shows one of the major advantages of calculating the hydrostatic pressure using two derivatives instead of one. For the rigid-lid formulation, then, the total contribution on the right-hand-side is therefore $\nabla_{xy} p_{hyd}^{k+1} + \nabla_{xy} p_s^{hyd,k+1}$. This completes the discretization of the explicit hydrostatic pressure.

5.2.3 Horizontal diffusion

All of the horizontal diffusion terms are computed explicitly at each time-step, ideally using the discrete HDG variables predicted at the previous time-step. While we would prefer using an HDG flux for the explicit horizontal diffusion terms, we have shown that this scheme would be globally coupled (§2.5). Therefore, for computational efficiency we employ a standard locally discontinuous Galerkin (LDG) discretization (Cockburn and Shu, 1998a) for the horizontal diffusion. The to-be-determined horizontal diffusion $\mathbf{F}^{\text{diff}}$ is obtained from the following DG discretization:

$$\left(\nu_{xy}^{-1}\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla_{xy}\bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left\langle \hat{\bar{\mathbf{v}}}^{k+1} - \bar{\mathbf{v}}^{k+1}, \hat{\mathbf{n}}_{xy} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} = 0, \tag{5.13}$$

$$\left(\nabla_{xy} \cdot \bar{\mathbf{Q}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle (\hat{\bar{\mathbf{Q}}}^{k+1} - \bar{\mathbf{Q}}^{k+1}) \cdot \hat{\mathbf{n}}_{xy}, \boldsymbol{\theta} \right\rangle_{\partial K} = \left(\mathbf{F}^{\text{diff}}, \boldsymbol{\theta}\right)_K, \tag{5.14}$$

where

$$\begin{aligned} \hat{\bar{\mathbf{Q}}}^{k+1} &= \left\{\left\{\bar{\mathbf{Q}}^{k+1}\right\}\right\} + \frac{\hat{\mathbf{n}}_{xy}^+}{2}\left[\left[\bar{\mathbf{Q}} \cdot \hat{\mathbf{n}}_{xy}\right]\right] - \frac{\tau}{2}\left[\left[\bar{\mathbf{v}}^{k+1}\hat{\mathbf{n}}_{xy}\right]\right], \\ \hat{\bar{\mathbf{v}}}^{k+1} &= \left\{\left\{\bar{\mathbf{v}}^{k+1}\right\}\right\} - \frac{\hat{\mathbf{n}}_{xy}^+}{2}\left[\left[\bar{\mathbf{v}}^{k+1}\right]\right], \end{aligned} \tag{5.15}$$

or simply

$$\begin{aligned} \hat{\bar{\mathbf{Q}}}^{k+1} &= \bar{\mathbf{Q}}^{+,k+1} - \frac{\tau_{LDG}}{2}\left[\left[\bar{\mathbf{v}}^{k+1}\hat{\mathbf{n}}_{xy}\right]\right], \\ \hat{\bar{\mathbf{v}}}^{k+1} &= \bar{\mathbf{v}}^{-,k+1}. \end{aligned}$$

This term is solved in two steps. First the gradients are calculated (5.13), followed by the divergence of those gradients (5.14). After the edge quantities have been deter-

mined, the remaining calculations are element-local and require no matrix inversion, resulting in an efficient scheme. For a minimum dissipation scheme, $\tau_{LDG} = 0$ can be used. But for additional stability, a finite value can be used, however its magnitude will be limited by an explicit time-stepping stability condition.

The discretization of the diffusion terms for the tracer equations are the same. In this case $\bar{\mathbf{v}}^{k+1}$ is replaced by the collection of tracers $\boldsymbol{\phi}$.

5.2.4 Coriolis and Forcing terms

We chose to discretize the Coriolis terms explicitly, and we assume that the forcing terms are known and can be evaluated at the implicit time step ($k+1$ in our notation), to be used for the next time-step. As such, the discretization of these terms are trivial; since we pre-multiply with a mass-matrix inverse, the discretization requires the evaluation of the linear Coriolis and forcing terms at the nodal points.

For nonlinear forcing terms, this does introduce an interpolation error, which can be eliminated by projecting these nonlinear forcing terms onto the polynomial basis instead. That involves evaluating them at sufficiently many quadrature points, and performing a numerical integration. For efficiency, we incur the interpolation error (see §3.2), and we expect that for our problems of interest, the error should not be significant with an appropriate discretization (that is, with sufficient resolution).

5.3 Summary

In this chapter, we discussed implementation issues related to the advection terms, the moving free-surface, vertical integration terms, and explicit Coriolis and other forcing terms. We detailed our flux choices for the advection terms, where we use the HDG edge-space value for the velocity, and an upwind flux for the tracer. We walked through the correct time-integration procedure for each IMEX-RK stage to correctly implement the free-surface and moving-meshes. Then we covered the single-derivative formulation of the vertical integral, and we introduced an additional vertical integration discretization which we do not utilize. Finally, we listed the LDG discretization

for the horizontal diffusion, and explained that the Coriolis and other forcing terms are calculated by simply evaluating them at the nodal points.

Chapter 6

Numerical Sensitivity of Biogeochemical Models: High-Order Finite-Element Schemes

6.1 Introduction and motivation

Accurate modeling of biogeochemical-physical ocean dynamics is required for multiple scientific and societal applications, covering a wide range of time and space scales. With the increased understanding of biogeochemical interactions (Lalli and Parsons, 1997, Robinson et al., 2002b, Fennel and Neumann, 2004), ecosystems models have substantially improved in the past decades (Fasham et al., 1990, Hofmann and Lascara, 1998, Robinson and Lermusiaux, 1999, Hofmann and Friedrichs, 2002, Lynch et al., 2009). Coupled biogeochemical-physical models have been used from coastal regions e.g. (Anderson et al., 2005, Spitz et al., 2005, Ji et al., 2008, Stow et al., 2009) to basins and global ocean domains e.g. (Oschlies and Garcon, 1998, Rothstein et al., 2006, Doney et al., 2009). However, in light of the strong nonlinearities observed in biological processes, an important subject that has been largely overlooked is the

numerical requirements for such simulation studies. One of the major objectives of this chapter is to address such computational questions for reactive ocean tracers, directly including the latest advances in computational fluid dynamics e.g. (Chung, 2002, Ferziger and Peric, 2002, Lomax et al., 2003, Cebeci et al., 2005, Karniadakis and Sherwin, 2005) and multiscale ocean modeling (Deleersnijder and Lermusiaux, 2008).

Previous numerical ocean studies related to ours have primarily focused on passive or dynamic (density-related) tracer advections. The most significant progress include the results of Hecht et al. (1995), Hanert et al. (2004), Budgell et al. (2007), but none of these advances have dealt with higher order advection of reactive tracers on unstructured meshes with curved geometries. Iskandarani et al. (2005) applied and studied high-order schemes for passive tracer and density dynamics in two-dimensions, including Hecht et al. (1995)'s test and the gravitational adjustment of density in a channel of constant depth (Haidvogel and Beckmann, 1999), but they did not consider curved elements. Lévy et al. (2001) assessed five different low-order finite volume advection schemes for biological modeling and found a 30% difference in new production estimates, highlighting the need for careful numerical studies. In Bernard et al. (2009), high-order discontinuous Galerkin (DG) methods are used to solve tidal flows around shallow water islands with non-trivial geometries and using curved triangular meshes. Here, we are interested in biogeochemical tracers with possibly highly nonlinear reactive or source terms, and we compare a set of low to high order schemes, both in time and in space. Due to numerical discretization, we expect to observe phase errors, path-accumulated errors, and errors that modify the phytoplankton dynamics. To assess these errors, we employ the DG Finite Element Method (Cockburn, B., 1998), using both straight and curved elements, and we study a varied set of numerical properties. As in previous computational studies, we restrict our numerical analyses to two-dimensional (2D) flows, focusing on coupled dynamics in idealized straits.

Our ultimate dynamics motivation is to allow quantitative simulation studies of fundamental nonlinear biological-physical dynamics in coastal regions with com-

plex bathymetric features such as straits, sills, ridges and shelfbreaks. Such features strongly affect flows and, if they are shallow enough, one can expect biological responses in the euphotic zone. Multiple physical scales are possible, from rapid tidal effects to slow water-mass driven overflows, and biological resonances at some of these scales are likely. Our focus is on the numerical requirements prerequisite to such studies. Our work is partly inspired by our experience in coastal regions with complex geometries (Haley and Lermusiaux, 2010b), especially with steep shelfbreaks such as the Massachusetts Bay and Stellwagen Bank (Besiktepe et al., 2003), Middle Atlantic Bight shelfbreak (Lermusiaux, 1999), Monterey Bay shelfbreak (Haley et al., 2009), Taiwan region shelfbreak (Lermusiaux and Xu, 2010) and Philippine Archipelago Straits (Haley and Lermusiaux, 2010b). The latter effort particularly motivated the present work, within the context of the Philippines Experiment (PhilEx) which is a five-year joint research project focused on interdisciplinary modeling, data assimilation, and dynamical studies in the straits regions of the Philippine Archipelago to better understand, model and predict sub-mesoscale and mesoscale physical and biogeochemical dynamics in complex regions. For realistic PhilEx simulations, we employ our MIT Multidisciplinary Simulation, Estimation, and Assimilation Systems (MSEAS-Group, 2010). It includes a free surface hydrostatic ocean model over complex geometries with novel implicit schemes for telescoping nesting (Haley and Lermusiaux, 2010b). This physical model is coupled to biological models (Besiktepe et al., 2003), forced with multiscale barotropic tides (Logutov and Lermusiaux, 2008) and initialized with new objective mapping schemes specific for multi-connected domains (Agarwal, 2009, Agarwal and Lermusiaux, 2010). The multi-resolution nested domains cover very shallow regions with strong tides, steep bathymetries and the deep ocean. The MSEAS system was employed in real-time, assimilating data sets from ships, gliders and satellite remote sensing, and issuing daily physical-biological forecasts with dynamical descriptions and adaptive sampling guidance (Lermusiaux et al., 2009). The complex, nonlinear and multiscale biology in the region confirmed the need for the present computational studies.

Our work is part of an incubation for the next-generation of ocean modeling

systems, focusing on key numerical questions for biogeochemical dynamics. The biological model we employ is based on Flierl and McGillicuddy (2002), Burton (2009) and Ueckermann (2009). We restrict ourselves to a relatively simple model to focus on the numerics. However, the model is complex enough to reveal important characteristics and to complete a large number of parameter sensitivity studies which we can synthesize. We study three biological regimes, one with single stable points at all depths and two with stable limit cycles. We consider idealized simulations of biological patchiness which is commonly observed in the coastal ocean. For these regimes and interactions, we study a wide range of temporal and spatial discretizations. The results in this chapter have been published in (Ueckermann and Lermusiaux, 2010). In what follows, we give our dynamical problem statement, definitions, and notation in §6.2. The results of our varied numerical and scientific investigations are described in §6.4. Finally, our conclusions are stated in §6.5.

6.2 Dynamical Problem Statement

The biological dynamics are governed by the following advection-diffusion-reaction equations:

$$\frac{\partial \Phi}{\partial t} + \nabla \cdot (\mathbf{u}\Phi) - \kappa \nabla^2 \Phi = \mathbf{S}(\Phi, \mathbf{x}, t), \quad \text{in } \Omega \tag{6.1}$$

with boundary conditions

$$\begin{aligned} \Phi &= \mathbf{g}_D, \quad \text{on } \Gamma_D \\ (\mathbf{u}\Phi - \kappa \nabla \Phi) \cdot \hat{\mathbf{n}} &= \mathbf{g}_N, \quad \text{on } \Gamma_N \end{aligned} \tag{6.2}$$

where $\Phi(\mathbf{x}, t) = [\phi^1(\mathbf{x}, t), \ldots, \phi^{N_c}(\mathbf{x}, t)]$ is the vector of N_c biological components, $\mathbf{u}$ is the prescribed velocity field, κ is a positive diffusivity coefficient, $\mathbf{S}(\Phi, \mathbf{x}, t)$ is the biological reaction terms, and $\mathbf{g}_D$, $\mathbf{g}_N$ are the boundary conditions for the Dirichlet and Neumann boundaries respectively. Equations 6.1-6.2 are solved on the domain $\Omega \in \mathbb{R}^d$, where d is the dimension of the problem, with boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$ such

that $\Gamma_D \cap \Gamma_N = \emptyset$.

Since we are interested in strait dynamics, for the flowfield $\mathbf{u}$, we assume that earth rotational effects are negligible, which is true if the ratio of the strait width to the Rossby radius is small (Pedlosky, 1987, Signell, 1989, Bourgault and Kelley, 2004, Cushman-Roisin and Beckers, 2011). Additionally, for uniform geometry across the strait with a rigid lid approximation, a small Froude number, and a homogeneous density, the velocity field can be approximated as a potential flow field. A similar setup was used by Signell (1989) for tidal flows. The potential velocity $\mathbf{u}$ is obtained by solving for the stream function

$$\nabla^2 \psi = 0, \quad \text{in } \Omega \tag{6.3}$$

$$\mathbf{u} = \nabla \times \psi \tag{6.4}$$

with boundary conditions

$$\psi = h_D, \quad \text{on } \partial\Omega. \tag{6.5}$$

Note, potential flow is usually solved with a velocity potential, but here we chose to use the stream function because it has convenient boundary conditions for this problem.

6.3 Comparing numerical codes: Defining Efficiency, Accuracy, and Performance

To be clear, we use the term "efficiency" or "cost" to refer exclusively to the computational resources (elapsed-time, memory) required for a simulation, and we do not use "efficiency" to imply any degree of correctness of the solution. We reserve the term "accuracy" to refer to the correctness of the solution. Finally, here we also use the term "performance" as the combined consideration between efficiency and accuracy (Chapra and Canale, 2006).

Comparing different numerical schemes is not a straight-forward task (see Kubatko et al. (2009)). First, results are not universally applicable and are generally problem dependent. After focusing on a particular class of problems, the usual approach is to fix the computational efficiency of both schemes and then compare the accuracy, or vice versa. The scheme that performs better will then have a superior accuracy, since the efficiency will be the same for both. However, the efficiency of the scheme is dependent on its implementation, as well as the computer architecture on which the simulations are performed. A simple approach, then, is to fix the number of degrees of freedom (DOFs) of the different schemes, that is, having the same number of unconstrained parameters in both schemes. Because the DOFs are related to computational efficiency, this approach is useful for comparing similar numerical schemes with different implementations. However, it is not a good approach when comparing different numerical schemes where the computational cost per DOF is inherently and significantly different between the schemes, which is the case for comparisons between high order and low order schemes. Finally, conclusions drawn about the performance is also dependent on the particular definition of accuracy. The accuracy is normally defined in terms of a quantity useful to a particular researcher. Thus, researchers with different quantities of interest may draw different conclusions about the performance of a scheme. We address the efficiency issue by presenting results for multiple efficiencies, and we address the accuracy issue by using generic global error measures (see §6.3.1) and by using difference plots.

6.3.1 Error Norm Calculation

A sufficiently complete set of error measures should be used to assess the accuracy of the numerical solutions. In our set of error measures, we include point-wise error measures to quantify phase and path-accumulated errors that lead to a loss of point-wise accuracy. In addition, integrated or global error measures are included as a bulk indication of the accuracy. Finally, to assess the accuracy of the numerical scheme in reproducing the dynamics described by the mathematical model and its parameters, we compare the phase-space dynamics of the biology for the biological

patch simulation in §6.4.6. The point-wise and integrated error norms are numerically calculated as follows.

Unless indicated otherwise, the global domain L^2 norm $\|e\| = \left(\int_\Omega e^2 d\Omega\right)^{\frac{1}{2}}$ is calculated using the quadrature-based approach as described in §3.2. That is, the numerical solution is interpolated unto the quadrature points, the error $e = \phi_h - \phi$ is evaluated, and then multiplied by the quadrature weights and summed for an approximate integration. In some cases, we evaluate the global error using an interpolation approach (similar to quadrature-free) and this is mentioned when we do. In these cases, the error is evaluated at the nodal points, then the error is interpolated to the quadrature points, multiplied by quadrature weights, and summed. Where ambiguous, we indicate the quadrature-based error evaluation using $\|e\|^{qp}$ (quadrature points), and the interpolated error evaluation using $\|e\|^{nd}$ (nodal points).

The infinity norm $\|e\|_\infty = \max|e|$ is calculated by evaluating the error at nodal points, and taking the maximum absolute value.

6.3.2 Higher-Order Mesh Generation

Since higher order DG schemes have more degrees of freedom per element, a coarse mesh with large elements is required to keep a similar performance across discretizations. To obtain an accurate solution with a coarse, high-order discretization, it is necessary to use curved boundary interfaces, as will be demonstrated in Ueckermann and Lermusiaux (2010). Here we describe our new method for creating such a coarse, high-order curved mesh.

When curving the boundary of an element, care needs to be taken because it is possible to create an element where two of the interfaces cross. The left triangular element shown in Fig. [6-1] has the true circular geometry crossing one of the straight interfaces. To avoid this situation, we need to ensure that

$$h < 2\rho(x)\sin(\theta), \tag{6.6}$$

where h is the length of the element side bordering the boundary, $\rho(x) = \frac{|1+f'(x)|^{3/2}}{|f''(x)|}$

is the radius of curvature of the boundary described by $f(x)$, and θ is the minimum angle of the two angles on the edge bordering the boundary. The element shown on the right side of Fig. [6-1] illustrates this limiting case for an equilateral triangle, but our condition (6.6) is trivially extended to arbitrary triangles as shown by the dashed lines.

Using our criteria (6.6), we define the minimum edge spacing on the boundary as $h_{min} = 2\rho\sin(30°) = \rho$. Then, we let the minimum edge length grow linearly by a certain percentage (fit to 12% here) away from the boundary upto a specified minimum edge length. Using these criteria, we create coarse base-meshes, then uniformly refine these meshes to obtain finer discretizations. To create the meshes, we primarily used the free mesher *Distmesh* (Persson and Strang, 2004), but we also used *Gmsh* (Geuzaine and Remacle, 2009). *Distmesh* uses an implicit geometry representation, that is, we define the geometry by a distance function that gives the distance between a queried point and the nearest boundary. Using *Distmesh*, we create meshes with straight sides.

To curve the boundary interfaces, we use the same distance function provided to *Distmesh*, and numerically calculate the gradient of the distance function to the boundary. The normalized gradient vector provides the direction of translation, but to determine the magnitude of the translation a weight needs to be applied to the calculated distance. That is, $p_1^{new} = p_1^{old} + \mathcal{W} d \frac{\nabla d}{|\nabla d|}$, where d is the distance from the point p_1^{old} to the boundary, and $\mathcal{W}$ is the weight. Now, points on the straight boundary interface are translated to the true, curved boundary with a weight 1, and points on interior interfaces are not translated, i.e. having weights 0. Points in the volume have weights defined by the same weighting functions used to create the nodal basis, that is

$$\mathcal{W}(e_1) = \left(\frac{2\lambda_3}{2\lambda_3 + \lambda_1}\right)\left(\frac{2\lambda_2}{2\lambda_2 + \lambda_1}\right)$$

where the point is defined by the barycentric coordinates λ_i corresponding to vertices i, and e_1 is the curved boundary interface defined by vertices 2 and 3. For details of

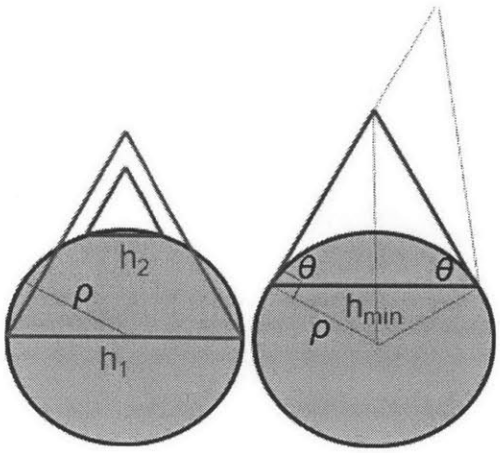


Figure 6-1: Minimum triangle angle criterion (6.6) demonstrated on a circle with equilateral triangles. $h_1 = 2\rho$ does not satisfy the criterion, $h_2 = 2/3\rho$ satisfies the criterion, and $h_3 = 2\rho\sin(\pi/3)$ demonstrates the limiting case. This result can be extended to arbitrary triangles as shown by the dashed lines.

this blending function, see Hesthaven and Warburton (2008).

The base mesh with three mesh refinements is shown in Fig. [6-2], and details of the base mesh for a curved and straight mesh boundary are shown in Fig. [6-3]. Using our criterion $h_{min} = \rho$, the minimum theoretical edge length for our geometry, that is a Gaussian bump defined by $\tilde{H}(x) = e^{-x^2}$, is $h_{min} = 0.25$. The mesh shown in Fig. [6-2]a) has a minimum edge length of $h_{min} = 0.2418$, close to the theoretical value.

6.4 Numerical Studies and Scientific Implications

Biogeochemical models may contain a large number of biological or chemical components (Hofmann and Friedrichs, 2002). The simplest models often only use Nutrient, Phytoplankton, and Zooplankton as components, and are commonly called NPZ models. More complicated models (Besiktepe et al., 2003) can be adaptive and contain many components. Each component requires the solution of an advection-diffusion-reaction (ADR) equation of the form (6.1). The source terms describe the commonly nonlinear "reactions", and may lead to stationary, periodic, or chaotic dynamics. For this numerical work, a non-dimensional version of a NPZ model (Flierl

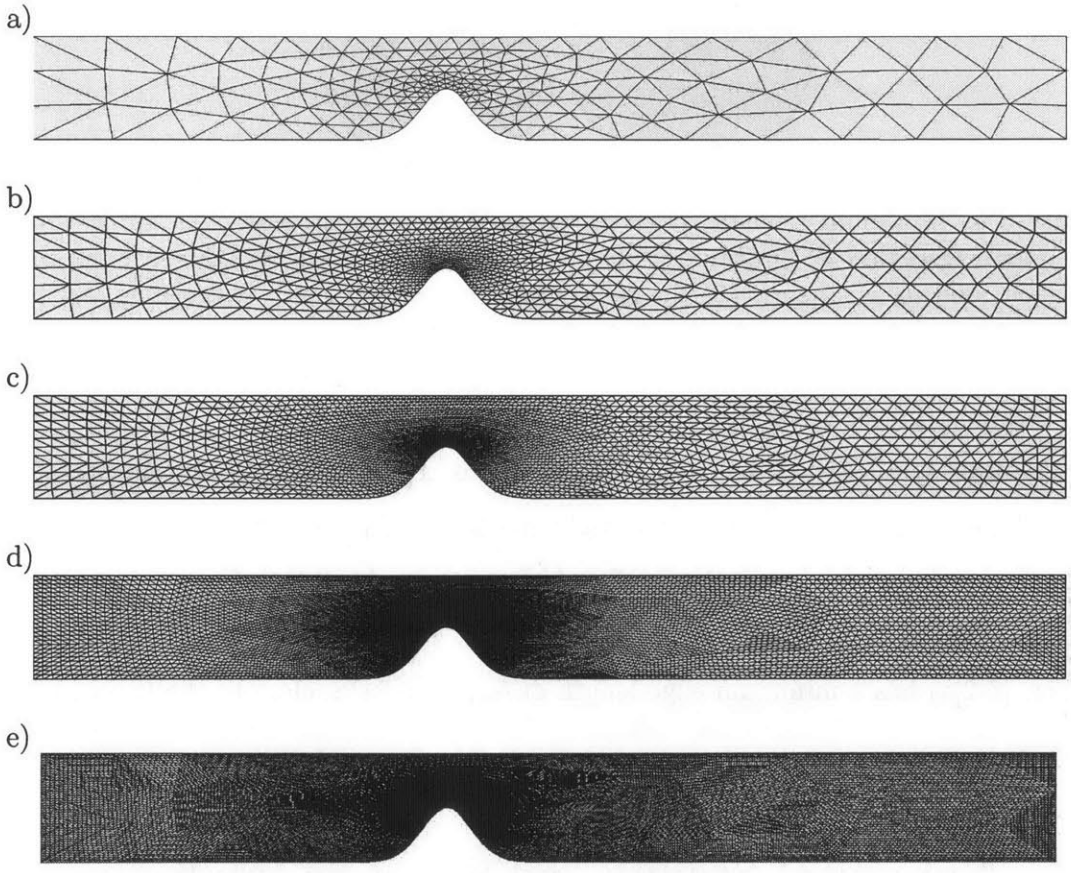
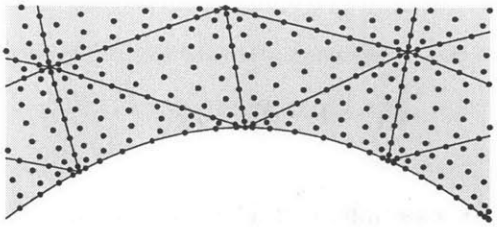


Figure 6-2: a) The base mesh ($g1$) with 350 elements. b) First ($g2$) (1,400 elements) c) second ($g3$) (5,600 elements), d) third ($g4$) (22,400 elements), and fourth ($g5$) (89,600 elements) grid refinements. The more-refined meshes are used for lower-order schemes whereas less-refined meshes are used for higher-order schemes such that the cost of the two schemes are comparable.

a)



b)

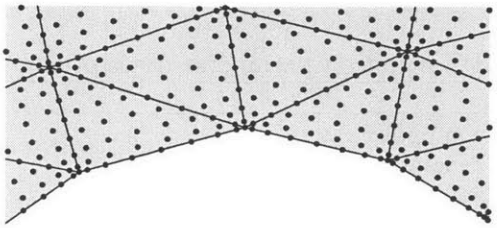


Figure 6-3: Details of ($g1$) using a a) curved and b) straight mesh for a $p = 8$ nodal basis.

and McGillicuddy, 2002) is used since it contains all characteristics required for our studies:

$$\frac{\partial \phi_N^*}{\partial t^*} + \nabla \cdot (\mathbf{u}^* \phi_N^*) - \nabla \cdot \frac{1}{P_e} \nabla \phi_N^* = -\mathcal{U}^* e^{z^*/h^*} \frac{\phi_P^* \phi_N^*}{\phi_N^* + k_s^*} + d_P^* \phi_P^* + d_Z^* \phi_Z^* + (1-a) g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*}) \quad (6.7)$$

$$\frac{\partial \phi_P^*}{\partial t^*} + \nabla \cdot (\mathbf{u}^* \phi_P^*) - \nabla \cdot \frac{1}{P_e} \nabla \phi_P^* = \mathcal{U}^* e^{z^*/h^*} \frac{\phi_P^* \phi_N^*}{\phi_N^* + k_s^*} - d_P^* \phi_P^* - g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*}) \quad (6.8)$$

$$\frac{\partial \phi_Z^*}{\partial t^*} + \nabla \cdot (\mathbf{u}^* \phi_Z^*) - \nabla \cdot \frac{1}{P_e} \nabla \phi_Z^* = -d_Z^* \phi_Z^* + a g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*}) \quad (6.9)$$

where $\phi_{(N,P,Z)}^* = \frac{\phi_{(N,P,Z)}}{\mathcal{N}_t}$, $u_x^* = \frac{u_x}{\bar{\mathbf{u}}}$, $u_z^* = \frac{u_z L}{\bar{\mathbf{u}} H}$, $x^* = \frac{x}{L}$, $z^* = \frac{z}{H}$, $t^* = \frac{t}{\bar{\tau}}$, the parameters are explained in Table [6.1], the non-dimensional groups with values are given in Table [6.2] with P_e the Peclet number and D^* the aspect ratio, the subscripts $(.)_N, (.)_P, (.)_Z$ refer to Nutrients, Phytoplankton, and Zooplankton respectively, $\nabla = \frac{\partial}{\partial x^*} + \frac{\partial}{\partial z^*}$, and lowercase z^* refers to the depth coordinate which is positive upward with $z^* = 0$ at the surface. Note that not all three equations (6.7)-(6.9) are required since the biological model satisfies the following conservation law for total nutrients, assuming

a closed ocean system:

$$\phi_N^* = 1 - \phi_P^* - \phi_Z^*. \tag{6.10}$$

The first equation (6.7), for example, could be eliminated in favor of (6.10); however here we still use (6.10) to check the conservation of the numerical schemes.

Table 6.1: NPZ equation parameter descriptions and units.

Parameter	Description	[units]
$\mathcal{U}$	Phytoplankton uptake rate	[1/day]
k_s	Saturation concentration of phytoplankton	[μmol/L]
d_P	Mortality rate of Phytoplankton	[1/day]
d_Z	Mortality rate of Zooplankton	[1/day]
g_{bio}	Grazing rate of Zooplankton	[L/(μmol·day]
a	Assimilation (efficiency) rate	[]
h	e-folding depth for light (photosynthesis)	[m]
ν_{bio}	Parameter for Ivlev form of grazing function	[L/μmol]
$\mathcal{N}_T$	Total biomass	[μmol/L]
$\bar{\mathbf{u}}$	Average inlet velocity	[km/day]
H	Height of bathymetry	[m]
D	Total maximum depth	[m]
L	Effective width of bathymetry	[km]
$\underline{\underline{\kappa}}$	Diffusion tensor (vertical and horizontal diffusion different)	[m^2/s]

Table 6.2: Values of the dimensionless numbers entering the NPZ equations (6.9), that are used in the examples for this manuscript. Bracketed triplets of values correspond to the three bio cases [1,2,3]. The other values are the same for the three cases.

Parameter	Value
$\mathcal{U}^* = \mathcal{U}\bar{\tau}$	7.5
$k_s^* = \frac{k_s}{\mathcal{N}_T}$	$[\frac{1}{30}, \frac{1}{50}, \frac{1}{100}]$
$d_P^* = d_P\bar{\tau}$	0.2
$d_Z^* = d_Z\bar{\tau}$	1
$g_\nu^* = \frac{g_{\text{bio}}\bar{\tau}}{\nu_{\text{bio}}}$	12.5
$a^* = a$	0.4
$h^* = \frac{h}{H}$	0.34
$\nu_{\text{bio}}^* = \mathcal{N}_T\nu_{\text{bio}}$	$[0.3, 0.5, 1]$
$P_e = \frac{\bar{\mathbf{u}}L}{\kappa}$	∞
$D^* = \frac{D}{H}$	2

The domain setup is depicted in Fig. [6-4] for the geometric parameter values given in Table [6.2]. With this setup, an upwelling of nutrients is created (see §6.4.2), and the study of idealized biological blooms, which may occur in straits or sills, can

be studied. In total, we consider three sets of parameter values, differing by the non-dimensional parameters ν^*_{bio} and k^*_s. In the absence of advection and diffusion, they lead to equations 6.9 with at most one physically relevant steady state solution (Burton, 2009). The three sets of non-dimensional parameters ν^*_{bio} and k^*_s correspond to biological dynamics with single stable points at all depths (bio case 1: $k^*_s = 1/30, \nu^*_{\text{bio}} = 0.3$), with stable limit cycles for depths around $z^* = 0.4 - 0.9$ and single stable points elsewhere (bio case 2: $k^*_s = 1/50, \nu^*_{\text{bio}} = 0.5$), and stable limit cycles everywhere in the euphotic zone (bio case 3: $k^*_s = 1/100, \nu^*_{\text{bio}} = 0.1$). The middle parameter values, bio case 2, correspond to those values used by Flierl and McGillicuddy (2002): they are idealized and not meant to represent a specific ocean region. We note that biological models with discontinuities in stable solutions are not always representative of nature. However, biology of interest is likely to have intrinsic oscillatory or chaotic time dependence, e.g. Flierl and McGillicuddy (2002). For our purposes, we address these issues by considering three sets of parameter values and so cover a range of biological dynamics. To handle non-physical negative concentrations due to numerics, we use $\max(0, \phi^*_{(N,P,Z)})$ when evaluating the source terms. In the absence of advection, a timescale of $\bar{\tau} = 1$[days] is used, while in the presence of advection, the advective timescale $\bar{\tau}_a = L/\bar{\mathbf{u}}$ is used, where $\bar{\mathbf{u}} = \int_{Inlet} |\mathbf{u}| dz$ is the average inlet velocity.

In our numerical study, we need to characterize the three dynamical regimes and their behaviour since these properties affect numerical errors. Specifically, for each dynamical regime, we study the balances of terms in (6.7)-(6.9): biological terms dominating; advection and biological terms balancing. When biological terms dominate, the advection is slow, and the problem reduces to a one dimensional problem, studied in §6.4.1. Finally, the case where the advection and biological terms are approximately balanced is studied in §6.4.2, with the effect of biological patches demonstrated in §6.4.6. Since the timescale of biology varies in depth, the advection and biological terms can only be exactly balanced for one depth. Also, for parameters sets where the biology exhibits steady behaviour, we determine the biological time-scale based on how quickly a perturbed initial condition returns to the steady one. While this results

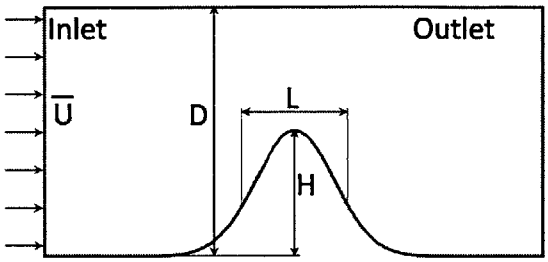


Figure 6-4: Test case domain with idealized strait bottom geometry described by $\tilde{H}(x) = He^{-\frac{x^2}{L^2}}$.

in many choices of approximately balanced terms, we will focus on one parameter set where $\bar{\tau}_a = 12.5$[days]. For more details on biological dynamics in straits, we refer to Burton (2009).

Results in this section will be reported for various grid and polynomial degree combinations, and the notation (grid number, polynomial degree) is used to denote this information. For example, $(g2, p4)$ refers to the second grid (Fig. [6-2]b) with a 4^{th} degree basis function. We provide a table with the number of DOFs and computational time estimates for the test cases we completed Table [6.3]. In what follows, we show the results of $(g1, p6)$ and $(g2, p5)$ for our high-order simulations and compare them to $(g4, p1)$. Normalizing by the average computational time of $(g4, p1)$ these simulations have relative computational times of 0.34, 1, and 1.2 for $(g1, p6)$, $(g4, p1)$, and $(g2, p5)$, respectively (see Table [6.3]). Following the discussion in §6.3, we note that, in terms of efficiency, $(g4, p1)$ and $(g2, p5)$ are comparable (in fact, $(g2, p5)$ would be cheaper if fully optimized, see Ueckermann and Lermusiaux (2010)). $(g1, p6)$ is included because it is comparable in accuracy to $(g4, p1)$ and it highlights the effect of under-resolution (here $g1$) when using higher-order schemes (here $p6$).

6.4.1 One-dimensional Biogeochemical Source terms studies

In this section we first illustrate the convergence of our numerical implementation. Following this, we examine the numerical behaviour of the biological source terms using three tests: *Perturbations of steady-states, Vertical resolution,* and *High-order bases.*

Table 6.3: Normalized run-times and DOFs for various grids/polynomial degree basis functions, for the simulations in §6.4.2. The times are normalized by the $(g4, p1)$ run-time, and numbers in parentheses are the DOFs.

Degree of basis	Grid 1	Grid 2	Grid 3	Grid 4	Grid 5
1	0.0014 (1,050)	0.014 (4,200)	0.12 (16,800)	**1.0** (67,200)	8.2 (268,800)
2	0.007 (2,100)	0.057 (8,400)	0.1 (33,600)	4.2 (134,400)	
3	0.026 (3,500)	0.21 (14,000)	1.8 (56,000)		
4	0.062 (5,250)	0.51 (21,000)	4.1 (84,000)		
5	0.15 (7,350)	**1.2** (29,400)			
6	**0.34** (9,800)	3.1 (39,200)			

Numerical convergence in space and time: Since an analytical solution to (6.9) does not exist, we verify the spatial implementation of the quadrature-free and quadrature-based source terms using the analytical test problem $\frac{\partial \phi}{\partial t} - 2\frac{\partial^2 \phi}{\partial z^2} = S(z,t)$ on $\Omega \in [-100, 0]$ integrating until $t = \pi/200$, with solution $\phi = \sin(t)\cos^2(\pi/50z)$ (for an appropriately chosen $S(z,t)$). We use a sufficiently small time-step, such that the errors are dominated by the spatial discretization. The results are shown in Table [6.4], with the norm of the error $e = \phi_h - \phi$ calculated as described in §6.3.1. From Table [6.4], we note that our implementation converges at the optimal rates for both the quadrature-based and quadrature-free treatments. While the error numbers are for a special case and not those for (6.7)-(6.9), they show that the solution using quadratures is more accurate than the solution without quadratures, and this result will be generally expected.

We verify the implementation of the 4^{th} order low-storage Runge-Kutta time integrator using the ordinary differential equation $\frac{\partial \phi}{\partial t} = \phi$ on $\Omega \in [-100, 0]$ integrating up to a time $t = 1$, with solution $\phi = \phi_0 e^t$. Here we choose $\phi_0 = 1$, such that the spatial discretization does not affect the error. The results are given in Table [6.5], from which we note that our implementation converges at the optimal rate for this

low-storage Runge-Kutta scheme. While this test corresponds to exponential biological growth, as above, the error values are of course not those that would occur for (6.7)-(6.9).

Table 6.4: Spatial convergence of 1D DG solver used to evaluate the source terms using N_h elements. The L^2 norm (see §6.3.1) of the error, $e = \phi_h - \phi$, is smaller for the quadrature-based scheme, but the order of convergence is the same for both. Order of convergence is computed in a standard way, e.g. Chapra and Canale (2006).

		$N_h = 10$		$N_h = 20$	
	Degree	$\|e\|_2$	Order	$\|e\|_2$	Order
Quadrature-based	p=1	5.550e-003	1.9	1.409e-003	2.0
	p=2	5.901e-004	2.9	7.491e-005	3.0
	p=3	4.690e-005	3.9	3.019e-006	4.0
	p=4	2.976e-006	4.9	9.673e-008	4.9
Quadrature-free	p=1	1.340e-002	1.9	3.435e-003	2.0
	p=2	1.068e-003	2.9	1.332e-004	3.0
	p=3	7.464e-005	3.9	4.724e-006	4.0
	p=4	4.282e-006	5.0	1.274e-007	5.1

Table 6.5: Temporal convergence of 1D DG solver using N_t time steps (different values of N_t given only to show that the order does not vary with N_t, but the absolute error of course changes). Order is computed using Chapra and Canale (2006).

Integration	$N_t = 16$		$N_t = 32$		$N_t = 64$	
Scheme	$\|e\|_2$	Order	$\|e\|_2$	Order	$\|e\|_2$	Order
RK4	5.683e-006	3.9	3.646e-007	4.0	2.308e-008	4.0

Perturbations of steady-states: The purpose of these studies is to examine how the biological dynamics behave as a result of perturbations away from the steady-state. Perturbations will arise due to the forcing and dynamics, and due to numerical reasons in the more complicated tests in §6.4.2, and it is important to understand how these perturbations will affect the biological dynamics. All three different biological regimes were examined in these perturbation tests. We focus on the behavior of the one-dimensional dynamics for the time interval $t^* = [0, 250]$, because this corresponds to the residence time of the biology for the dynamics in §6.4.2. We initialize all tests

using a perturbed or unperturbed exact steady state, which can be found by setting $\frac{\partial \Phi^*}{\partial t} + \nabla \cdot (\mathbf{u}^* \Phi^*) - \nabla \cdot \frac{1}{P_e} \nabla \Phi^* = 0$ in (6.7)-(6.9). The steady state solution is perturbed by setting $\Phi^*_{(P,Z)perturb} = (1+\gamma)\Phi^*_{(P,Z)steady}$, where γ is some constant, and using (6.10). Where required, we impose $\phi^*_Z + \phi^*_P < 1$, by setting $\Phi^*_{(P,Z)perturb} = \frac{\Phi^*_{(P,Z)steady}}{(\phi^*_Z + \phi^*_P)_{steady}}$. This initialization is done numerically by setting the value of the numerical solution equal to the calculated solution *at the nodal points.*

First we ensure that the exact steady state solution can be maintained, and then we initialize with a perturbation from the exact steady state, and the results are reported in Table [6.6]. For these runs we used 100, second-order accurate linear elements ($p = 1$), which roughly corresponds to the resolution at the inlet for $(g5, p1)$. We find that the steady solution can be maintained for all cases up to machine precision for the quadrature-free implementation when evaluating the error at the nodal points. This is because we initialized the numerical simulation using the exactly calculated steady state *at the nodal points.* Note that the quadrature-based scheme has a smaller difference than the quadrature-free version when evaluating the error at the quadrature points, except for the case with stable limit cycles in the euphotic zone (bio case 3). Because the quadrature version evaluates the source term at the quadrature points, and the interpolation of the solution onto the quadrature points is not exactly at the analytical steady-state, the source-terms are non-zero, and the solution evolves. If the source-terms were polynomials of lower degree than the basis in the z direction, this would not happen.

Finally, Table [6.6] gives a rough description of the dynamical properties of the equations. Here the norm of the initial difference, $\|D_i\|$ should be compared to the norm of the final difference at quadrature points for the quadrature-based treatment, and at the nodal points for the quadrature-free implementation. For the case with single stable points (bio case 1), the initial difference of the perturbed solution to the analytical steady state is greater than the final difference, which indicates that the solution is approaching the calculated steady-state value. For the case with stable limit cycles in the euphotic zone (bio case 3), the final difference is greater than the initial perturbed difference, showing the solution is logically not approaching the

steady state, but instead spiraling outward toward the stable limit cycles present at each depth. Additionally, plotting the solution (Fig. [6-5]) profile for the largest perturbation, we can see that the perturbed solution tends towards the steady state for the entire column for bio case 1, only for the top part of the water column for bio case 2, and nowhere for bio case 3. Thus, the parameter set with limit cycles in the euphotic zone (bio case 3) has the most structure in the vertical, and will require the most resolution to model accurately. Also, numerical perturbations will be most important for bio case 3 because the differences will grow away from the calculated steady state, as opposed to decaying.

Table 6.6: Difference between analytical steady state solution, and perturbed solution at $t^* = 250$. Here $D = \frac{\Phi - \Phi_h}{\int_\Omega 1 d\Omega} \cdot 100\%$ is the percent error per area in the domain. The column $\|D_i\|$ gives the initial difference, D_q indicates using quadratures, D_{qf} indicates quadrature-free, $\|.\|^{qp}$ indicates the error evaluated at quadrature points, $\|.\|^{nd}$ indicates the error evaluated at nodal points.

Stability	$\gamma \cdot 100\%$	$\|D_i\|_2^{qp}$	$\|D_i\|_2^{nd}$	$\|D_q\|_2^{qp}$	$\|D_{qf}\|_2^{qp}$	$\|D_q\|^{nd}$	$\|D_{qf}\|^{nd}$
Single	0	0.361	0.000	0.296	0.361	0.117	0.000
stable	0.05	0.361	0.014	0.296	0.361	0.117	0.002
points	0.50	0.377	0.126	0.295	0.363	0.122	0.015
	5.00	0.906	0.860	0.318	0.402	0.208	0.141
Stable limit	0	0.364	0.000	0.352	0.364	0.015	0.000
cycles at	0.05	0.366	0.017	0.353	0.365	0.019	0.011
bottom of	0.50	0.410	0.169	0.377	0.387	0.116	0.112
euphotic zone	5.00	1.360	1.29	1.01	0.974	0.945	0.915
Stable limit	0	0.109	0.000	0.112	0.109	0.038	0.000
cycles in whole	0.05	0.111	0.021	0.637	0.517	0.736	0.619
euphotic zone	0.50	0.234	0.205	1.40	1.20	1.51	1.36
	5.00	1.880	1.88	2.39	2.17	2.51	2.37

Vertical resolution: By varying the resolution of the problem, we found that a minimum of 25 degrees of freedom were necessary to roughly capture the vertical structure of the biological model dynamics at the final time. For the tests in §6.4.2, $(g1, p6)$ has approximately 21 degrees of freedom, indicating that it will be under-resolved.

High-order bases: We find that the quadrature-based treatment of the source

a)

b)

c)

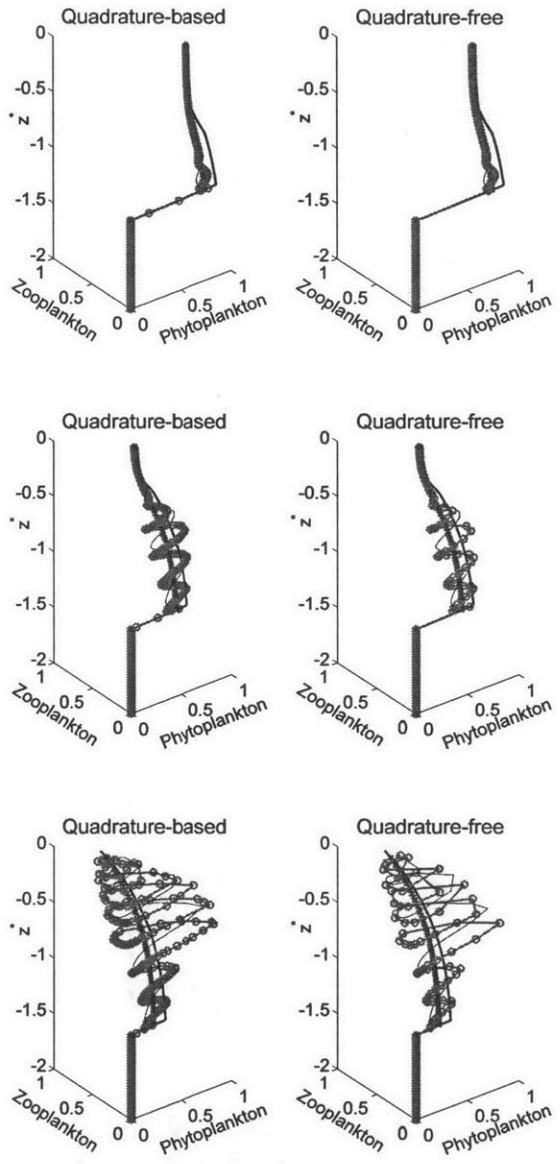


Figure 6-5: Solution profiles at all depths with $\gamma = 5\%$. Magenta crosses show the analytical steady-state solution, the thick black dashed lines show the initial condition, green circles show the profile at $t^* = 250$, and thin blue lines show the profile at $t^* = 125$. Plotted for biological dynamics with a) single stable points at all depths, b) stable limit cycles at bottom of euphotic zone, and c) stable limit cycles in entire euphotic zone. The quadrature-based solution is plotted at the quadrature points, whereas the quadrature-free solution is plotted at the nodal points.

a)

b)

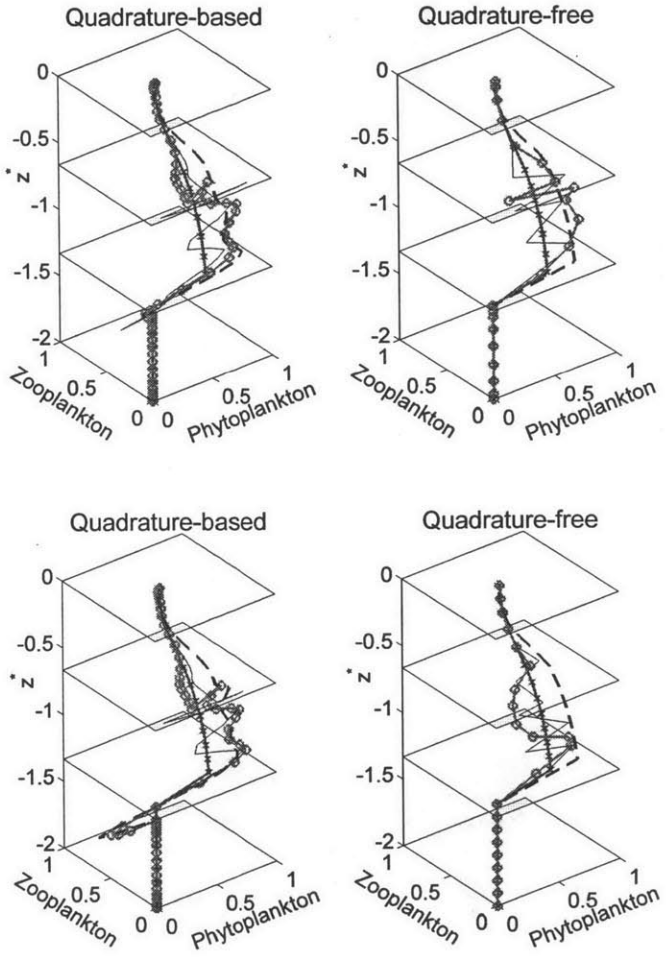


Figure 6-6: Solution profiles for all depths at $t^* = 500$ using a 15^{th} degree polynomial and 3 elements with $\gamma = 5\%$ for dynamics with stable limit cycles at the bottom of the euphotic zone. As in Fig. [6-5], the magenta crosses show the analytical steady-state solution, thick black dashed lines show the initial condition, green circles show the profile at $t^* = 250$, and thin blue lines give the profile at $t^* = 125$. The solution is plotted at the quadrature points for the quadrature version, and at the nodal points for the quadrature-free (i.e. where the source terms are evaluated). a) Uses well-behaved (Gauss-Lobatto) nodal points, b) uses uniform nodal points.

terms results in large jumps of the solution between elements. This is illustrated in Fig. [6-6] after 500 time units of integration using a 15^{th} degree polynomial and 3 elements. The problem is amplified when using a uniform nodal spacing, due to a larger interpolation error. Also, increasing the number of quadrature points used for integration did not solve this problem. The problem originates from the discontinuous jump in the solution, causing oscillations known as Gibbs phenomena. Note that both simulations are initialized in the same manner, but the Gibbs oscillations can only be "seen" when evaluating the initial condition at points other than the nodes. The quadrature-based integration, then, "sees" these oscillations because the source terms are evaluated at the quadrature points. Using the quadrature-free approach for this one-dimensional problem essentially decouples the vertical nodes, so numerically, the quadrature-free version does not "see" the oscillations. The Gibbs oscillations would have occurred in the quadrature-free scheme if the initialization was done at the quadrature-points instead. This example illustrates one of the drawbacks of using increasingly higher order schemes, that is, without special treatment, large oscillations occur for non-smooth functions. Using lower-order but on a finer discretization (more elements) can be a better strategy if special treatment is not used.

In this section we showed that, with our implementation, both the quadrature-based and quadrature-free treatment of the source-terms give accurate, convergent results (see Table [6.4]), although the absolute error of the quadrature-based implementation is smaller than the quadrature-free implementation. Then we showed that the analytical steady state solution could be maintained, and illustrated the dynamical behaviour of three different biological parameter sets through perturbations of the steady state solutions. With the vertical resolution tests we found a minimum of 25 degrees of freedom necessary to roughly capture the vertical solution features of our particular setup. Finally, we showed that oscillations can occur solely due to numerics for a high-order discretization. While the quadrature-based algorithm was shown to be more accurate, the oscillations at element interfaces and the added numerical cost need to be considered. The additional accuracy may be warranted when a bifurcation of the solution could occur, or when the solution is under-resolved. As a

whole, a key result is that, for any numerical scheme, careful numerical studies should be performed in one-dimension to understand the potential errors arising from the nonlinear source term discretization before proceeding with advective models.

6.4.2 Full NPZ equations

In this section we explore the case where the advection and biological source terms are approximately balanced. We examine effects of low-order and high-order temporal discretizations in §6.4.3. In §6.4.4, we illustrate the difference between using a quadrature-based and quadrature-free scheme to discretize the nonlinear biological source terms. Finally, in §6.4.5, we study effects of spatial resolution, through both grid resolution and polynomial degree.

We still study the three biological parameter sets: single stable points, stable limit cycles at the bottom of the euphotic zone, and stable limit cycles for the entire euphotic zone, as given in Table [6.2]. Since the timescale of biology varies in depth, the advection and biological source terms can only be balanced for one depth. While this results in many choices of approximately balanced parameter sets, we focus on one where $\bar{\tau}_a = 12.5$ [days]. For these tests, the inlet is specified as the steady state solution with a smoothed discontinuity. The discontinuity is smoothed by fitting it with a cubic polynomial which can be resolved on $(g1, p6)$. The fit is biased such that 3/4 of the polynomial is below the euphotic zone. For the outlet boundary we use $\frac{\partial \Phi}{\partial \hat{\mathbf{n}}} = 0$. Also these results are compared to a $(g5, p1)$ simulation, which is taken as the true solution.

We use a potential flow–field for this case. A potential flow-field is calculated by solving (6.3) using HDG as described in §2. Once ψ is found, we take $\mathbf{u} = [\psi_z, -\psi_x]$. The value of ψ is specified on all boundaries. The top and bottom boundaries are specified as constants $\psi = \psi_{top}$ and $\psi = \psi_{bot}$. The inlet and outlet stream functions are specified to vary linearly in z^*, i.e. $\psi = \psi_{top} + z^*(\psi_{bot} - \psi_{top})/D^*$.

The final solution fields for the three different regimes of biological dynamics (from Table [6.2]) and using quadrature-based source terms for $(g5, p1)$ (the reference solution) is plotted in Fig. [6-7]. The results show that idealized strait bathymetry

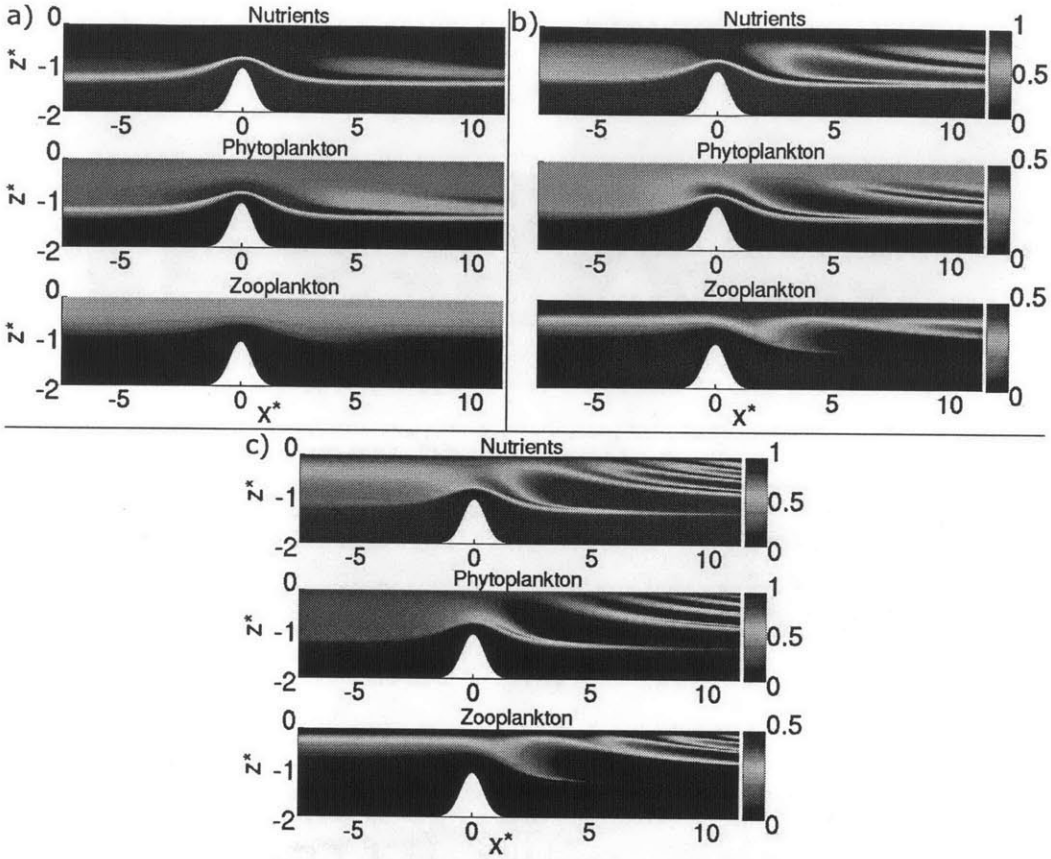


Figure 6-7: Biological dynamics at $t^* = 20$ (with $\bar{\tau}_a = 12.5$[days]) using $(g5, p1)$. Biolocial dynamics with a) single stable points, b) stable limit cycles for depths $z^* = 0.4 - 0.9$, and c) stable limit cycles in whole euphotic zone. This is the reference solution against which all other solutions are compared.

effectively perturbs the biology away from the inlet conditions. The case with single stable points (bio case 1) adjusts back to the stable equilibrium, whereas the two cases with limit cycles show complex structures in the vertical. In all cases, a Phytoplankton bloom over the bump is observed.

To qualitatively evaluate the effect of refining the grid or polynomial degree, we show the solution field for Phytoplankton for $(g3, p1)$, $(g3, p2)$, $(g4, p1)$, and $(g4, p2)$ in Fig. [6-8], and these discretizations have 16,800, 33,600, 67,200, and 134,400 DOFs respectively. This figure shows that the solution is converging with increased resolution. More quantitative comparisons are completed next.

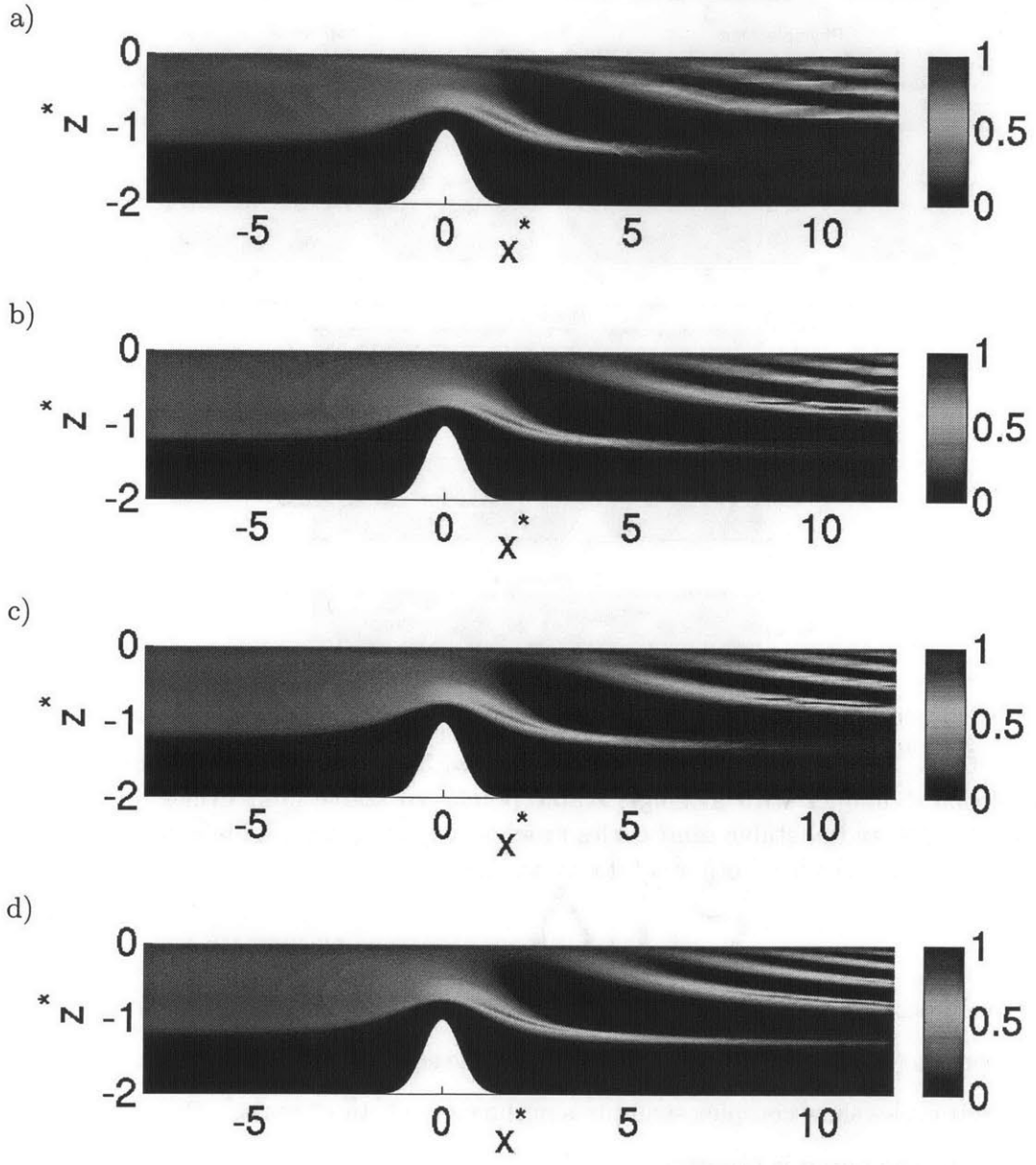


Figure 6-8: Phytoplankton fields at time $t^* = 20$ (with $\bar{\tau}_a = 12.5$[days]), as computed using four different spatial resolutions and order of the FE scheme: a) $(g3, p1)$, (16,800 DOFs) b) $(g3, p2)$ (33,600 DOFs), c) $(g4, p1)$ (67,200 DOFs), and d) $(g4, p2)$ (134,400 DOFs). All fields are for biological dynamics with stable limit cycles in the euphotic zone (bio case 3 in Table [6.2]).

6.4.3 Comparing low-order and high-order temporal discretizations

We compare the solutions using 4^{th} order Runge-Kutta, 2^{nd} order Runge-Kutta, and 1^{st} order Explicit Euler on $(g2, p4)$ for the biology with stable limit cycles in the euphotic zone (bio case 3). The differences of the lower order schemes compared to 4^{th} order Runge-Kutta at $t^* = 40$ is plotted in Fig. [6-9] for the Phytoplankton field. These difference fields are plotted at a point in time, and will measure the effect of phase and path-accumulated errors. Note, the timestep size for the 1^{st} order scheme is half of the 2^{nd} order scheme, such that the cost of the two are the same. For this test case we used periodic boundary conditions. From the figure, we note that the major differences occur within the euphotic zone. The stable explicit timestep for the 2^{nd} order scheme is set by the Courant condition for the advection discretization, and since the largest velocity occurs in the smallest element for this discretization, the timestep size is approximately four orders of magnitude smaller than the biological timescale. Therefore, it is expected that temporal errors in the source-term should be small even for the low order scheme. Nonetheless, we still observe differences between the 1^{st}, 2^{nd} and 4^{th} order schemes. We found that the difference at $t^* = 40$ is approximately two orders of magnitude larger than at $t^* = 20$, which indicates the errors are growing quickly. The 1^{st} order scheme will have a bias which is more affected by path-accumulated errors, and for it we observe a maximum error of $\mathcal{O}(1)$ at $t^* = 40$. This suggests that a low-order time discretization may result in significant errors when long integration times or fast biological timescales are involved. For example, the latter occurs in coastal applications. As another example, for stiff biogeochemical source terms, Burchard et al. (2005) found that even 4^{th} Runge-Kutta integration is insufficient to maintain the non-negativity of the biological components. They suggest that positivity preserving Patankar-Runge-Kutta schemes should be used to obtain a non-negative, conservative solution.

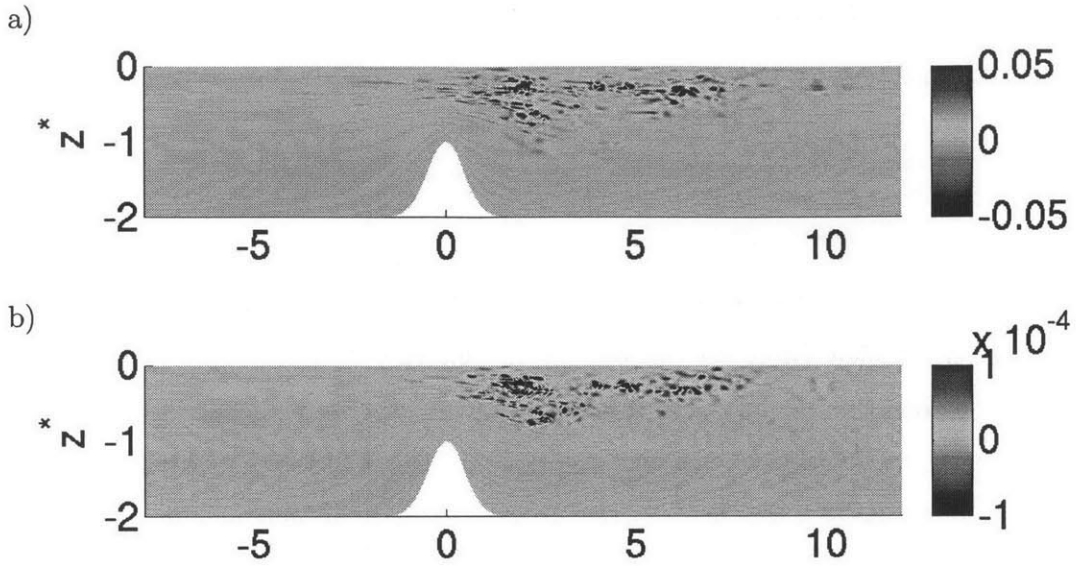


Figure 6-9: Temporal discretization differences for Phytoplankton field with stable limit cycles in euphotic zone at $t^* = 40$ using periodic boundary conditions and on spatial grid $(g4, p2)$. a) "1^{st} order Euler" minus "4^{th} order Runge-Kutta", and b) "2^{nd} order Runge-Kutta" minus "4^{th} order Runge-Kutta".

6.4.4 Comparing quadrature-based and quadrature-free source terms

In §6.4.1 we found that the greatest difference between the quadrature and quadrature-free treatment of the source-terms occurred for the biological parameter set with stable limit cycles in the euphotic zone (i.e. bio case 3). Here we examine this case for full ADR dynamics using the $(g1, p6)$ discretization. Note that we obtained the same results and conclusions with the $(g4, p1)$ and $(g5, p2)$ discretizations (not shown). Plotting the difference (quadrature-free minus quadrature-based) of the solution in Fig. [6-10] for $(g1, p6)$ we see that the largest differences occur near the outlet of the domain where the mesh solution is under-resolved. The quadrature-based solution is more accurate in the under-resolved region because the source-term integral is more accurately evaluated, and this was verified by comparing the errors of the two schemes. However, where the solution is sufficiently resolved, the quadrature-free and quadrature-based treatments of the source terms have similar accuracy, that

is, they differ by approximately 0.1%. From the one-dimensional studies, we found that the quadrature-free algorithm was less oscillatory at element interfaces than the quadrature-based algorithm, and we observed the same effect in these 2D simulations for $p > 7$ on $g1$, although the difference between quadrature-free and quadrature-based was less drastic. The largest differences between the quadrature-based and quadrature-free schemes did occur at element boundaries, and the quadrature-based algorithm was more accurate when under-resolved.

Using equation (6.10), we verify the conservation of the scheme. The results for the quadrature-free and quadrature-based source terms were similar up to floating point precision. Also, we find that the conservation error is dominated by the flow field divergence error. Therefore, the conservation properties of the source term discretization does not affect the choice between quadrature-free and quadrature-based algorithms.

Because the quadrature-free and quadrature-based algorithms had similar accuracy in well-resolved regions, we recommend using the quadrature-free treatment in these regions because of the improved efficiency. However, when the solution is poorly resolved, the quadrature-based treatment of the source terms is more accurate. Now, depending on the total solution cost of a particular numerical scheme, a finer resolution quadrature-free scheme may be more efficient for the same accuracy than a quadrature-based scheme.

6.4.5 Comparing low-order and high-order spatial discretizations

Fig. [6-11] shows the differences between the reference solution and the solutions using other grids and polynomial degrees in Fig. [6-11] for Zooplankton. We see that the $(g2, p5)$ simulation has the smallest differences, and is therefore the most accurate. This is a key result since it indicates that when the solution is resolved, for the same cost/efficiency, a higher order scheme on a coarser grid performs better than a lower-order scheme on a finer grid. Results for the biological dynamics with

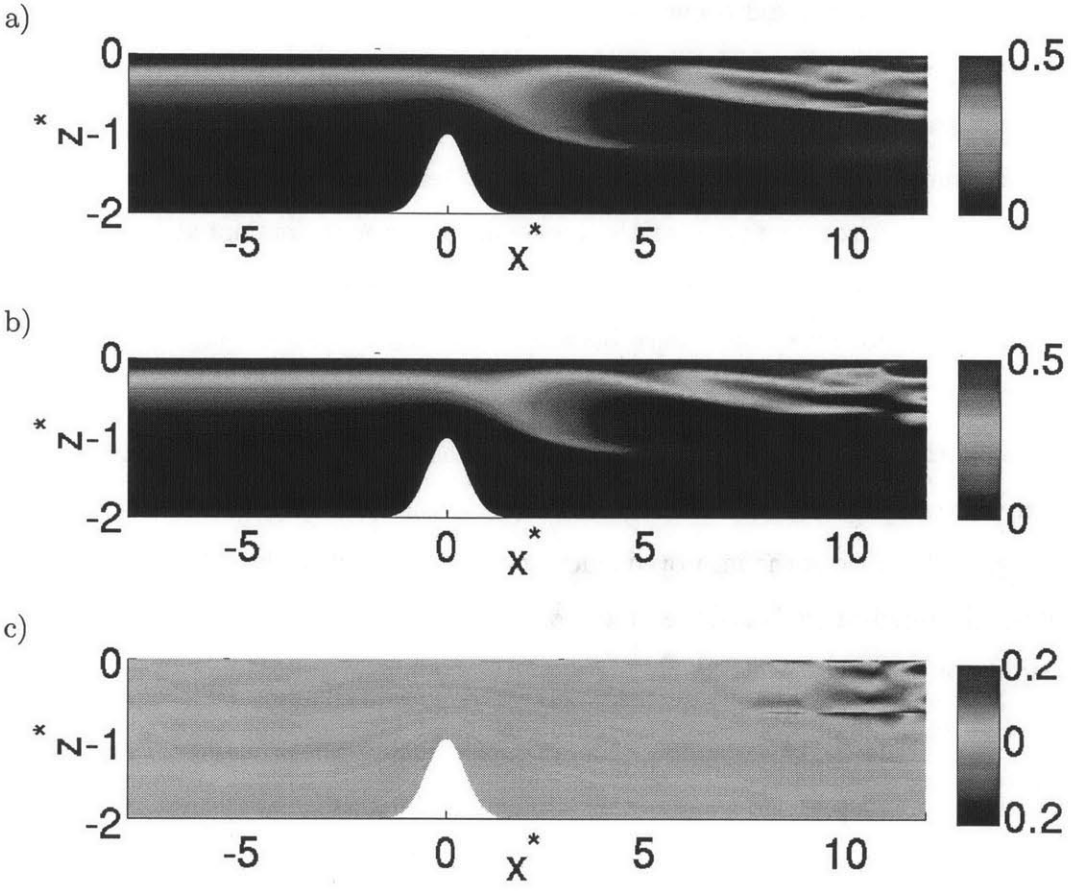


Figure 6-10: Zooplankton fields at $t^* = 20$ computed using $(g1, p6)$ and a) quadrature-based source terms, b) quadrature-free source terms. c) The difference between the quadrature-free and quadrature-based source-term simulations. The biological dynamics used has stable limit cycles within the euphotic zone (bio case 3).

limit cycles at the bottom of the euphotic zone (bio case 2) are similar, but for the biology with limit cycles in the entire euphotic zone (bio case 3), both high and low-order schemes are under-resolved for $x^* > 7$, especially for the $(g1, p6)$ scheme. The differences between a fine grid solution $(g5, p1)$ and the low-order and high-order schemes are plotted in Fig. [6-12] for Zooplankton. From Fig. [6-12], we note that the errors in the low-order scheme are more localized in the $x^* > 7$ region. However, the differences for $(g4, p1)$ and $(g2, p5)$ are similar in the $x^* > 7$ region. The $(g1, p6)$ scheme has the least localized and largest magnitude errors in the $x^* > 7$ region. However, as plotted in Fig. [6-13] where the solution is smooth and the biology has less structure in the vertical, both the high-order schemes are more accurate than the low order scheme. Particularly, note the solution near the surface for $x^* < 7$ in Fig. [6-13].

We examine the error characteristics of these fields more closely by considering the truncated Taylor expansions of the true solution. By the mean value theorem, the truncation error for $(g4, p1)$ is $\frac{h^2}{2!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^2 \phi(\mathbf{x}_\eta)$ for some unknown point $\mathbf{x}_\eta$, and for $(g2, p5)$ and $(g1, p6)$ these terms are $\frac{h^7}{6!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^6 \phi(\mathbf{x}_\zeta)$ and $\frac{h^7}{7!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^7 \phi(\mathbf{x}_\xi)$ for unknown points $\mathbf{x}_\zeta$ and $\mathbf{x}_\xi$, where h is the characteristic discretization length of an element. Now, we can examine the approximate truncation error by running simulations $(g4, p2)$, $(g2, p6)$, and $(g1, p7)$, and evaluating the highest order non-zero derivatives of the approximate solution ϕ_h. To evaluate the derivatives, we interpolate the solution onto an orthogonal modal polynomial basis, that is we find the coefficients a_{ij} such that $\phi_h = \sum_i \phi_i \theta_i = \sum_{ij} a_{ij} P_{ij}$, where P_{ij} is a modal orthogonal polynomial with maximum degree of i on x and j on y, for a maximum total degree of $i + j$. The derivatives then evaluate as $\frac{1}{2!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^2 \phi_h(\mathbf{x}_\eta) = \sum_{i+j=2} a_{ij}$, $\frac{1}{6!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^6 \phi_h(\mathbf{x}_\xi) = \sum_{i+j=6} a_{ij}$, and $\frac{1}{7!}\left(\frac{\partial}{\partial x}+\frac{\partial}{\partial y}\right)^7 \phi_h(\mathbf{x}_\xi) = \sum_{i+j=7} a_{ij}$, that is, we simply need to sum the coefficients of the modal orthogonal polynomial basis which correspond to terms with total degree of 2, 6, and 7, respectively. Since the coefficients of the polynomials are evaluated on the reference element, $h \approx 1$ will be the same for all elements. Also, while this approach gives an estimate of the leading order truncated term, it does not give an exact value. In our case, we are not interested in

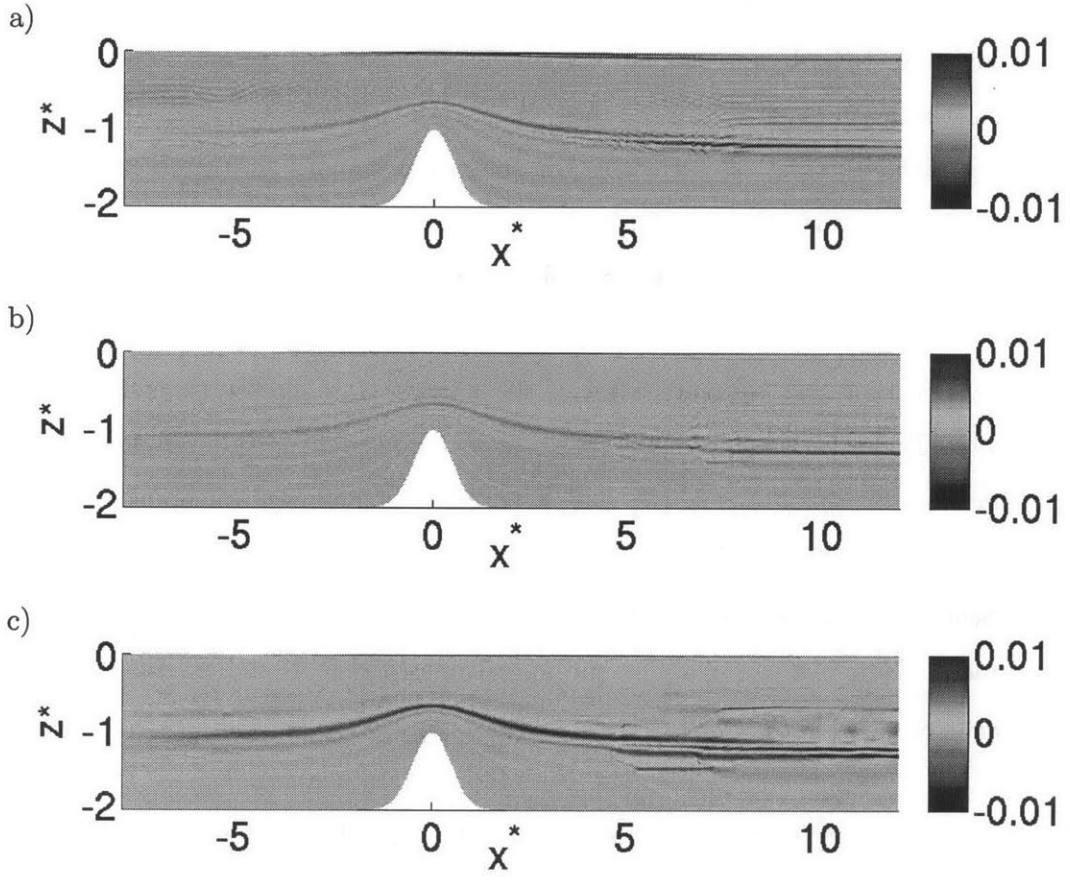


Figure 6-11: Difference between Zooplankton fields at $t^* = 20$ (with $\bar{\tau}_a = 12.5$[days]) computed using $(g5, p1)$ and a) $(g4, p1)$, b) $(g2, p5)$, and c) $(g1, p6)$. This shows the locations of the largest numerical errors for the high-order and low-order schemes. The biological dynamics used have single stable points at all depths (bio case 1).

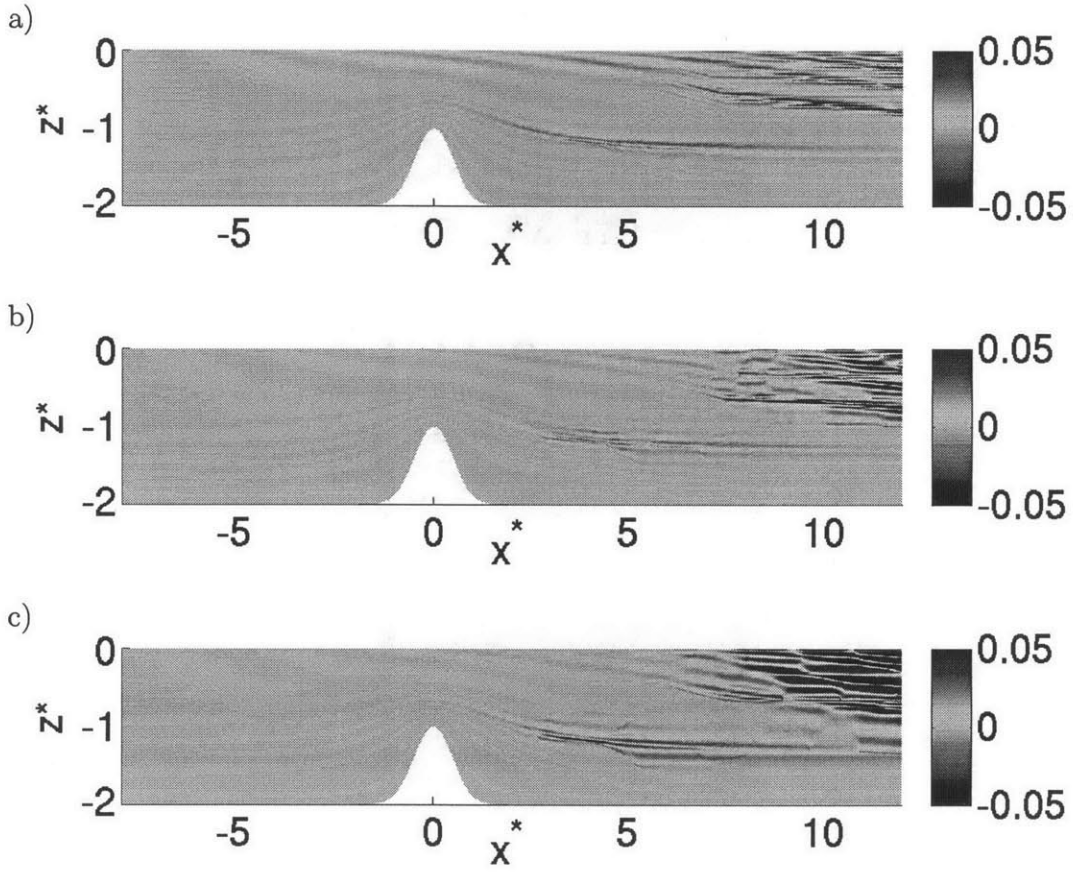


Figure 6-12: As Fig. [6-11], but for the biological dynamics with stable limit cycles within the euphotic zone (bio case 3).

a)

b)

c)

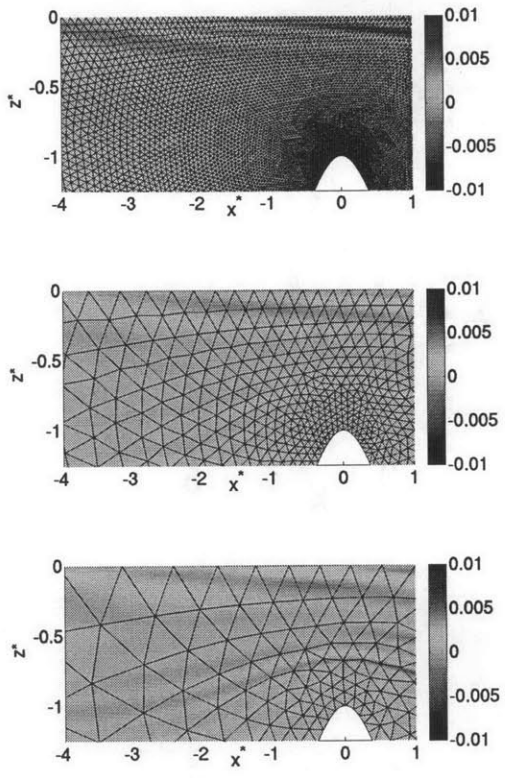


Figure 6-13: As Fig. [6-12], but zoomed in the region above the bathymetry. The difference between Zooplankton fields using $(g5, p1)$ and a) $(g4, p1)$, b) $(g2, p5)$, and c) $(g1, p6)$.

a rigorous error estimator, but instead we only require an estimate of the error to aid the discussion.

Our approach is similar to that followed by Mavriplis (1989), where Legendre polynomials were used instead. The author proposed a smoothness estimator, where the coefficients a_{ij} are fit to the exponentially decaying function $a(i+j) = Ce^{\sigma(i+j)}$. There the author claims that $\sigma < -1$ indicates good resolution or smooth functions, and $\sigma > -1$ indicates poor resolution or non-smooth functions. The adaptive strategy used was to increase the polynomial degree for elements with $\sigma < -1$, and to refine the mesh for elements with $\sigma > -1$, if the error level in that element was insufficient. We evaluate this smoothness indicator σ on $(g1, p7)$ by doing a least squares fit of the coefficients to $Ce^{\sigma(i+j)}$. In regions where the magnitude of the solution is close to 0, that is below the euphotic zone for the Zooplankton field, a_{ij} $\forall i, j$ will be small, and the smoothness indicator σ will not be accurate. The approximate size of the truncated derivative terms along with the smoothness indicator are plotted in Fig. [6-14]. Only the smoothness indicators calculated on $(g2, p5)$ and $(g1, p7)$ are plotted since the accuracy of the smoothness indicator improves with the number of terms in the polynomial expansion, and is not accurately represented on $(g4, p2)$ (Mavriplis, 1989).

From Fig. [6-14], we note that the largest differences in Fig. [6-12] correspond to the regions with the largest approximate truncations errors in Fig. [6-14]. Also, in the region $x^* > 7$ where the low-order solution is more accurate than $(g1, p6)$, we have $\sigma > -1$, which suggests that refining the elements instead of the order of accuracy is more appropriate. After one level of refinement on $(g2, p5)$, we see that the smoothness indicator shows a smaller region of non-smooth elements. This illustrates that the smoothness is defined in terms of the numerical discretization, and is not solely a function of the solution field. Also note in the region where the high-order solution is more accurate (see Fig. [6-13]), the approximate derivative of the truncation term is small in both fields, and $\sigma < -1$, suggesting a higher degree polynomial basis is more appropriate in this region. This shows that our mesh is not optimized in terms of the solution field, and highlights the importance of

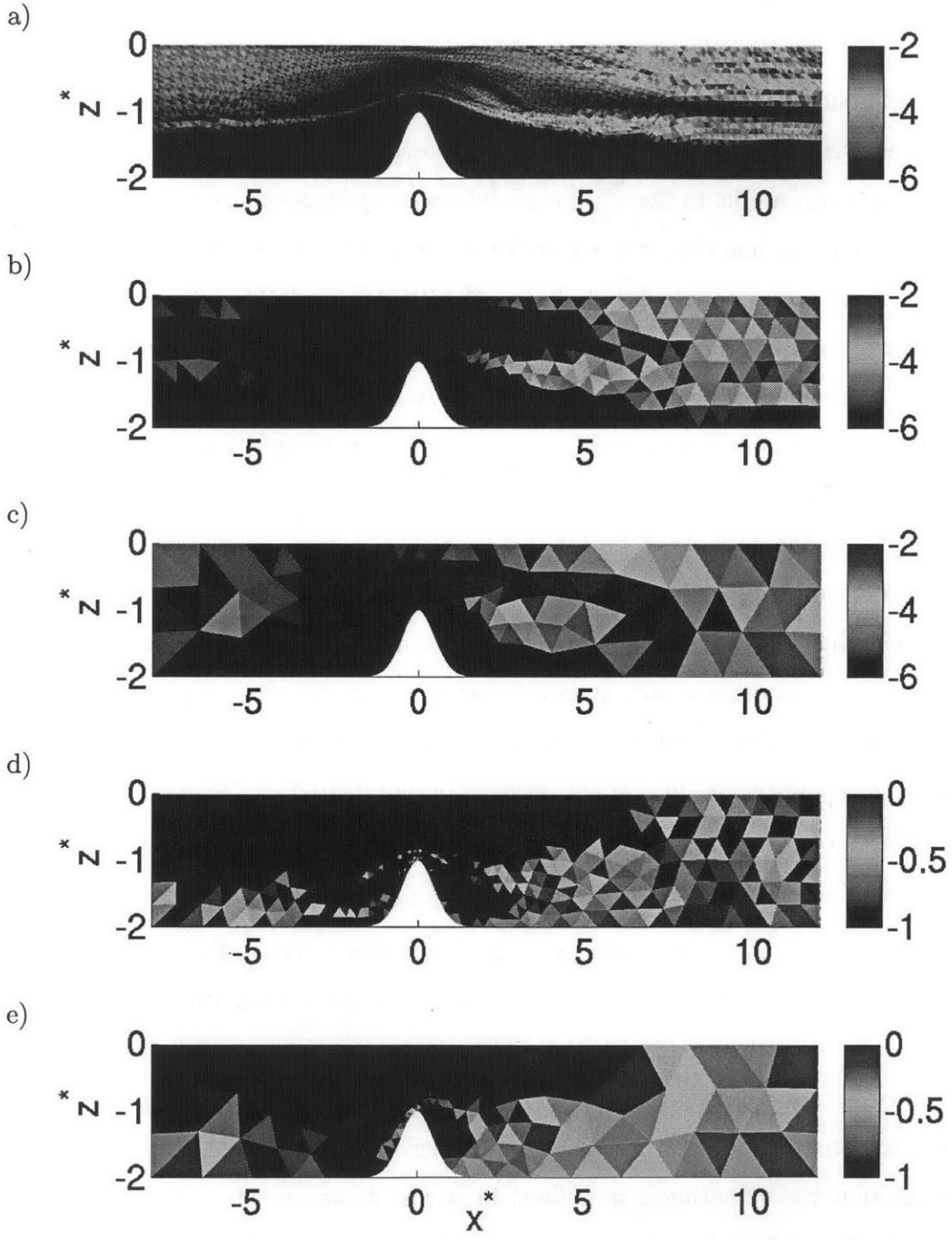


Figure 6-14: Approximate truncation errors for Zooplankton fields at $t^* = 20$ (with $\bar{\tau}_a = 12.5$[days]). Calculated on a) $(g4, p2)$ using $log_{10}(\sum_{i+j=2} a_{ij})$, on b) $g2, p6$ using $log_{10}(\sum_{i+j=6} a_{ij})$, and on c) $g1, p7$ using $log_{10}(\sum_{i+j=7} a_{ij})$. d-e) Smoothness indicator σ calculated on d) $(g2, p6)$ and e) $(g1, p7)$. The biological dynamics used has stable limit cycles within the euphotic zone (bio case 3).

using both mesh refinement and polynomial basis adaptation to generate an optimal discretization for complex biological ocean dynamics. Also, this shows that whether a coarsely discretized higher order scheme is better than a finely discretized lower order scheme depends on the smoothness of the solution and can vary spatially across the solution. The benefit from a higher-order solution is as follows: when the solution is smooth, increasing the polynomial degree causes the error to decrease exponentially, whereas the error would only decrease algebraically if decreasing the element size. The cost of increasing the polynomial degree also scales algebraically, and because of this, a higher-order scheme performs better for smooth or well-resolved fields. Using our implementation, the $(g1, p6)$ simulation took approximately 0.34 of the time taken by the $(g4, p1)$ simulation. We also ran $(g3, p1)$, which was approximately three times more efficient than $(g1, p6)$, but this solution (not shown) was less than 1% accurate for the majority of the domain. When the solution is not resolved (i.e. not smooth for the grid resolution or polynomial degree chosen), higher-order schemes will lead to Gibbs oscillations and filtering is required (see §3.5), while lower-order schemes may "look good" but will be very dissipative. When the solution is resolved (i.e. smooth enough for the grid resolution or polynomial degree chosen), higher-order discretizations perform better than lower-order ones: they are more accurate and less dissipative for the same cost.

Finally, we note that the approximate truncation error and smoothness metrics were different for the different biological components. Therefore, the optimal discretization for one component is not the same as the optimal discretization for another component. Ueckermann (2009) proposed a scheme that uses a different order basis function for different biological components, but also cautions that an incurred interpolation cost needs to be considered for adaptation strategies.

6.4.6 Evolution of Biological Patch

In this section we demonstrate how biological activity can enhance the differences between low order and high order discretizations beyond the effect of numerical dissipation alone. For this example, we modify bio case 1 (single stable points at all

depths) from §6.4.2 by introducing a vertical column, or "patch", of biology that uses the parameters from bio case 2 (stable limit cycles at depths $z^* = 0.4 - 0.9$ and single stable points elsewhere). This is easily done in the dimensional form of the equations by increasing the value of $\mathcal{N}_T$ locally in the patch. Such situations occur frequently in nature, e.g. an eddy or front upwelling additional nutrients locally towards the surface. The initial condition and boundary condition is the same as in bio case 1 (the steady state solution with smoothed discontinuity), except inside the patch where the initial conditions for bio case 2 are used instead, that is:

$$\phi^*_{(\text{patch})} = \phi^*_{(\text{bio case 1})} + \left[\phi^*_{(\text{bio case 2})} - \phi^*_{(\text{bio case 1})}\right] \cdot e^{-\frac{(x^*+6.4)^4}{2\cdot(8^4)}}, \qquad (6.11)$$

where $\phi^*_{(\text{patch})}$ is the initial condition used for this example, $\phi^*_{(\text{bio case 1})}$ is the steady state with smoothed discontinuity for bio case 1, and $\phi^*_{(\text{bio case 2})}$ is the steady state with smoothed discontinuity for bio case 2. Note, for this example, we non-dimensionalize $\phi^* = \frac{\phi}{\mathcal{N}_{T,\text{bio case 1}}}$ by the total biomass for bio case 1. In addition we superimpose a periodic velocity onto the mean velocity,

$$\mathbf{u}^* = \mathbf{u}^*_{\text{mean}}[1 + 5 \cdot \text{sign}\{\cos(0.16\pi t^*)\}], \qquad (6.12)$$

where $\mathbf{u}^*$ is now the velocity used for this example, and $\mathbf{u}^*_{\text{mean}}$ is the potential flow-field.The superimposed velocity increases the distance traveled, as well as the number of time integration steps (due to the CFL condition), and therefore has the effect of increasing the numerical dissipation.

Fig. [6-15] plots the phytoplankton fields and total biomass for $(g2, p5)$ and $(g4, p1)$ around the patch, as well as the difference of the solution, $(g2, p5)$ minus $(g4, p1)$, at $t^* = 14.4$. We do not use the $(g5, p1)$ solution (as was done in §6.4.2) for the difference plots here because we found that even $(g5, p1)$ is more dissipated than $(g2, p5)$, and therefore $(g2, p5)$ is more accurate inside the biological patch where our calculations are performed. These results show that the total biomass peak is not maintained by the low order scheme, $(g4, p1)$, and the details in the phytoplankton fields are also dissipated. Since these simulations do not contain physical diffusion,

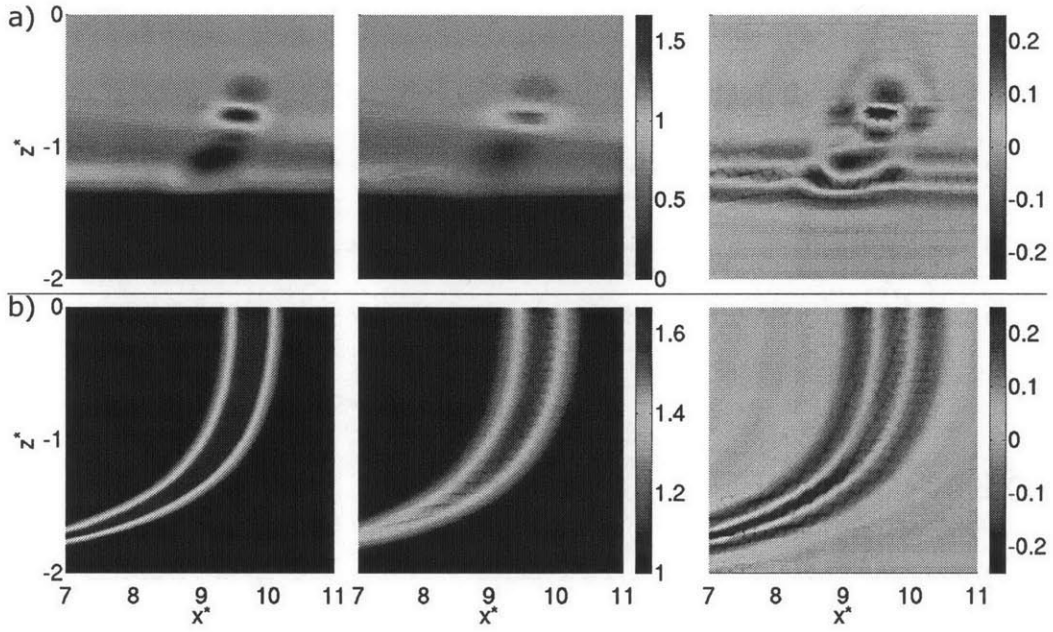


Figure 6-15: Detail around the biological patch with stable limit cycles at the bottom of the euphotic zone at time $t^* = 14.4$ for a) the phytoplankton fields and b) the total biomass. The solution for $(g2, p5)$ is plotted on the left, $(g4, p1)$ in the middle, and the difference between the solutions, $[(g2, p5) - (g4, p1)]$, is plotted on the right. This shows that $(g2, p5)$ correctly maintains the full peak of the biological patch, while $(g4, p1)$ does not, leading to large differences in the phytoplankton fields.

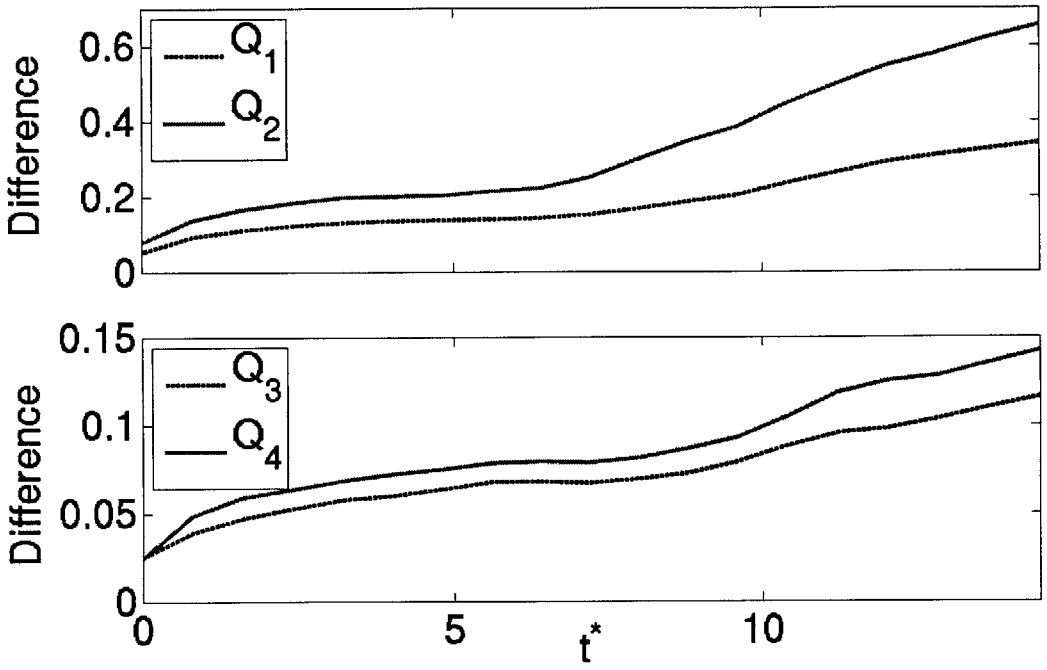


Figure 6-16: The relative normed difference between the total biomass of the two solutions, (Q_1, (6.13)), the sum of relative normed differences between the biological components (Q_2 (6.14)), the relative normed difference in production (Q_3 (6.15)), and the relative normed difference in grazing (Q_4 (6.16)) over time from $t^* = 0$ to $t^* = 14.4$. This shows that the difference in biological components is amplified beyond the effect of numerical dissipation due to differences in the source terms such as the production and grazing.

any diffusion is due to the numerical scheme, and therefore the $(g2, p5)$ solution is more accurate than $(g4, p1)$ because it does maintain the total biomass peak. Apart from the effects of the periodic velocity, the solution inside the patch should resemble that of Fig. [6-7]b), and $(g2, p5)$ resembles this solution more closely than $(g4, p1)$.

While some of the differences between the $(g2, p5)$ and $(g4, p1)$ simulations can be accredited solely to the numerical dissipation, the error due to numerical dissipation is amplified by the change in biological activity. To illustrate this point, we show in Fig. [6-16]: the relative normed difference between the total biomass of the two solutions, (Q_1, (6.13)), the sum of relative normed differences between the biological components (Q_2 (6.14)), the relative normed difference in production (Q_3 (6.15)), and the relative normed difference in grazing (Q_4 (6.16))

$$Q_1 = \frac{\|\{\phi_N^* + \phi_P^* + \phi_Z^*\}_{(g2,p5)} - \{\phi_N^* + \phi_P^* + \phi_Z^*\}_{(g1,p4)}\|^{\text{patch}}}{\|\{\phi_N^* + \phi_P^* + \phi_Z^* - 1\}\|^{\text{patch}}} \tag{6.13}$$

$$Q_2 = \frac{\sum_{I=(N,P,Z)} \|\phi_{I,(g2,p5)}^* - \phi_{I,(g1,p4)}^*\|^{\text{patch}}}{\|\{\phi_N^* + \phi_P^* + \phi_Z^* - 1\}_{(g4,p1)}\|^{\text{patch}}} \tag{6.14}$$

$$Q_3 = \frac{\left\|\left\{\mathcal{U}^* e^{z^*/h^*} \frac{\phi_P^* \phi_N^*}{\phi_N^* + k_s^*}\right\}_{g2p5} - \left\{\mathcal{U}^* e^{z^*/h^*} \frac{\phi_P^* \phi_N^*}{\phi_N^* + k_s^*}\right\}_{g4p1}\right\|^{\text{patch}}}{\left\|\left\{\mathcal{U}^* e^{z^*/h^*} \frac{\phi_P^* \phi_N^*}{\phi_N^* + k_s^*}\right\}_{g4p1}\right\|^{\text{patch}}} \tag{6.15}$$

$$Q_4 = \frac{\left\|\left\{a g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*})\right\}_{g2p5} - \left\{a g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*})\right\}_{g4p1}\right\|^{\text{patch}}}{\left\|\left\{a g_\nu^* \phi_Z^* (1 - e^{-\nu_{\text{bio}}^* \phi_P^*})\right\}_{g4p1}\right\|^{\text{patch}}}, \tag{6.16}$$

where $\|e\|^{\text{patch}} = \left(\int_{\text{patch}} e^2 dx^* dz^*\right)^{\frac{1}{2}}$ with the patch area is determined from $(g4, p1)$, and the quantity $\|\{\phi_N^* + \phi_P^* + \phi_Z^* - 1\}\|^{\text{patch}}$ gives the size of the difference between the base solution and the solution inside the patch since the base number of nutrients (non-dimensionalized to 1) is subtracted out.

Since our numerical scheme conserves the total biomass, the first quantity, Q_1, gives a quantitative estimate of the numerical dissipation error only. Note, the initial differences between the two solutions are due to interpolation errors, since the polynomial representation and number of degrees of freedom are not the same for the two simulations. The second quantity, Q_2, should be the same as Q_1 if the only difference is due to numerical dissipation. However, from Fig. [6-16], we note that $Q_2 > Q_1$,

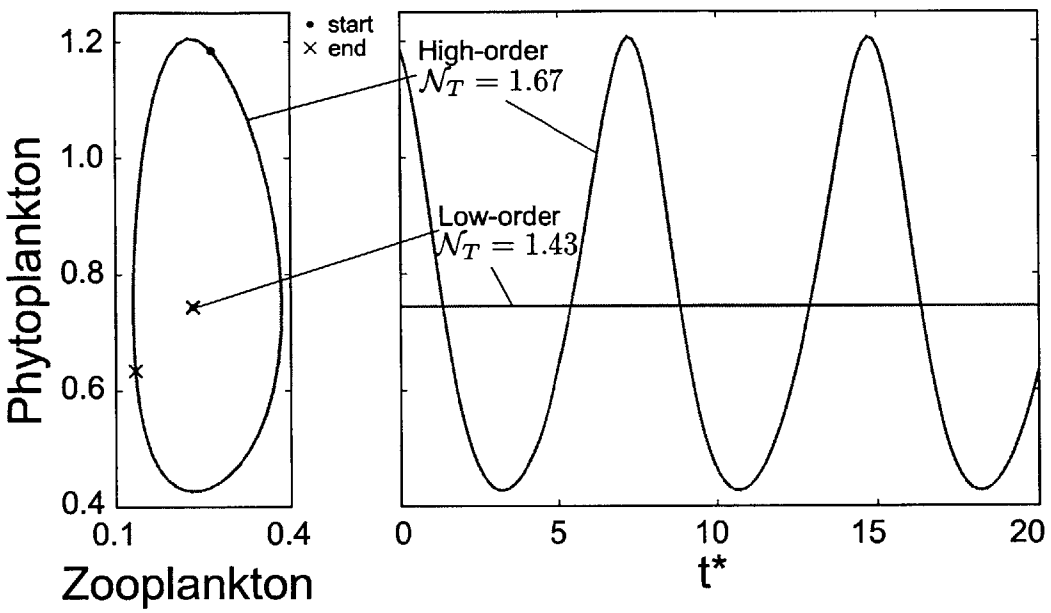


Figure 6-17: Long term dynamical behaviour of NPZ system for high (green) and low (red) order schemes at a depth of -0.667. The phytoplankton versus zooplankton phase diagram (left) and phytoplankton versus time (right) plots show that the high-order scheme retains a stable limit cycle, while the low-order scheme collapses to a single stable point.

which means the differences in dissipation is amplified by nonlinearities and the biology. This is explained by the differences in biological terms in the two simulations, for example in the production and grazing terms, Q_3 and Q_4. Also, note that these differences are growing over time, and for longer integration periods, the differences will be even greater. We can examine the long-term dynamics by integrating a point in the patch until steady behaviour is observed (Fig. [6-17]). From Fig. [6-17], we see that the dynamics of the high-order and low-order simulations are different. This is expected because the dynamical behaviour of this NPZ model is affected by the total biomass in the system. This shows that the high-order scheme can more faithfully reproduce the dynamics described by the mathametical model and its parameters.

The example shows that the numerical dissipation due to a lower-order numerical scheme can be amplified by the biological reaction terms. This is significant since for accurate biological ocean science through numerical simulations, it is important to maintain the amplitudes of biological patches. This is particularly true for biology

with multiple attractors, where relatively small perturbations can lead to vastly different dynamics. The conclusion is that for the same cost, higher-order schemes on coarser grids are more accurate than lower-order schemes on finer grids.

6.5 Conclusions

We completed a set of computational studies for the modeling of multi-scale biogeochemical dynamics in coastal ocean regions with complex bathymetric features, utilizing recent advances in computational fluid dynamics. Specifically, we compared low to high order discretization schemes, both in time and space, employing standard and hybrid discontinuous Galerkin Finite Element Methods, on both straight and curved elements. We studied the effects of a varied set of numerical properties including: quadrature-free and quadrature-based discretizations of the source terms; order of the spatial discretizations of advection and diffusion operators; order of the temporal discretization in explicit schemes; and, resolution of the spatial mesh, with and without our new curved elements. We verified the convergence of our numerical schemes for both the biology and flow fields, validated the codes on analytical solutions, and completed a rigorous truncation error analysis.

Our numerical analyses concentrated on the nonlinear nutrient-phytoplankton-zooplankton dynamics under advection and diffusion within an ocean strait or sill, in an idealized 2D geometry. We first non-dimensionalized the PDEs, evaluated stability regions and selected three biological dynamical regimes: single stable points at all depths, stable limit cycles at the bottom of the euphotic zone, and stable limit cycles within the whole euphotic zone (the latter two cases have limit cycles that are depth and light dependent). We evaluated the effects of numerical parameters on the three biological regimes, but illustrated only the most relevant results. In addition, for each of these biological regimes, we examined two types of coupled physics-biology interactions: biological terms dominating; and advection and biological terms balancing. For the balanced situation, relatively common in the real ocean, we considered biological dynamics that were either as fast as (e.g. coastal ocean) or slower than

advection time-scales.

In the regime where biological terms dominate, we found that both the quadrature-based and quadrature-free treatment of the source-terms give accurate, convergent results, although the quadrature-based algorithm had slightly smaller errors. We also showed that oscillations can occur solely due to numerics (Gibbs-like phenomena) for a high-order discretizations. A key result is that, for any numerical scheme, careful one-dimensional studies should be performed to understand the potential errors from the nonlinear source-term discretization.

For the case of approximately balanced advection and biological terms, we compared low and high order temporal and spatial discretizations, and studied quadrature-based and quadrature-free discretizations of the source terms. Using point-wise error measures that quantify phase and path-accumulated errors, we found that for lower-order temporal discretizations, the errors grew rapidly and would lead to inaccurate solutions for applications with faster biological timescales or longer integration times. We also showed that the quadrature-based source-term discretization was more accurate in regions where the solution was under-resolved, but in well-resolved regions, there was only a 0.1% discrepancy, and the quadrature-free algorithm could be used for efficiency purposes. By quantitatively evaluating the truncation error and smoothness of the solution fields, we confirmed that higher-order spatial discretizations were more accurate in regions where the solution was smooth (i.e. resolved enough), but less accurate where non-smooth (un-resolved) due to Gibbs-like oscillations. Finally, we demonstrated the importance of non-dissipative numerical schemes when biological patches are present which is common in the real ocean. First, we found that effects of numerical dissipation were amplified by biological activity, causing dissipation errors to increase faster with integration time. Higher-order spatial discretizations were more accurate when modeling biological patches because they maintained the patches while lower-order schemes did not. For resolved biology (e.g. as in Fig. [6-11]), higher-order schemes on coarser grids were for the same cost more accurate than lower-order schemes on finer grids. This conclusion is most important for longer-term simulations, since we showed that the long-term phytoplankton dynamics can be altered

by the numerical error. We showed that the low-order scheme had a single stable point, while the high-order one retained the stable limit cycle described by the mathematical model and its parameters. This has major implications for fundamental studies of biological blooms, patchiness and other nonlinear dynamics in coastal regions with complex bathymetric features such as straits, sills, ridges and shelfbreaks. One can expect similar implications for longer-term eddy-resolving ecosystem studies or climate applications.

Based on our results, future research directions are to further develop schemes to reduce Gibbs-like oscillations without significant loss of accuracy and efficiency (e.g. Persson and Peraire (2006)). Without oscillation limiters or filtering, the optimal performance could be obtained by using different polynomial degree basis functions in the domain, where low-order elements could be used in non-smooth regions while high-order elements could be used in smooth regions. Because the smoothness can be determined from the discretization, an adaptive grid and polynomial degree scheme could be developed. Another possibility in this case could be to increase the grid resolution and decrease the order of schemes (e.g. medium order schemes, i.e. $(g3, p3)$ or $(g3, p4)$) up to the point when numerical oscillations reach the size of other errors. Another research direction is to develop and evaluate schemes that would preserve the non-negativity of the biological solution. Our results can now be utilized for idealized studies of biological dynamics in straits or sills. Uncertainty quantifications (Lermusiaux, 2006, Sapsis and Lermusiaux, 2009) as well as adaptive model learning (Lermusiaux, 2007) for biological predictions would also be useful. Finally, we are now well positioned to implement these new methods in three-dimensional ocean modeling systems (e.g. MSEAS-Group (2010)) for realistic coupled biogeochemical-physical ocean science and applications.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Sensitivity of Phytoplankton Productivity to Non-Hydrostatic Dynamics over Idealized Banks

Internal tides/internal waves can affect biological productivity through a number of different mechanisms. These include: breaking internal waves that can enhance mixing, transporting nutrients into the euphotic zone (MacIntyre and Jellison, 2001, Sangrà et al., 2001, Gaxiola-Castro et al., 2002, Yang et al., 2010); the aggregation of phytoplankton above internal wave troughs and below internal wave crests (Kushnir et al., 1997, Lennert-Cody and Franks, 2002); and the movement of phytoplankton through the non-linear light-field (Patterson, 1991, Holloway, 1984, Evans et al., 2008). With the advent of non-hydrostatic ocean models, quantitative numerical studies of the effect of internal waves/tides on phytoplankton productivity over shelfbreaks and banks have recently become possible. To our knowledge, little work has been conducted to quantify the effect of non-hydrostatic effects on phytoplankton productivity on regional scales with steep topography. In Lai et al. (2010), the non-hydrostatic FVCOM model is used to model phytoplankton growth over Stellwagen Bank for an entire cycle of high-frequency internal waves, including their origin, propagation, shoaling, and dissipation. While Lai et al. (2010) do compare their results from the non-hydrostatic model to a hydrostatic one, they do not try to quantify

the importance of non-hydrostatic effects for various stratifications and tidal forcing amplitudes.

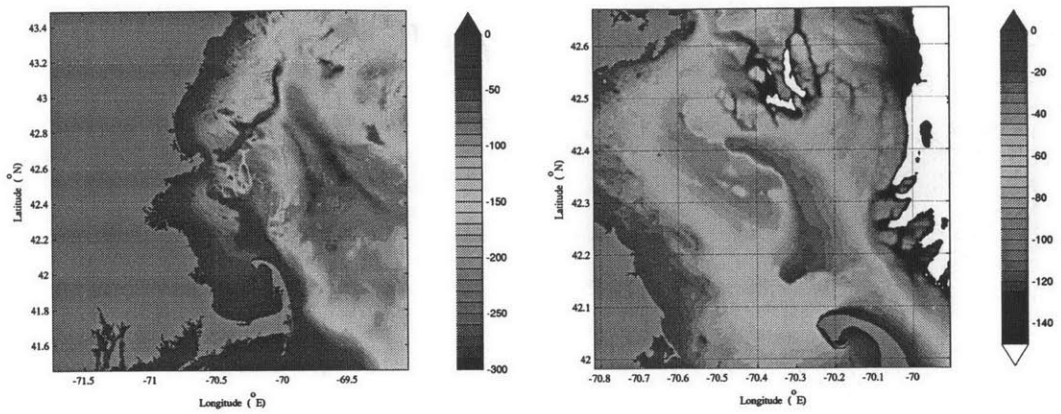
We are also motivated by the fascinating Stellwagen Bank (SB) ecosystem and its dynamics. Within Massachusetts Bay and the greater Gulf of Maine system, SB is widely known for its highly productive pelagic ecosystem with actively feeding populations of fish and cetaceans. Even though (sub)-mesoscale physical and biogeochemical dynamics in Massachusetts Bay has been studied (e.g. Lermusiaux (2001), Besiktepe et al. (2003) and Moreno (2007)), the detailed dynamics of the SB ecosystem remains relatively unquantified. Therefore, our goal here is to study and quantify one aspect of such an ecosystem, specifically, the importance of non-hydrostatic effects on phytoplankton productivity for tidally-forced circulations over a bank, considering various stratifications and tidal forcing amplitudes. In our focused study, the biological productivity will be primarily affected by the vertical motion through the non-linear light field, and possibly due to breaking internal waves. However, we do not have a particle drag model or biological motility models that would allow biological constituents to aggregate, so these effects are not captured. Only aggregation due to flow-driven accumulation of passive tracers is considered (for example, accumulation due to non-coherent structure dynamics, e.g. Pasquero et al., 2004).

7.1 Problem setup

In this section we describe the geometric setup, including the mesh and time-discretization, we give the parameters and equations we are solving and we give our initial and boundary conditions.

7.1.1 Geometry

The idealized geometry for our study is based on that of Stellwagen Bank and its region, illustrated in Fig. [7-1]. Our goal is to capture essential features of SB using an idealized analytical expression with a minimal number of parameters. To this end, we choose to use a piece–wise linear representation of the bank as shown in Fig. [7-2].



(a) Massachusetts Bay

(b) Stellwagen Bank

Figure 7-1: Massachusetts Bay and Stellwagen Bank geometries, showing the coastlines and the complex bathymetry (red shallowest, blue deepest, with ranges left 0-300m and right 0-160m). This bathymetry and the corresponding multiscale dynamics require high-resolution simulations.

We use a total domain length of 100 km. For the geometric parameters defined in Fig. [7-2], we used the values from Table [7.1]. Next, we explain how we obtain this idealized but representative SB geometry.

This simplified representation of the geometry captures a number of important features. The slopes of the geometric features are important for internal wave generation since some of these slopes can be critical (Vlasenki et al., 2005, Gerkema and Zimmerman., 2008, Kelly et al., 2010); in our case, particularly s_1, since it is the steepest. The bathymetric slopes also affect the vertical displacement of phytoplankton through the non-linear light-intensity field. The depth (or height) of the bank H_{bank} is important for phytoplankton growth, because as this depth decreases, phytoplankton above this point is brought closer to or within the euphotic zone. The depth of the Bank at the coast, H_{coast}, and at the open boundary, H_{OB}, are are less critical, but they serve to constrain our domain size. We keep these parameters constant for the current study, but their values may have important effects, e.g. for other banks.

To find realistic values for the slopes and heights, we characterized SB's bathymetry. Using Smith and Sandwell (1997) topography, we first extracted the -34m depth contour Fig. [7-3]. Then, using a steepest descent algorithm, we found the slopes

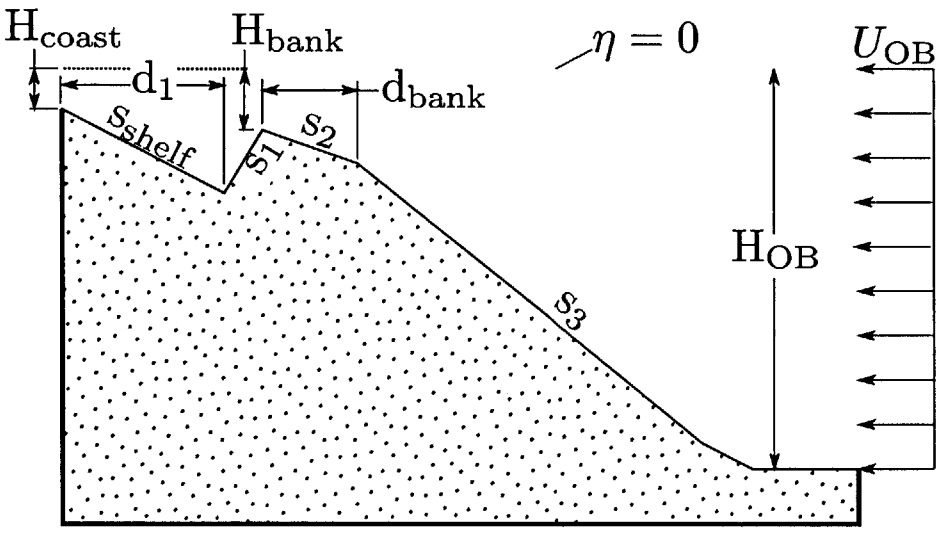


Figure 7-2: Idealized bank geometry. The extents of the domain are 200 m deep by 100 km long.

Table 7.1: Values of the geometric parameters.

Parameter	Value	Units
s_{shelf}	-2.1	[m/km]
s_1	6.74	[m/km]
s_2	-1.4	[m/km]
s_3	-3.2	[m/km]
d_1	20.11	[km]
d_{bank}	12.0	[km]
H_{bank}	30	[m]
H_{coast}	20.0	[m]
H_{OB}	200.0	[m]

surrounding SB Fig. [7-4]. These lines were then simplified using the Recursive Ramer-Douglas-Peucker Polyline Simplification (Ramer, 1972, Douglas and Peucker, 1973). The slopes were then calculated for the simplified lines, and sorted according to the normal of their initial descent direction. Using the lines sorted by their direction, we calculate mean slopes for s_1, s_2, and s_3.

7.1.2 Spatial and Temporal Discretizations

To discretize the geometry, we use a variable size mesh with the smallest horizontal resolution of 200 m and the largest of 2 km. We also employ 30 vertical sigma-layers,

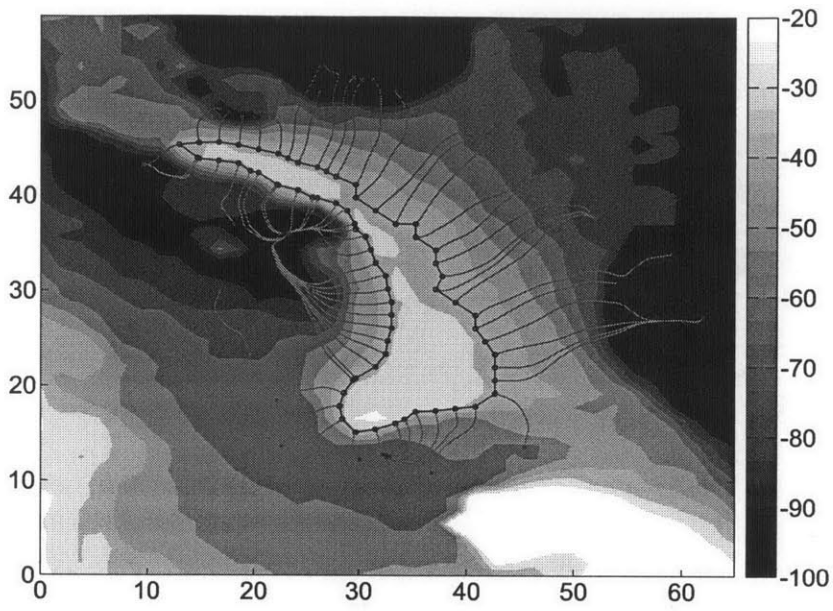


Figure 7-3: Stellwagen Bank bathymetry with -34 m contour, and steepest-descent paths. The color of the steepest-descent paths correspond to the colors in Fig. [7-4]. This procedure was used to characterize the Stellwagen Bank geometry to obtain realistic values for the idealized bathymetry.

giving a minimum vertical mesh size of 1 m (see Fig. [7-5]). We use a linear (p=1) polynomial basis for second-order spatial accuracy. Also, since we are using a 3D simulation code, we also need to specify a width of 100 km, but we only use 1 linear element in this direction to give an approximately 2D simulation.

For the time-step, we divide a tidal cycle into 4096 equal sections to give $\Delta t \approx 11$ s. This time-step satisfies the CFL stability criteria for the explicit treatment of the advection and internal gravity wave propagation. The horizontal advection speed is expected to be on the order of 50 [cm/s], with a CFL restriction $\Delta t < A_1 \frac{\Delta x}{c}$, where c is the magnitude of the velocity, and A_1 is a constant dependent on the time-integrator. The internal gravity waves propagate at $\sqrt{g'H}$, where g' is the reduced gravity $g' = g\frac{\Delta\rho}{\rho}$. The maximum internal gravity wave speed for our highest stratification is approximately [70 cm/s]. This wave speed has a CFL-like condition of $\Delta t < A_2 \frac{\Delta x}{\sqrt{g'H}}$, where A_2 is, again, a constant dependent on the time-integrator. We do not have to deal with a CFL-like restriction for surface gravity waves because we have treated the free-surface implicitly, and the internal gravity wave speed is approximately 5 times

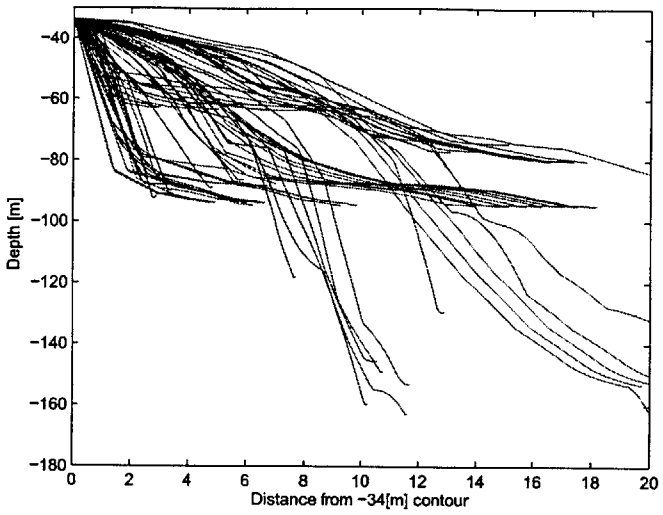


Figure 7-4: Depth variation in the steepest-descent direction from the -34[m] contour on Stellwagen Bank used to calculate the slopes of the idealized bathymetry. The color of the steepest-descent paths correspond to the colors in Fig. [7-3]

smaller for our highest stratification.

7.1.3 Equations and Parameters

Biological fields: The dimensional parameters used in the biological model (6.7)-(6.9) are summarized in Table [7.2]. Note that we are using no vertical or horizontal diffusion for these fields. Also, we primarily use values from Lai et al. (2010) who studied the SB region, unless the functional form for the term in the equation is different than that of our model.

Table 7.2: Values of the dimensional numbers entering the NPZ equations (6.7)-(6.9), that are used for our idealized bank study.

Parameter	Value	Reference
$\mathcal{U}$	1.5 [1/day]	Ji et al. (2008)
k_s	1 [μmolN]	Lai et al. (2010)
d_P	0.1 [1/day]	Ji et al. (2008)
d_Z	0.1 [1/day]	Lai et al. (2010)
g_{bio}	0.25 [1/day]	Lai et al. (2010), Ueckermann and Lermusiaux (2010)
ν_{bio}	0.1 [L/μmolN]	Lai et al. (2010)
a	0.4	Lai et al. (2010)
h	5.88 [m]	Lai et al. (2010)
κ	0 [m^2/s]	

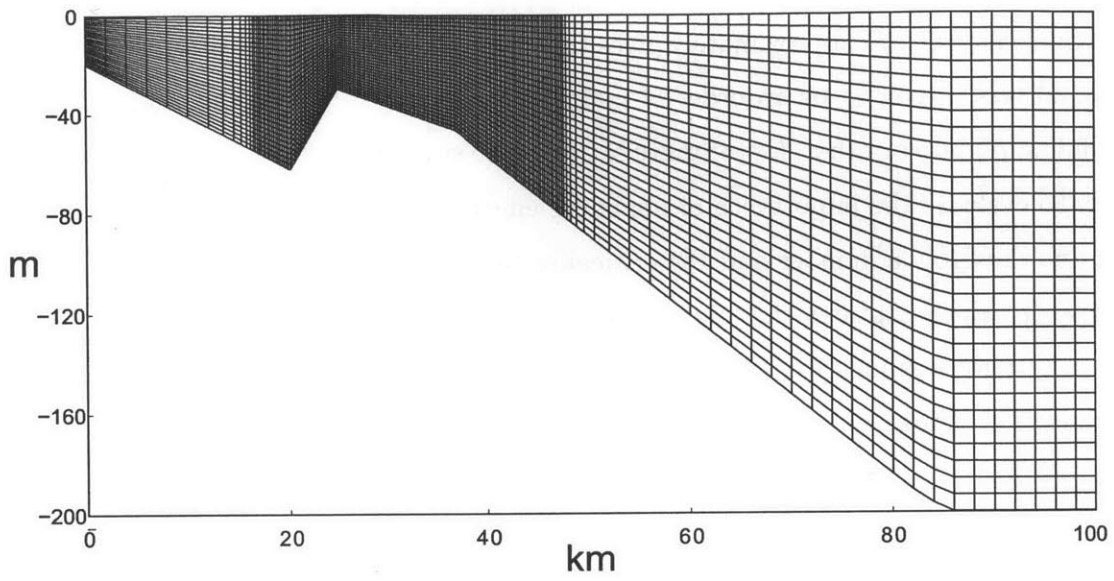


Figure 7-5: Variable resolution mesh used for idealized banks simulations.

Physical fields: For the physical fields we are solving the hydrostatic and non-hydrostatic equations, (4.28)–(4.30) and (4.20)–(4.23), respectively, using a horizontal turbulent diffusion coefficient of $\nu_{xy} = 200$ cm^2/s a vertical turbulent diffusion coefficient of $\nu_z = 5$ cm^2/s, and $f = 0$. We are also solving an additional tracer advection equation directly for the density perturbation $\frac{\rho'}{\rho_0}$, which does not contain any diffusion.

For the uniform (linear) stratification, the Brunt–Väisälä frequency $N = \sqrt{\frac{-g}{\rho_0}\frac{\partial \rho'}{\partial z}}$, we examine a range of $N = [0.005, 0.05]$. This range was obtained by first averaging density profiles obtained from CTD casts over SB during the Assessment of Skill for Coastal Ocean Transients (ASCOT) project in 2001 (Robinson et al., 2002a). The buoyancy frequency was then numerically calculated from the averaged density profile, giving a maximum stratification of 0.045, and a minimum of 0.003. To simplify the range, we rounded to 0.05 for the maximum, and took the minimum an order of magnitude smaller so that we could examine the effect of stratification.

For the tidal amplitude, we use a range of $U = [0.02, 0.2]$ m/s for the barotropic velocity at the open boundary. From measurements, the maximum velocity at the bank is approximately 0.5 m/s. We ran numerical experiments with unstratified flows

to find that 0.2 m/s approximately gives this 0.5 m/s velocity at the bank. The lower bound was chosen to be an order of magnitude smaller, such that the effect of the tidal amplitude or other flow amplitudes could be studied.

To evaluate the importance of nonlinear effects for these parameters, we can examine the Froude number and the criticality (or steepness) parameter. The Froude number, which captures the joint effect of tidal amplitude and stratification, serves as a test for nonlinearity, where $\mathrm{Fr} = \frac{U}{NH} > 1$ roughly indicates nonlinearities are important. Additionally, we can calculate the criticality (or steepness) parameter

$$\epsilon = \frac{s_1}{\omega/N}, \tag{7.1}$$

which compares the slope of the topography to the slope of the internal wave rays at the tidal frequency ω e.g. (Garrett and Kunze, 2007). When $\epsilon = 1$, the slope is described as "critical," whereas gentler slopes are subcritical, and steeper ones are supercritical. If we plot the internal Froude number and the criticality parameter for the parameter space chosen, we obtain Fig. [7-6]. We note that for the majority of the parameter space, either the Froude number or the criticality parameter is larger than 1. The shaded (with grey lines) region in the Fig. [7-6] indicates where both ϵ and Fr are greater than 1. Also, note that we have plotted the Froude number for the mode 1 internal waves. The higher mode internal waves will exhibit nonlinearities sooner, for example, the mode 2 internal wave will be greater than 1 when the plotted $\mathrm{Fr} > 0.5$. Finally, these boundaries between linear and nonlinear are not sharp, and Fig. [7-6] is only meant to serve as a rough guide.

7.1.4 Initial Conditions

Biological fields: We initialize the NPZ model as in §6.2 by first calculating the steady–state of the equations (6.7)-(6.9) which can be found by setting $\frac{\partial \Phi}{\partial t} + \nabla \cdot (\mathbf{u}\Phi) - \nabla \cdot \frac{1}{P_e} \nabla \Phi = 0$. The background nutrient field is specified as

$$\mathcal{N}_T = \min\left(5 - 5\frac{z}{0.25h}, 10\right), \tag{7.2}$$

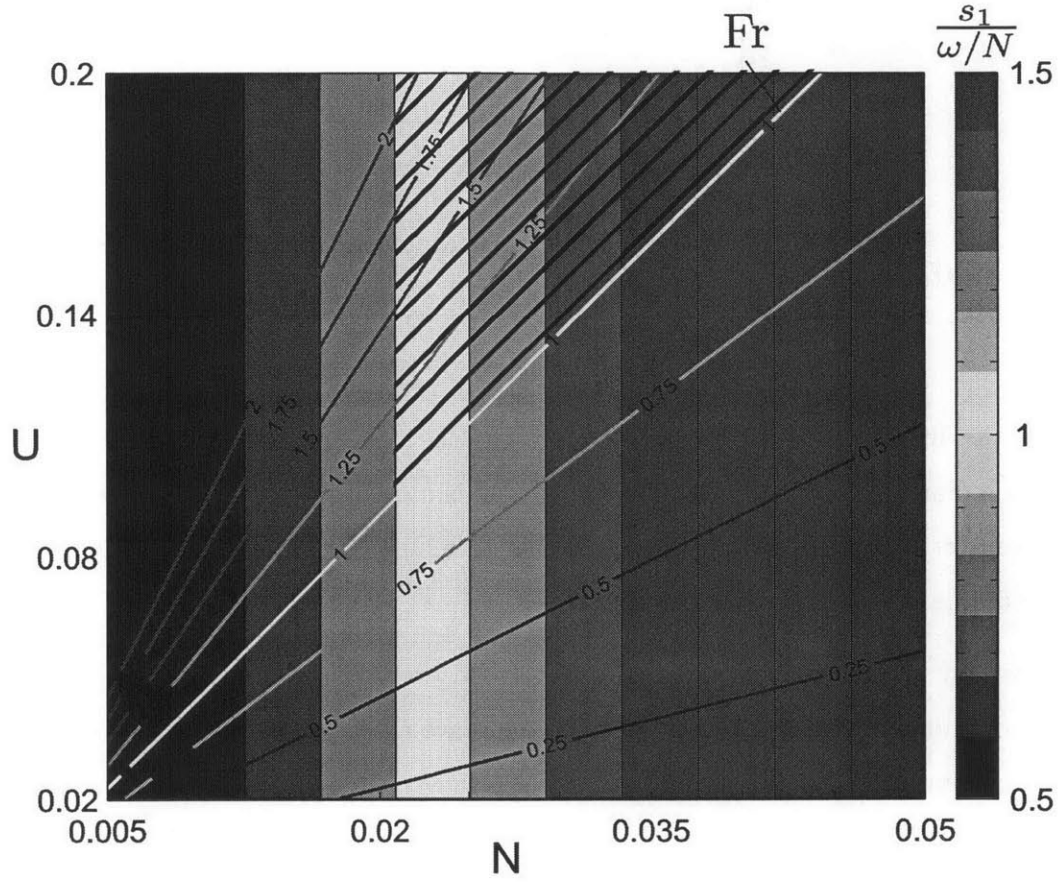


Figure 7-6: Contours of the mode-1 Froude number over the criticality parameter. The lines of the Froude number are marked by Fr and colored accordingly. The criticality parameter space is filled in color according to the colorbar and varies linearly with N. The triangle region shaded with grey lines indicates parameters where both Fr and ϵ are greater than one, and where we expect both nonlinearities to be important.

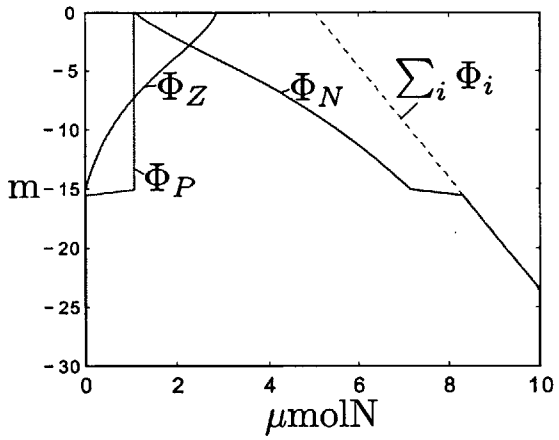


Figure 7-7: Steady state concentration of biological tracer fields over the first 30 m in depth. Plotted versus depth are the concentrations of phytoplankton P, zooplankton Z, nitrogen N and the total biomass (their sum).

where the linear profile is chosen to represent the nutrient profile due to sinking and to sediment-mixing sources. Such a profile is commonly observed in the region Besiktepe et al. (2003). We note that Lai et al. (2010) used a constant-with-depth $\mathcal{N}_T \approx 6.8$ μmolN.

The phytoplankton field is slightly perturbed everywhere to ensure that phytoplankton initially below the euphotic zone will have an opportunity to grow when it is brought into the light field

$$\Phi_P = \Phi_P^{\text{steady}} + 0.01. \tag{7.3}$$

These analytically calculated initial conditions are shown in Fig. [7-7], and on the mesh they are shown in Fig. [7-8]. Note that the sharp gradient in the phytoplankton field results in a saw-tooth pattern on the mesh (Fig. [7-8]). This happens because when using a sigma-coordinate mesh for the vertical discretization, the degrees of freedom are not horizontally aligned but form as saw-tooth pattern as observed. This continuity will require a good slope-limiter to avoid non-physical, negative, phytoplankton, and is a good test of our numerical scheme.

Physical fields: To initialize the density, we use a linear stratification, with the

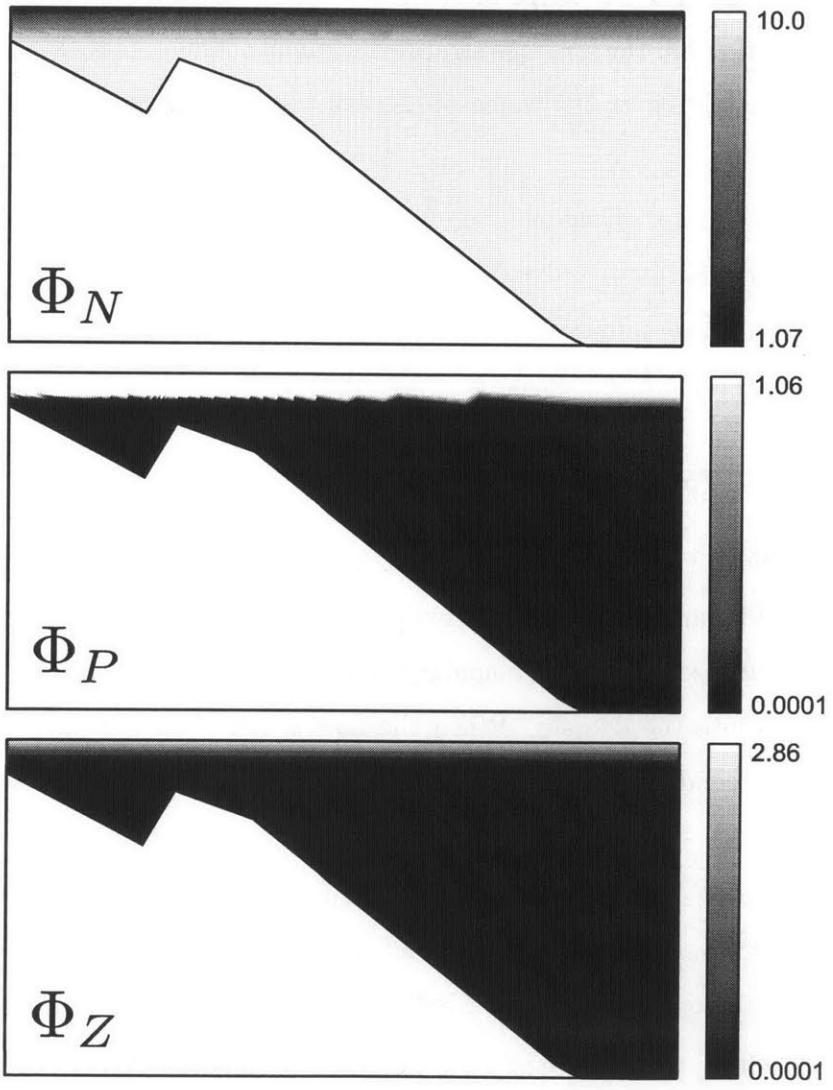


Figure 7-8: Initial biological tracer fields on the numerical mesh.

degree of stratification dependent on N

$$\frac{\rho'(z,\, t=0)}{\rho_0} = \frac{N^2}{\mathrm{g}} z - \frac{125 N^2}{\mathrm{g}}. \tag{7.4}$$

Recall that we computed our range of N values by examining an averaged density stratification profile from the strongly-stratified summer season. As such, our density profiles approximately cover the range of density variations observed at SB in the summer.

The velocities are initialized as zero everywhere, and for the free-surface we use an analytical expression for a wedge (4.129), as an initial conditions. The free-surface is initialized at the lowest level calculated from the analytical solution.

7.1.5 Boundary conditions

Biological fields: For the biological fields, we use the no-normal flux boundary condition at solid boundaries (coast, bottom boundary) and at the free-surface. At the open boundary, we use a Dirichlet boundary condition equal to the initial conditions when the flow is into the domain. When the flow is out of the domain, we do not need a boundary condition since we calculate our numerical flux using the upwind value.

Physical fields: For the density, we use the same boundary conditions as for the biological tracers, that is, no-normal flux at the solid boundaries and free-surface, and Dirichlet at the open boundary for inflows.

For the velocities, we use free-slip on the bottom and free-surface, no-slip at the coast, and a zero Neumann boundary condition at the open boundary. For the free-surface, we use a zero Neumann boundary condition at the coast, and a Dirichlet boundary condition at the open-boundary. Therefore, the M_2 tides, with tidal frequency $\sigma = \frac{2\pi}{12.42 \times 3600s}$, are forced with the free-surface height only, and the velocity is free to vary in depth at the open boundary. The non-hydrostatic pressure uses zero Neumann boundary conditions everywhere.

7.2 Results

In this section we begin by performing a scale analysis, followed by detailed numerical simulations. The scale analysis serves as a baseline comparison to ensure that the detailed numerical solution is producing reasonable results. We present numerical simulation results of phytoplankton productivity for hydrostatic and non-hydrostatic simulations. We study the effects of the internal-tide-bottom-slope criticality and of the Froude number. We show results for constant criticality with varying Froude number, as well as approximately constant Froude number with varying criticality. Additionally, we show detailed phytoplankton fields over time, and we examine the flow-field for a particular case.

7.2.1 Scale analysis

In this section we perform a scale analysis on the biological productivity to serve as a baseline for the detailed numerical model. We begin by describing how vertical displacements are estimated. Then we use the calculated vertical displacements to obtain a depth-averaged light-intensity field. Using the averaged light-intensity field, we estimate the time-integrated ("steady-state") of the system and the corresponding productivity (this follows the circadian biogeochemical balance approach of Besiktepe et al. (2003)).

To estimate the effects of vertical displacement on the biology, we begin by estimating the effect of moving the phytoplankton field vertically through the non-linear light profile. We can approximate the magnitude of vertical velocity using the scale analysis (see Cushman-Roisin and Beckers (2011)):

$$\begin{aligned} \frac{W/H}{U/L} &= \mathrm{Fr}^2, \\ \Rightarrow W &= \frac{UH}{L}\min(1, \mathrm{Fr}^2), \\ &= Us_1 \min(1, \mathrm{Fr}^2), \end{aligned} \tag{7.5}$$

where the Froude number is used to estimate the reduction in vertical displacement

due to stratification, and should not increase the vertical velocity. Now we can estimate the maximum vertical displacements by integrating the vertical velocity over half a tidal period

$$\begin{aligned}\delta H &= \int_0^{\pi\omega^{-1}} W dt \\ &= \int_0^{\pi\omega^{-1}} \sin(\omega t) \mathrm{d}t U s_1 \min(1, \mathrm{Fr}^2) \\ &= \frac{2}{\omega} U s_1 \min(1, \mathrm{Fr}^2) \end{aligned} \tag{7.6}$$

where the Froude number $\mathrm{Fr} = \frac{U}{NH}$, is calculated using $H = H_{\text{bank}} = 30$ [m], and $U = \frac{U_{\text{OB}}}{H_{\text{OB}}} H_{\text{bank}}$ (from mass conservation).

Then, we integrate the non-linear light fields in the phytoplankton source-term to find an average light-intensity over a tidal period

$$\begin{aligned}\bar{I}(z) &= \frac{1}{2\pi} \int_0^{2\pi} \exp\left(hz + \frac{hz}{30}\delta H \sin(t)\right) dt, \\ &= \exp(hz)\, I_0\left(\frac{hz}{30}\delta H\right),\end{aligned}$$

where I_0 is the modified Bessel function of the first kind, and we have assumed the vertical displacement varies linearly from the top of the bank to the free-surface. This also assumes that the timescale of the vertical displacements are much faster than the biological timescales. Once the averaged light-intensity is found, we calculate the circadian-averaged steady-state biology at all depths, and numerically integrate the productivity

$$\mathcal{P}_{\delta H} = \int_{-H}^{0} \mathcal{U} \mathrm{e}^{z/h} \frac{\phi_P \phi_N}{\phi_N + k_s} dz. \tag{7.7}$$

This integrated productivity is compared to the steady-state of the $\delta H = 0$ case, $\mathcal{P}_{\text{steady}}$. The vertical displacements and the relative productivities are plotted in Fig. [7-9]. From the figure we note that we would expect an increase in productivity on the order of a 25% using this biological model, with these parameters. Additionally,

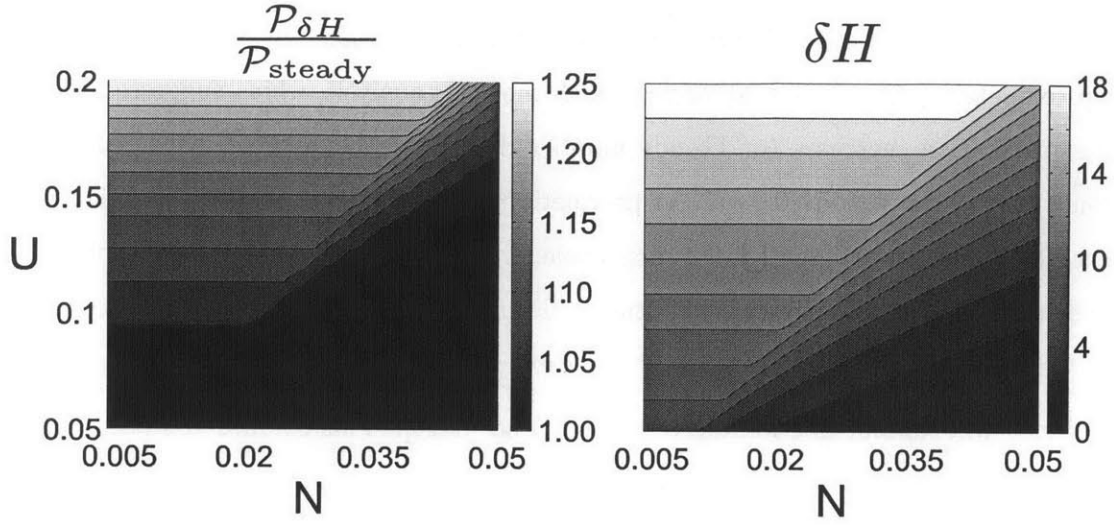


Figure 7-9: Order of magnitude estimation of vertical displacement (right) and its effect on phytoplankton productivity (left). The maximum increase in productivity is on the order of 25%.

as the stratification increases, the vertical displacements are expected to decrease, and the biology is expected to be less affected (on linear average). Based on this, we focus several of our model runs on the lower stratification levels. However, we also note that local nonlinear wave effects not accounted for in our scale analysis can become significant when stratification is large and this thus also investigated.

7.2.2 Numerical simulations and dynamical results

Productivity. To estimate the importance of non-hydrostatic effects with increasing Froude number (or tidal forcing amplitude), we plot the depth-integrated productivity normalized with the steady-state depth-integrated productivity over time for various Froude numbers (Fig. [7-10]). The value of the criticality parameter chosen for this first study is relatively small, $\epsilon = 0.23$. The left (right) column shows the effect of the unsteady tidally forced flow on the biological productivity for the hydrostatic (non-hydrostatic) simulation. For areas in the domain where $\frac{\mathcal{P}_{\text{HS}}-\mathcal{P}_{\text{steady}}}{\mathcal{P}_{\text{steady}}} = 0$, the unsteady forcing has no effect on the depth-integrated productivity. When $\frac{\mathcal{P}_{\text{HS}}-\mathcal{P}_{\text{steady}}}{\mathcal{P}_{\text{steady}}} \neq 0$, the unsteady forcing either increases or decreases the depth-integrated biological productivity. Examining Fig. [7-10], it shows that as the

Froude number increases, the unsteady forcing has a larger effect on the biology, up to approximately 20%. This in agreement with the scale analysis, which suggested that as stratification increases (or Froude number decreases) the vertical displacements would be smaller, leading to smaller productivity. Additionally, once the simulation reaches approximately the 5^{th}-6^{th} tidal cycle, the depth-integrated productivity differences reach their maximum amplitudes. Finally, we note that the wavy patterns over tidal cycles are approximately at the same length-scale as the tidal excursion.

Next, we examine the middle column of Fig. [7-10]. This column compares the effect of hydrostatic to non-hydrostatic simulations on the depth-integrated productivity normalized with the steady-state productivity. We note that the differences are small far away from the top of the bank (indicated with a cross in the plots), while near the top of the bank, they are on the order of 1% (for this small critically $\epsilon = 0.23$). The hydrostatic simulation tends to produce a larger depth-integrated productivity to the left of the bank, while the non-hydrostatic tends to produce a larger depth-integrated productivity to the right and top of the bank. As the Froude number increases, the differences between the hydrostatic and non-hydrostatic simulations also tend to increase, although the second-largest Froude number shows larger differences than the largest Froude number.

To estimate the importance of non-hydrostatic effects with increasing the criticality parameter (that is, when increasing stratification and tidal amplitude as to keep the effect of stratification the same), we plot the depth-integrated productivity normalized with the steady-state depth-integrated productivity over time for various criticality parameters, with an approximately constant Froude number (Fig. [7-11]). As in Fig. [7-10] the left (right) column shows the effect of the unsteady flow on the biological productivity for the hydrostatic (non-hydrostatic) simulation. Examining Fig. [7-10], it shows that as the criticality increases, the unsteady forcing has a larger effect on the biology, up to approximately 50%. This value is on the same order, but logically larger, than the 25% increase we calculated for the linear scale analysis. Also, as the criticality increases, the depth-integrated productivity is smaller, suggesting that biological activity is decreased compared to the steady reference. Finally, with

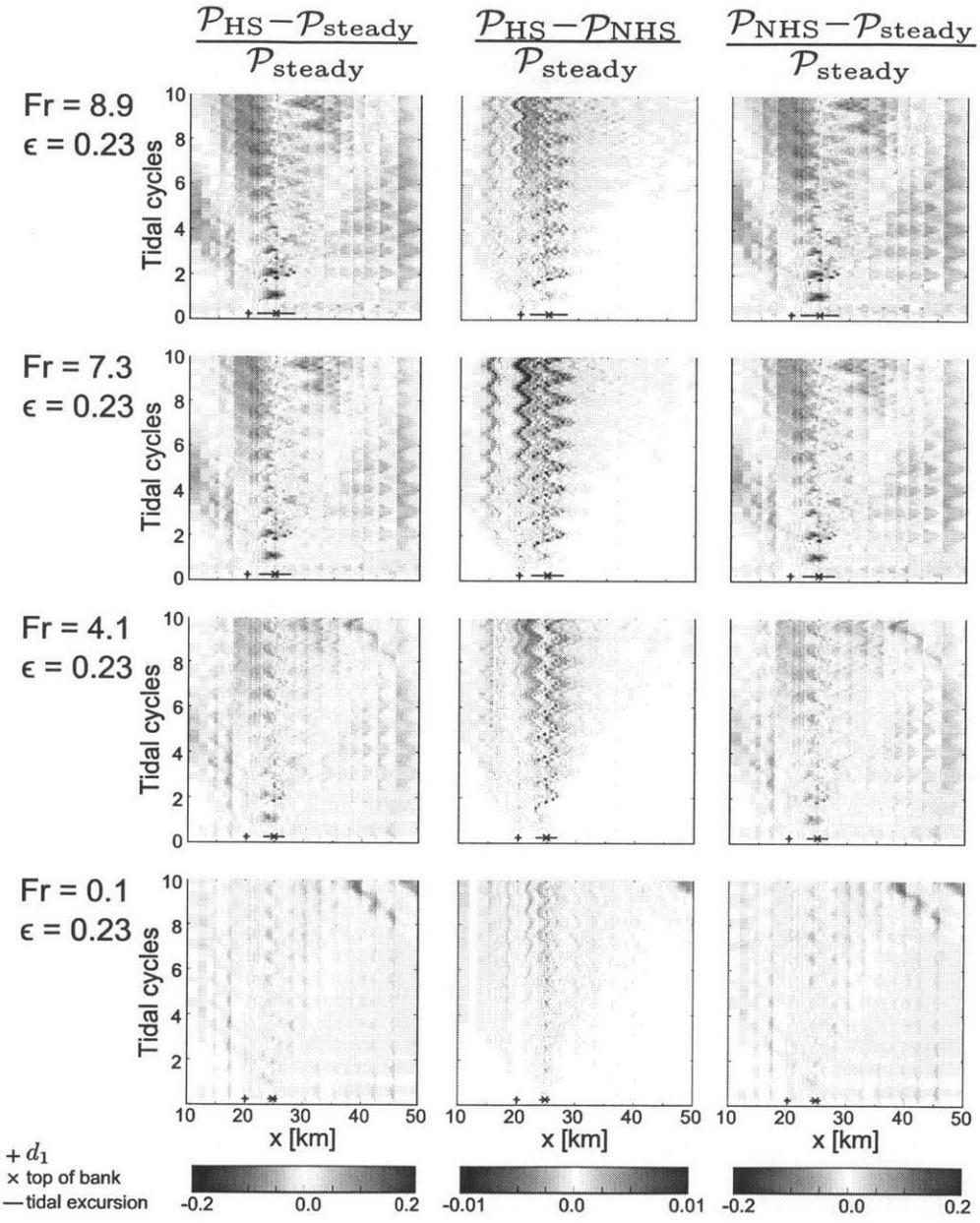


Figure 7-10: Depth-integrated relative productivity in time and across the bank for various Froude numbers and constant criticality of $\epsilon = 0.23$ for hydrostatic minus steady (left), hydrostatic minus non-hydrostatic (middle) and non-hydrostatic minus steady (right). The bottom side of the bank d_1 is indicated with a plus, the top of the bank with a cross, and the approximate tidal excursion with a line. Note the colorbar for middle column differs from that of the left and right columns.

increasing criticality, the wavy patters are no longer periodic, and episodic events in the productivity can be observed, which indicates an increase in nonlinearity.

Examining the middle column of Fig. [7-11], we note that the differences between the hydrostatic and non-hydrostatic simulations are not as localized to the top of the bank (indicated with a cross in the plots), and larger than observed previously (Fig. [7-10]) for the larger criticality parameters. Near the top of the bank, they are on the same order as the differences between the steady and unsteady simulations (side columns). As the criticality increases, the differences between the non-hydrostatic and hydrostatic runs increase. This is particularly evident when the criticality increases past unity.

We can also examine the effect of increasing criticality at a lower Froude Number. At a constant tidal forcing amplitude, we can increase the stratification to increase the criticality. Fig. [7-12] shows the effect of increasing criticality with a constant tidal forcing amplitude. As in Fig. [7-11] we see an increase in the difference between the hydrostatic and non-hydrostatic simulations. However, the differences are more localized and not as large compared to the higher Froude-number case in Fig. [7-11], even though again much larger than for the lower critically number (Fig. [7-10]).

Phytoplankton. While we have been examining the differences in depth-integrated productivity, we can also look at the actual phytoplankton fields over time. The phytoplankton fields approximately give a time-integrated view of the effect of the productivity differences. In other words, over time, even a small increase or decrease of the instantaneous and circadian-averaged productivity can lead to a large difference in phytoplankton fields. Differences can be large in mean concentrations but also in the spatial features of the phytoplankton field. These properties are illustrated and analyzed next.

We show the hydrostatic (left) and non-hydrostatic (right) phytoplankton fields at every 2^{nd} tidal cycle in Fig. [7-13], and over the 7^{th} tidal cycle in Fig. [7-14] for the case that showed the largest differences (Fr=1.9, $\epsilon = 1.1$, which are both well within our range of expected values, see Fig. [7-6].). From Fig. [7-13] we see that the differences between the hydrostatic and non-hydrostatic simulations are significant

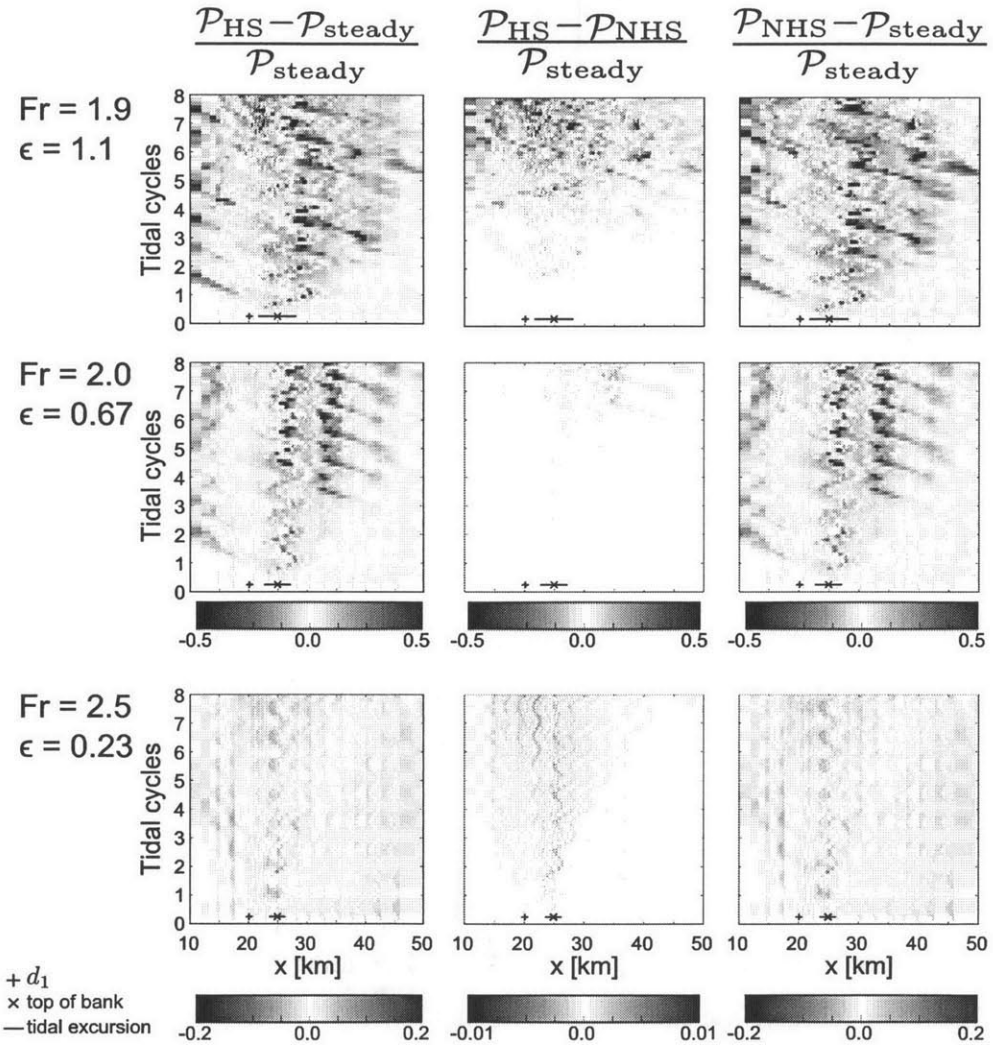


Figure 7-11: As in Fig. [7-10], but for Froude numbers Fr ≈ 2 and various criticality parameters. Note that here the colorbars for the top two rows are the same, but differ from the colorbars of the bottom row.

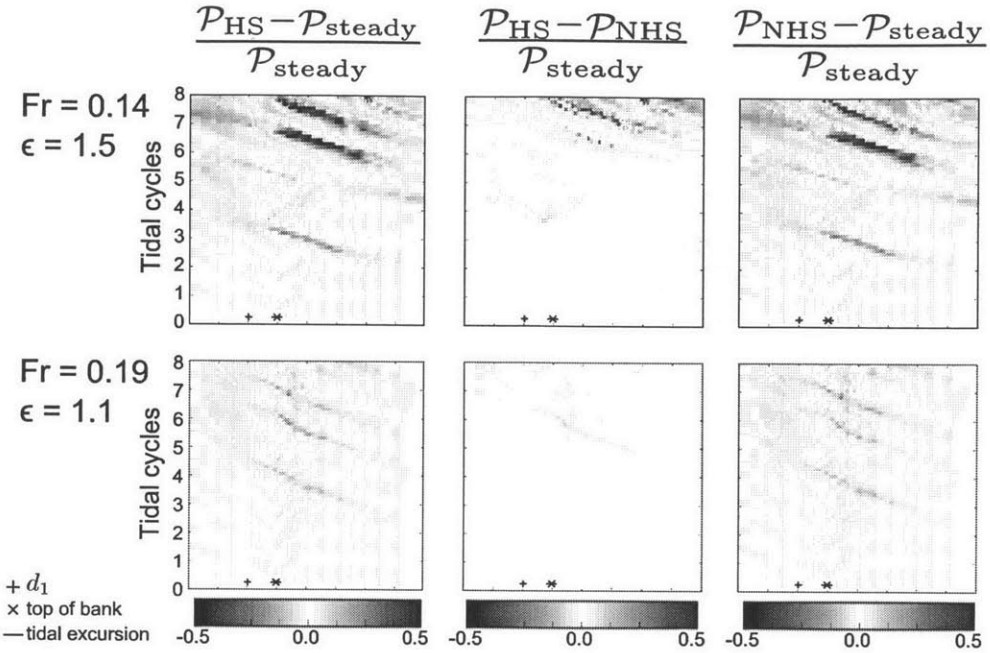


Figure 7-12: As in Fig. [7-11], but for smaller Froude numbers and at constant tidal forcing amplitude ($U = 0.2$) and increasing stratification ($N = [0.023, 0.032]$. Here, colorbars are all the same.

after the 6^{th} tidal cycle. Examining Fig. [7-14], we can see finer-scale features and larger vertical displacements in the hydrostatic simulation to the left of the top of the bank. As we decrease the criticality, we see that the non-hydrostatic and hydrostatic results become more similar (Fig. [7-15]).

Velocity. To further examine the differences observed between the hydrostatic and non-hydrostatic simulations, we compare the vertical velocity and the baroclinic horizontal velocity, see Fig. [7-16] – Fig. [7-17]. Examining Fig. [7-16], we note that the larger-scale features are similar for both simulations. The hydrostatic simulation, however, has smaller-scale features with larger vertical amplitudes to the left of the top of the bank.

Similar observations can be made for the baroclinic velocities (Fig. [7-17]). We again see larger velocities in the hydrostatic case, and smaller scale features, and the large scale features are again similar for both. Several of the internal tidal wave beams are similar but the nonlinear wave effects and smaller scale (non-hydrostatic) features are different.

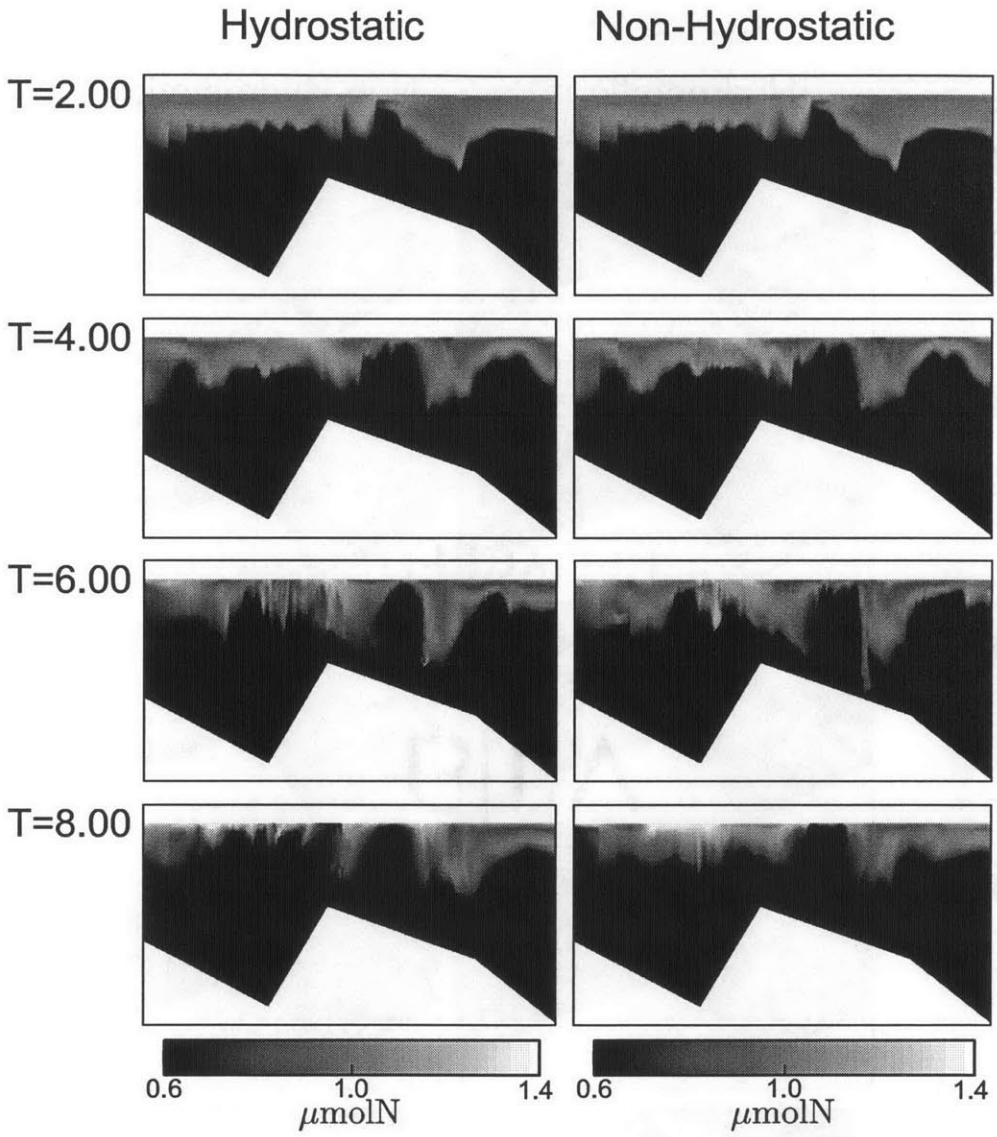


Figure 7-13: Phytoplankton field using hydrostatic (left) and non-hydrostatic (right) simulations plotted at every second tidal cycle for Fr ≈ 1.9, $\epsilon = 1.1$, focusing on the Bank region proper. The non-hydrostatic simulation has enhanced phytoplankton to the left of the bank, and lower phytoplankton to the right.

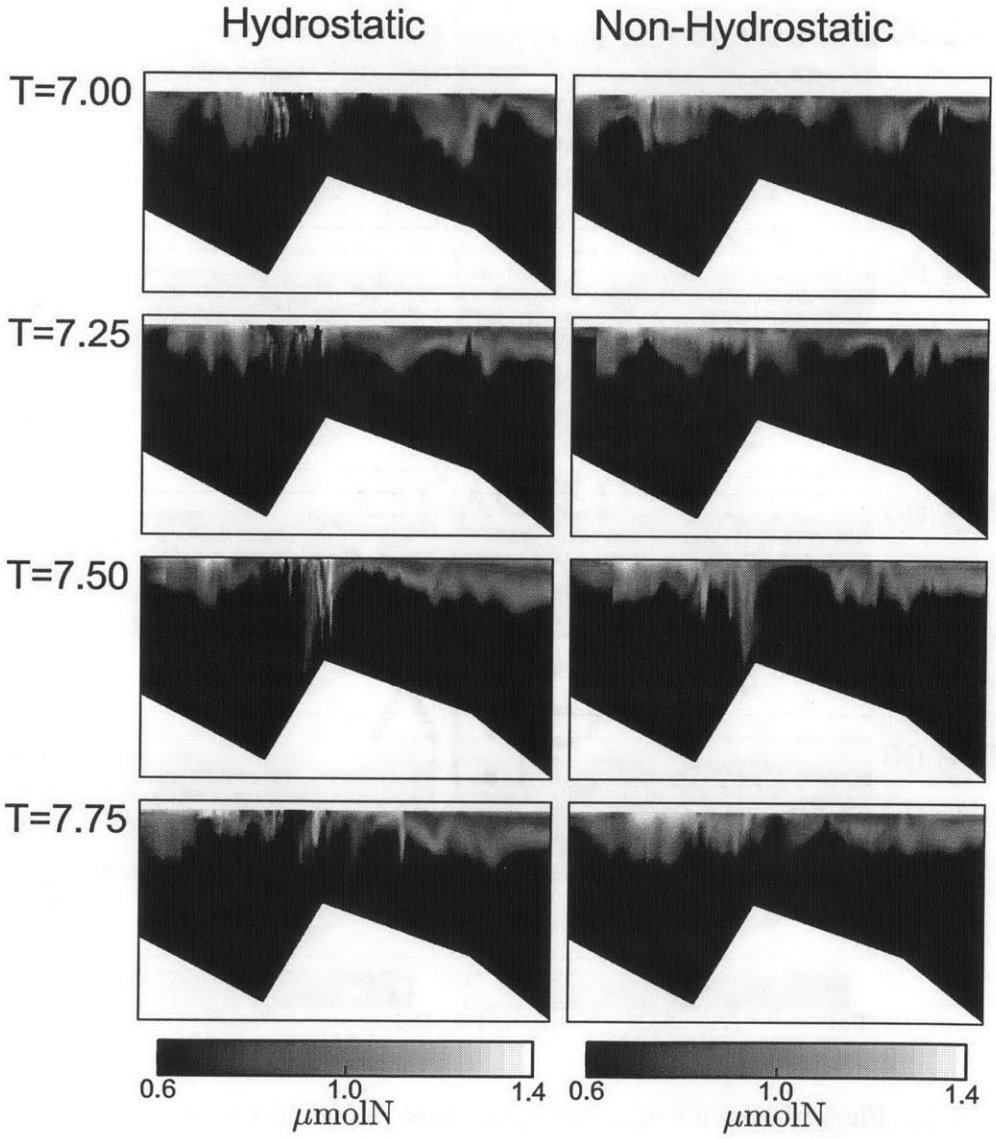


Figure 7-14: As in Fig. [7-13] but for the 7^{th} tidal cycle.

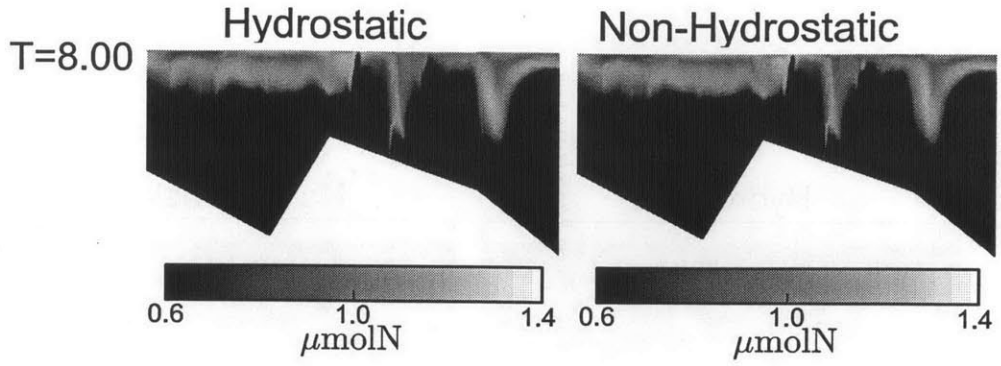


Figure 7-15: As in Fig. [7-13] but at the 8^{th} tidal cycle and for Fr ≈ 2, $\epsilon = 0.67$.

7.3 Discussion and Conclusions

We completed a set of computational studies of coupled physical-biogeochemical dynamics over an idealized coastal bank with steep bathymetric features, utilizing our new non-hydrostatic and hydrostatic HDG finite element ocean modeling schemes. We studied the effects of a varied set of physical parameters including tidal amplitude and stratification. We analyzed effects of non-hydrostatic physics by examining the depth-integrated biological productivity, the phytoplankton concentration, and the velocity fields.

The results of the numerical simulation experiments suggest that non-hydrostatic physics is important for accurate and quantitative modeling of biogeochemistry over steep topography when the flow is supercritical. The non-hydrostatic effects increase with increasing nonlinearity, both when the Froude number and criticality parameter increase.

Scientific or societal applications that require physical-biological coupling (that is, those that require unsteady effects to be modeled) would benefit from including non-hydrostatic effects when the flow is super-critical. From Fig. [7-10], the differences between non-hydrostatic and hydrostatic simulations at low criticality are small compared to the differences between the steady and unsteady simulations. This shows that increasing the Froude number at low criticality has little influence on the biological-physical coupling with non-hydrostatic effects. When going from sub-critical to super-critical at a constant Froude, however, the effect of non-hydrostatic

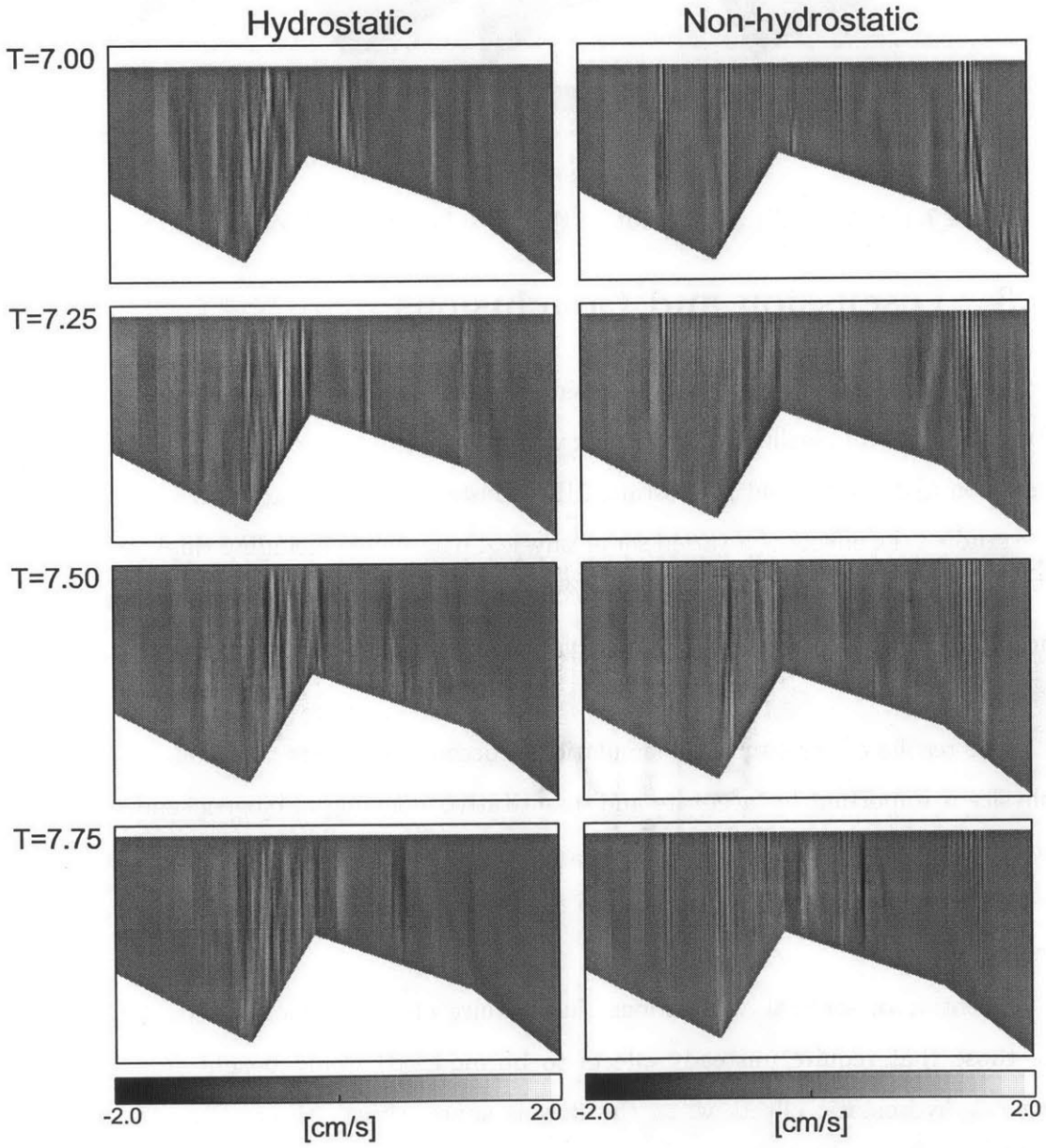


Figure 7-16: Vertical velocity using hydrostatic (left) and non-hydrostatic (right) simulations plotted for the 7^{th} tidal cycle, with Fr ≈ 1.9, $\epsilon = 1.1$.

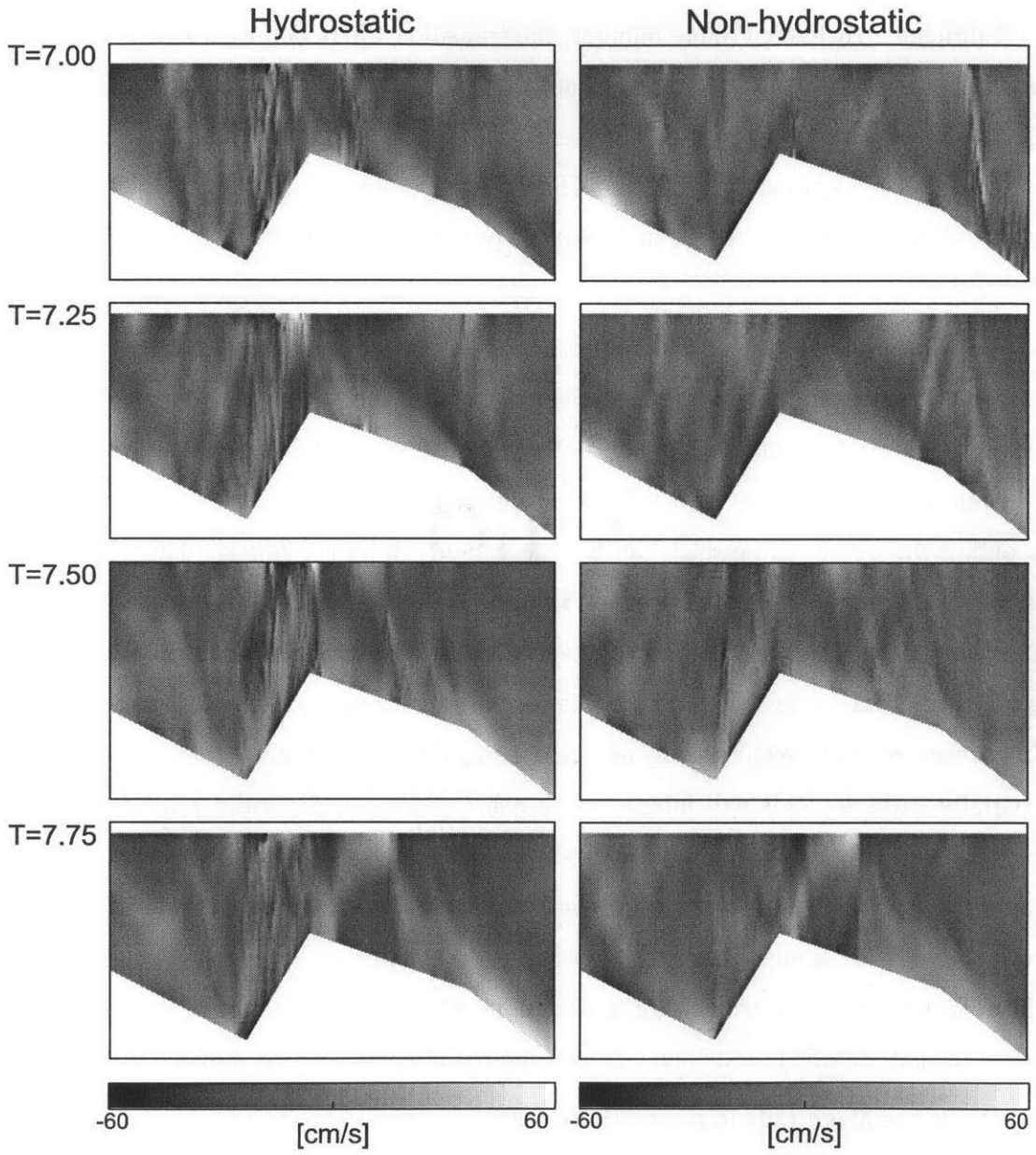


Figure 7-17: As in Fig. [7-16], but for the baroclinic velocity.

physics becomes as large as the effect of unsteadiness (Fig. [7-11]). That is, the unsteady physical dynamics account for a 50% difference compared to the steady calculation, while the difference between the hydrostatic and non-hydrostatic is also 50% different. At lower Froude number, the transition where non-hydrostatic physics are important occur at a larger criticality. This shows that increasing the Froude number at higher criticality also increases the influence of non-hydrostatic physics on the biological-physical coupling. Thus, for super-critical flows, non-hydrostatic physics have a significant effect on biological productivity and on biological fields and dynamics.

While the biological concentrations and spatially varying details of the coupled physical-biogeochemical system are affected by the non-hydrostatic physics, a bulk or averaged productivity over the bank may not change significantly. In some cases, the enhanced productivity and reduced productivity could even cancel, which means a bulk estimate of the productivity for the region will be unaffected. Then, the hydrostatic simulations could appear sufficient since the larger-scale averaged productivity behavior is captured. However, over time, the productivity variability, however small, will lead to variations in the total concentrations and the spatial features. In a more realistic setting than modeled here, the concentration changes and the spatially variable fields will interact with other processes. Episodic wind-driven or rotationally-driven circulations are likely to interact with the tidally driven fields, amplifying differences between hydrostatic and non-hydrostatic simulations. The biological dynamics, limit-cycles and other nonlinear equilibrium, would also be affected. Therefore, we argue that even for applications where bulk estimates of productivity are sufficient, detailed modeling with non-hydrostatic physics may still be important.

While the magnitude in productivity reaches its maximum amplitude after about 5 tidal cycles for the Fr=1.9, $\epsilon = 1.1$ case (Fig. [7-11]), we note from Fig. [7-13] that the differences in phytoplankton between the non-hydrostatic and hydrostatic runs continue to increase. While larger-scale features remain similar, smaller-scale features, particularly toward the coast, become more different between the hydrostatic and non–hydrostatic simulations. It is generally expected that the hydrostatic dy-

namics will have enhanced phytoplankton concentration due to increased (erroneous) vertical velocities. We see from Fig. [7-16] that the hydrostatic simulation indeed has increased vertical velocities and at finer scales toward the shore. This is an expected behavior because the non-hydrostatic simulations can communicate changes in continuity over larger length–scales than the hydrostatic case. This is because the non-hydrostatic pressure enforces the continuity constraint by changing both the horizontal and vertical velocities, while only the vertical velocity is modified in the hydrostatic case. Therefore, larger horizontal and vertical velocities are expected for the hydrostatic case, and these are observed in Fig. [7-16]–Fig. [7-17]. The small–scale vertical oscillations in the hydrostatic case, then, increase the time-averaged light-intensity field.

However, we do not necessarily observe higher phytoplankton concentration for the hydrostatic case everywhere through the domain at all times. Additionally, away from the coast, we observe higher velocity, small-scale nonlinear internal wave features that are generated at the bank in the non–hydrostatic simulation and propagate away from the coast (Fig. [7-16]). Therefore, at such super-critical flow conditions, the effect on biological productivity of non-hydrostatic physics (compared to hydrostatic physics) are complex, and do not seem amendable to simple rules.

We also note that the phytoplankton fields are nearly identical for lower–criticality Fig. [7-15]. Thus, for sub-critical flows, the effect of non-hydrostatic physics on biological productivity and fields does not seem to be important, at least for the dynamics considered here. For more realistic cases where wind-driven and rotation-driven dynamics are included, differences are likely to be larger, even for sub-critical flows.

Based on our results, future research directions are to further examine the effect of non-hydrostatic physics in three-dimensional settings. For simulations with higher stratification and in realistic conditions, three-dimensional effects will be important. Another direction is to study the coupled effects of tidal forcing with episodic wind-driven forcing, and/or rotational effects. Parameterized studies using reduced order methods (Lermusiaux, 2006, Sapsis and Lermusiaux, 2009, Ueckermann et al., 2012a)

as well as adaptive model learning (Lermusiaux, 2007) for biological predictions would also be useful. Our results indicate that non-hydrostatic effects are important for coupled physical-biological dynamics in coastal regions with steep topography and super-critical flows. Overall, we are now well positioned to extend our idealized studies to more realistic coupled physical–biogeochemical ocean science and societal applications.

Chapter 8

Conclusions and Future work

In this thesis we derived and developed new numerical schemes to solve the Navier-Stokes equations using a hybrid discontinuous Galerkin (HDG) spatial discretization with a projection method. We then derived and developed new high-order HDG schemes for ocean modeling. The corresponding ocean model dynamics can be solved hydrostatically, non-hydrostatically, and with a nonlinear free-surface or a rigid lid. These schemes were implemented in a python framework. We also developed a new quadrature-free implementation which is consistent with hybrid discontinuous Galerkin methods. Then, based on existing low-order slope limiters, we developed a higher-order nodal slope limiter, and combined it with a selectivity criterion to retain higher-order accuracy. We verified the correctness of our new numerical schemes through various benchmark tests, confirming the theoretical convergence rates.

Using our new model codes, we studied the effect of higher-order discretizations on patchy biology, and showed that higher order accuracy is important to retain the concentrations in such patches. It is the high-order accuracy that allows sustaining the correct nonlinear dynamics over a significant duration. To address the issue of when high-order polynomials should be used, we evaluated the numerical truncation error to show which regions of the discretization would benefit from increased polynomial order versus decreased mesh resolution. We argued that where the solution had sharply decaying modal coefficients, the polynomial order should be increased.

We also studied effects of non-hydrostatic physics on biological productivity and

phytoplankton fields over idealized banks, focusing on Stellwagen Bank in Massachusetts Bay. We found that non-hydrostatic physics are the most important when the flow is super-critical, that is, when the topography has a larger slope than the internal wave characteristics. The non-hydrostatic effects increase with increasing nonlinearity, both when the Froude number and criticality parameter increase. With only hydrostatic physics, vertical and baroclinic velocities are often too large and it is the dynamics at smaller scales that is most strongly modified by the non-hydrostatic physics. As a consequence, these smaller scales non-hydrostatic advections alter the biological productivity. Even in cases were the instantaneous biological productivity was not largely modified, we showed that the phytoplankton biomass as well as the phytoplankton spatial variability and patchiness could be significantly altered by non-hydrostatic processes over the idealized bank. For the real Stellwagen Bank, with its wind-driven and rotation-driven dynamics, non-hydrostatic influences are likely to be larger, even for sub-critical flows.

8.1 Discussion and Future Work

During the course of this work, we have identified several area of future research directions, both in numerical research and in multiscale ocean dynamics research. Some of them are discussed next.

8.1.1 Numerical Directions

- Nodal limiter's TVD bound determinations. Currently our nodal limiter calculates the bounds of the solution in an element based on the maximum and minimum values of itself and its immediate neighbors at the nodal points. This does not provide a tight bound for higher-order elements, and localized violation of the total variation diminishing criterion may occur. One improvement is to calculate individual node bounds based on surrounding nodes in the up-wind direction. Furthermore, to improve the accuracy, the bounds may be evaluated by sampling the solution between nodes, although, depending on the number

of sampling locations, this may significantly increase the cost of the method. Sampling between nodes tries to avoid numerical dissipation that often occurs at the apex of sinusoids, for example, where the top of the sine wave is flattened.

- TVD HDG diffusion. Our tests have shown that the numerical HDG diffusion operator can introduce over/under-shoots in the solution when a small diffusivity is applied to a problem with sharp interfaces. Specifically, for the lock-exchange problem §3.8.4, the right-hand-side contributions from the diffusive terms had to be limited to prevent over/under-shoots of the density. Our selective nodal limiter is designed for advective terms with local support, and is not well-suited to diffusive terms. As such, creating TVD-satisfying HDG diffusion operators, or designing an appropriate limiter for the diffusion are areas of potential research.

- Numerical consistency and conservation: free-surface and secondary variables. The vertically integrated 2D continuity constraint that determines the free-surface displacement is not consistent with the 3D continuity constraint that determines the non-hydrostatic pressure or vertical velocity (in the hydrostatic case). This is because in the former case, we integrate the velocity *before* taking its 2D divergence, and the operators do not commute. Additionally, the time integration of the non-hydrostatic pressure and free-surface are not consistent, since the non-hydrostatic pressure is evaluated instantaneously at Runge-Kutta stages, while the free-surface is calculated as a time-integral over multiple Runge-Kutta stages. While the inconsistency of the current scheme does not seem to cause any significant problems, the accuracy and stability of the scheme may be improved by using a fully consistent discretization. The HDG consistency with secondary quantities such as kinetic energy, vorticity or enstrophy as well as the numerical conservation of such quantities is also a major area of research, especially for geophysical dynamics.

- Uncertainty quantification: Ocean prediction is an inherently probabilistic venture. Uncertainties arise from model errors and from unknown initial and

boundary conditions constructed from sparse and noisy measurements. Additionally, uncertainty in the system continues to grow due to the chaotic dynamics. In this thesis, we developed a high-order ocean model to mitigate the effect of uncertainty due to numerical discretization error, but numerical error may remain a source of model uncertainty. As such, to quantify all of the modeling uncertainties, the Dynamically Orthogonal (DO) method (Sapsis and Lermusiaux, 2009, Ueckermann et al., 2012b, Sapsis and Lermusiaux, 2012) can be implemented. When coupled with a non-Gaussian data assimilation method (Sondergaard and Lermusiaux, 2011a,b), this modelling system would be more capable of providing realistic ocean field estimations and predictions.

Additionally, coupling the DO method with high-order discretization could could lead to accurate quantification of numerical discretization errors. The decay of modal polynomial coefficients from a high-order HDG discretization gives an accurate approximation of the local numerical error. Feeding this local error into the DO framework allows this uncertainty to propagate, grow, and dissipate. This area of research could enable the accurate time-space integration of numerical errors, which would allow researchers to assess the accuracy of their predictions, and evaluate how to improve their numerical model.

8.1.2 Multiscale Multi-Dynamics Directions

- Coupled dynamics: In this thesis we studied a simplified coupled physics-biology system §7. This application can be further extended by studying additional parameters. For example, the effect of geometric parameters could be examined. Particularly, the relative shelf/bank slope, the depth of the bank, and the width of the bank could have significant impacts. Additionally, the criticality was calculated using the steepest slope, which was composed of a line. What if the side of the bank had a variable slope (which means the criticality would also be variable)? Would the criticality need to be large enough at one point on the slope, or for a certain percentage of the slope for non-hydrostatic dynamics to

be relevant? Additionally, extending this case to 3D would allow the study of rotational effects, and 3D flow interactions, helping to answer questions such as: Do nonlinear non-hydrostatic processes modify 4D physical exchanges and biogeochemical productivity at banks? Then, the interaction of various forcings, such as the tidal and wind forcing, can also be studied.

- Multi-scale, inner-shelf dynamics: The high-order accuracy of the new code allows simulations of phase resolved larger-scale dynamics. This can then be coupled with a phase-resolved surface wave model to extend the modeling and computational capability for the inner shelf regions. This extension would allow scientific investigations into the dynamics and interactions of the flow domain and wave domain. Resolving the phase of the dynamics is important for nonlinear interactions in regions with complex geometry such as the inner shelf, and with this new capability the issues behind using phase-averaged/spectral descriptions can be examined. This can also lead to improved parameterizations of the wave field. Additionally, the effects of the slowly-varying 3D current and density variations on the surface and internal wave fields can be studied to answer questions such as: Are the interactions of internal tides/waves with background currents affected by non-hydrostatic and nonlinear free-surface processes?. The cross-scale interactions and dynamics in the coupled flow and wave fields could potentially be examined.

- Realistic Regional Ocean Model and High Performance Computing: While the high-order unstructured grid numerical schemes are developed in this thesis, issues such as initialization, grid optimization, objective analysis, and turbulence closures require further consideration before realistic applications can be attempted. In particular, an optimized C++, and MPI parallel implementation (Mirabito et al., 2013), would significantly enhance the utility of the code to handle the resolution required for realistic applications.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Alternate splitting equations based on hybrid discontinuous Galerkin discretizations

Hybrid discontinuous Galerkin methods offer unique possibilities for splitting schemes. Because the full, un-split equations can be cheaply solved on each element once the boundary conditions are known, it is possible to time-split the boundary condition solution instead of the solution of the full equations. That is the basic idea behind the method. One algorithm based on this idea is presented next, but a family of schemes exist.

If we decompose the boundary velocity into normal and tangential components as $\boldsymbol{\lambda} = \lambda_n \hat{\mathbf{n}}_e + \boldsymbol{\lambda}_t$, where the magnitude of the normal component, λ_n, is a scalar in any dimension, the tangential components $\boldsymbol{\lambda}_t = \lambda_t^1 \hat{\mathbf{t}}_e^1 + \lambda_t^2 \hat{\mathbf{t}}_e^2$ will be a vector in 3D, $\hat{\mathbf{n}}_e$ is the unique normal vector as defined on the hybrid discontinuous Galerkin space, and $\hat{\mathbf{t}}_e^1$ and $\hat{\mathbf{t}}_e^2$ are orthogonal tangential vectors. If we take the full hybrid discontinuous Galerkin discretized system that should be solved simultaneously ((2.74), (2.75)) and

we substitute for the normal/tangential decomposed boundary velocity we obtain:

$$
\begin{aligned}
\left((\mathrm{Re})\mathbf{Q}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla \mathbf{v}^{k+1}, \boldsymbol{\Theta}\right)_K + \left\langle \mathbf{v}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} &= \left\langle \boldsymbol{\lambda}_t^{k+1} + \lambda_n^{k+1} \hat{\mathbf{n}}_e, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} \\
\left(\frac{\mathbf{v}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla \cdot \mathbf{Q}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau \mathbf{v}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} + \left(\nabla p^{k+1}, \boldsymbol{\theta}\right)_K &= \left\langle \tau(\boldsymbol{\lambda}_t{}^{k+1} + \lambda_n^{k+1} \hat{\mathbf{n}}_e), \boldsymbol{\theta} \right\rangle_{\partial K} \\
&\quad + \left(\mathbf{F}, \boldsymbol{\theta}\right)_K \\
\left(\nabla \cdot \mathbf{v}^{k+1}, \theta\right)_K - \left\langle \mathbf{v}^{k+1} \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K} &= -\left\langle \lambda_n^{k+1} \hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}}, \theta \right\rangle_{\partial K} \\
\left(p^{k+1}, \frac{1}{|K|}\right)_K &= |p|^{k+1}.
\end{aligned}
$$

$$
\begin{aligned}
\left\langle \left[\!\left[-\mathbf{Q}^{k+1} \cdot \hat{\mathbf{n}} + p^{k+1} \hat{\mathbf{n}} + \tau \left(\mathbf{v}^{k+1} - \boldsymbol{\lambda}_t^{k+1} - \lambda_n^{k+1} \hat{\mathbf{n}}_e \right) \right]\!\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \left\langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon, \\
\left\langle \lambda_n^{k+1} \hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}}, \frac{1}{|\partial K|} \right\rangle_{\partial \mathcal{T}_h} &= 0, \\
\boldsymbol{\lambda}^{k+1}|_{\partial \Omega_D} &= \mathbf{g}_D.
\end{aligned}
$$

We now time-split these equations by initially setting the normal velocity $\lambda_n^{k+1} = \lambda_n^{k\star}$ and averaged pressure $|p|^{k+1} = |p|^{k\star}$ to guessed values. The solution method is then as follows. First solve the local equations

$$
\begin{aligned}
\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1}, \boldsymbol{\Theta}\right)_K - \left(\nabla \bar{\mathbf{v}}^{k+1}, \boldsymbol{\Theta}\right)_K + \left\langle \bar{\mathbf{v}}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} &= \left\langle \boldsymbol{\lambda}_t^{k+1} + \lambda_n^{k\star} \hat{\mathbf{n}}_e, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \right\rangle_{\partial K} \\
\left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - \left(\nabla \cdot \bar{\mathbf{Q}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \tau \bar{\mathbf{v}}^{k+1}, \boldsymbol{\theta} \right\rangle_{\partial K} + \left(\nabla \bar{p}^{k+1}, \boldsymbol{\theta}\right)_K &= \left\langle \tau(\boldsymbol{\lambda}_t{}^{k+1} + \lambda_n^{k\star} \hat{\mathbf{n}}_e), \boldsymbol{\theta} \right\rangle_{\partial K} \\
&\quad + \left(\mathbf{F}, \boldsymbol{\theta}\right)_K \\
\left(\nabla \cdot \bar{\mathbf{v}}^{k+1}, \theta - |\theta|\right)_K - \left\langle \bar{\mathbf{v}}^{k+1} \cdot \hat{\mathbf{n}}, \theta - |\theta| \right\rangle_{\partial K} &= -\left\langle \lambda_n^{k\star} \hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}}, \theta - |\theta| \right\rangle_{\partial K} \\
\left(\bar{p}^{k+1}, \frac{1}{|K|}\right)_K &= |p|^{k\star},
\end{aligned}
$$

where we had to modify the local solver by introducing $|\theta| = \int_K \theta \mathrm{d}K$ to make the equations solvable. These local equations are subject to the global equations

$$
\begin{aligned}
\left\langle \left[\!\left[-\bar{\mathbf{Q}}^{k+1} \cdot \hat{\mathbf{n}} + \bar{p}^{k+1} \hat{\mathbf{n}} + \tau \left(\bar{\mathbf{v}}^{k+1} - \boldsymbol{\lambda}_t^{k+1} \right) \right]\!\right] \cdot \hat{\mathbf{t}}_e, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \left\langle \mathbf{g}_N \cdot \hat{\mathbf{t}}_e, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon \\
\boldsymbol{\lambda}^{k+1}|_{\partial \Omega_D} \cdot \hat{\mathbf{t}}_e &= \mathbf{g}_D \cdot \hat{\mathbf{t}}_e,
\end{aligned}
$$

where $\lambda_n^{k\star}\hat{\mathbf{n}}_e \cdot \hat{\mathbf{t}}_e = 0$. Next, we substitute for the known $\boldsymbol{\lambda}_t^{k+1}$, and solve the next set:

$$
\begin{aligned}
((\mathrm{Re})\mathbf{Q}^{k+1}, \boldsymbol{\Theta})_K - (\nabla \mathbf{v}^{k+1}, \boldsymbol{\Theta})_K + \langle \mathbf{v}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \rangle_{\partial K} &= \langle \boldsymbol{\lambda}_t^{k+1} + \lambda_n^{k+1}\hat{\mathbf{n}}_e, \hat{\mathbf{n}} \cdot \boldsymbol{\Theta} \rangle_{\partial K} \\
\left(\frac{\mathbf{v}^{k+1}}{a\Delta t}, \boldsymbol{\theta}\right)_K - (\nabla \cdot \mathbf{Q}^{k+1}, \boldsymbol{\theta})_K + \langle \tau \mathbf{v}^{k+1}, \boldsymbol{\theta} \rangle_{\partial K} + (\nabla p^{k+1}, \boldsymbol{\theta})_K &= \left\langle \tau(\boldsymbol{\lambda}_t{}^{k+1} + \lambda_n^{k+1}\hat{\mathbf{n}}_e), \boldsymbol{\theta} \right\rangle_{\partial K} \\
&\quad + (\mathbf{F}, \boldsymbol{\theta})_K \\
(\nabla \cdot \mathbf{v}^{k+1}, \theta)_K - \langle \mathbf{v}^{k+1} \cdot \hat{\mathbf{n}}, \theta \rangle_{\partial K} &= -\langle \lambda_n^{k+1}\hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}}, \theta \rangle_{\partial K} \\
\left(p^{k+1}, \frac{1}{|K|}\right)_K &= |p|^{k+1},
\end{aligned}
$$

These local equations are subject to the global equations

$$
\begin{aligned}
\left\langle \left[\!\left[-\mathbf{Q}^{k+1} \cdot \hat{\mathbf{n}} + p^{k+1}\hat{\mathbf{n}} + \tau\left(\mathbf{v}^{k+1} - \lambda_n^{k+1}\hat{\mathbf{n}}_e\right) \right]\!\right] \cdot \hat{\mathbf{n}}_e,\, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon &= \langle \mathbf{g}_N \cdot \hat{\mathbf{n}}_e,\, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon \\
\left\langle \lambda_n^{k+1}\hat{\mathbf{n}}_e \cdot \hat{\mathbf{n}},\, \frac{1}{|\partial K|} \right\rangle_{\partial \mathcal{T}_h} &= 0, \\
\boldsymbol{\lambda}^{k+1}|_{\partial\Omega_D} \cdot \hat{\mathbf{n}}_e &= \mathbf{g}_D \cdot \hat{\mathbf{n}}_e,
\end{aligned}
$$

where $\boldsymbol{\lambda}_t^{k+1} \cdot \hat{\mathbf{n}}_e = 0$. The splitting error can be calculated from noting that the jump of the tangential diffusive fluxes will not longer be zero. That is, the splitting error manifests itself in an inexact diffusive flux conservation. Or, where we had

$$
\left\langle \left[\!\left[-\bar{\mathbf{Q}}^{k+1} \cdot \hat{\mathbf{n}} + \bar{p}^{k+1}\hat{\mathbf{n}} + \tau\left(\bar{\mathbf{v}}^{k+1} - \boldsymbol{\lambda}_t^{k+1}\right) \right]\!\right] \cdot \hat{\mathbf{t}}_e,\, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon = \langle \mathbf{g}_N \cdot \hat{\mathbf{t}}_e,\, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon ,
$$

we now have

$$
\left\langle \left[\!\left[-\mathbf{Q}^{k+1} \cdot \hat{\mathbf{n}} + p^{k+1}\hat{\mathbf{n}} + \tau\left(\mathbf{v}^{k+1} - \boldsymbol{\lambda}_t^{k+1}\right) \right]\!\right] \cdot \hat{\mathbf{t}}_e,\, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon \neq \langle \mathbf{g}_N \cdot \hat{\mathbf{t}}_e,\, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon ,
$$

and the difference of the two is

$$
\left\langle \left[\!\left[(-\mathbf{Q}^{k+1} + \bar{\mathbf{Q}}^{k+1}) \cdot \hat{\mathbf{n}} + (p^{k+1} - \bar{p}^{k+1})\hat{\mathbf{n}} + \tau\left(\mathbf{v}^{k+1} - \bar{\mathbf{v}}^{k+1}\right) \right]\!\right] \cdot \hat{\mathbf{t}}_e,\, \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon = 0.
$$

Now, the size of this splitting error depends both on the time and space discretization. The time discretization error depends on the accuracy of the approximations $\lambda_n^{k\star}$ and $|p|^{k\star}$. For example, if we use the solution at the previous time-step as the guessed

value, then the splitting error above should be $\mathcal{O}(\Delta t)$. For the spatial error, we consider the jump of the splitting error across elements. For the velocity term, both $\mathbf{v}^{k+1} \cdot \hat{\mathbf{t}}$ and $\bar{\mathbf{v}}^{k+1} \cdot \hat{\mathbf{t}}$ are calculated by weakly enforcing the same tangential boundary condition $\boldsymbol{\lambda}_t^{k+1} \cdot \hat{\mathbf{t}}$ on the boundary of the element. While this boundary condition is only enforced weakly, we can still expect $\mathbf{v}^{k+1} \cdot \hat{\mathbf{t}}$ and $\bar{\mathbf{v}}^{k+1} \cdot \hat{\mathbf{t}}$ to be comparable, their total difference of $\mathcal{O}(\Delta x^{\mathrm{p}+1} \Delta t)$ for polynomial of degree p. The difference in the velocities will also be affected by the difference in the pressures $(p^{k+1} - \bar{p}^{k+1})\hat{\mathbf{n}}$, since the pressures and velocities are related. We can expect this error to be similar to the error in the velocities. The error in the diffusive flux, $(-\mathbf{Q}^{k+1} + \bar{\mathbf{Q}}^{k+1}) \cdot \hat{\mathbf{n}}$ is not generally expected to be small, but it will be for high Reynolds numbers, or $\mathcal{O}\left(\frac{\Delta t}{\mathrm{Re}}\right)$. Therefore, we expect the splitting error to be $\mathcal{O}\left(\frac{\Delta t}{\mathrm{Re}}\right) + \mathcal{O}(\Delta x^p \Delta t)$.

Numerical experiments show that the diffusive fluxes are indeed the leading order term in the splitting error. We also see that this scheme results in larger errors than the projection method described in §2.3. As such, we did not further pursue this type of time-splitting algorithm. However, we note that the method described above is one part of a larger family of potential time-splitting methods. For example, instead of using an old value of λ_n in the first step, we could have calculated one by also solving for the normal component of the diffusive-flux. We would have had to solve it again in the next step, but then it would have included a velocity correction similar to that of projection methods. Alternatively, while we solved for the tangential component of the diffusive fluxes first, we could have solved for the normal component instead, followed by the tangential components. Other possibilities also exist; perhaps a least-square approach would improve accuracy. So, even though time did not permit further research in this area, we feel there may be opportunities using methods similar to the one described above. There should be advantages to this approach over projection methods.

One main advantage this method has over the projection method approach is that here we do not have to carefully choose the value of τ_p. Additionally, as discussed in §2.2.3, projection methods have an irreducible splitting error in the tangential velocity components. In particular, we have to correct $\boldsymbol{\lambda}$ on the hybrid discontinuous

Galerkin space for projection methods, while with this method we would assign the correct value of $\boldsymbol{\lambda}$ at boundaries. These advantages make alternative splitting methods based on hybrid discontinuous Galerkin discretizations attractive, but further research is needed.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Alternate Hybrid discontinuous Galerkin formulation of Stokes equations

In this Appendix we present an alternative discretization of the scheme presented in §2.3.2. The main difference between this scheme and the one presented in §2.3.2 is the specification of the pressure–flux.

B.1 HDG discretization

Here we begin by stating the new element–local set of equations, along with their global flux–conservation equations. Following this, we define the HDG fluxes that were used to complete the discretization of (2.58)–(2.61) and obtain these equations.

The element–local equations for $\bar{\mathbf{v}}^{k+1}$, which complete the DG discretization of (2.58) are

$$\begin{aligned}\left((\mathrm{Re})\bar{\mathbf{Q}}^{k+1},\boldsymbol{\Theta}\right)_K - \left(\nabla\bar{\mathbf{v}}^{k+1},\boldsymbol{\Theta}\right)_K + \left\langle\bar{\mathbf{v}}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K} &= \left\langle\bar{\boldsymbol{\lambda}}^{k+1},\hat{\mathbf{n}}\cdot\boldsymbol{\Theta}\right\rangle_{\partial K},\\ \left(\frac{\bar{\mathbf{v}}^{k+1}}{a\Delta t},\boldsymbol{\theta}\right)_K - \left(\nabla\cdot\bar{\mathbf{Q}}^{k+1},\boldsymbol{\theta}\right)_K + \left\langle\tau\bar{\mathbf{v}}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} &= \left\langle\tau\bar{\boldsymbol{\lambda}}^{k+1},\boldsymbol{\theta}\right\rangle_{\partial K} - \left(\mathbf{q}_p^k,\boldsymbol{\theta}\right)_K + \left(\mathbf{F}^{k,k+1},\boldsymbol{\theta}\right)_K,\end{aligned} \tag{B.1}$$

where $\tau = 1$ is the HDG stability parameter (see §2.3.4). The global flux–conservation

equations for $\bar{\boldsymbol{\lambda}}^{k+1}$ are

$$\left\langle \left[\!\left[\widehat{\bar{\mathbf{Q}}}^{k+1} \cdot \hat{\mathbf{n}} \right]\!\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon = \langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon$$

$$\Rightarrow \left\langle \left[\!\left[\bar{\mathbf{Q}}^{k+1} \cdot \hat{\mathbf{n}} - \tau \left(\bar{\mathbf{v}}^{k+1} - \bar{\boldsymbol{\lambda}}^{k+1} \right) \right]\!\right], \boldsymbol{\theta}_\varepsilon \right\rangle_\varepsilon = \langle \mathbf{g}_N, \boldsymbol{\theta}_\varepsilon \rangle_\varepsilon , \tag{B.2}$$

$$\bar{\boldsymbol{\lambda}}|_{\varepsilon_D^\partial}^{k+1} = \mathbf{g}_D \tag{B.3}$$

where $\mathbf{g}_D$ and $\mathbf{g}_N$ are the values of Dirichlet and Neumann boundary conditions for the momentum equations, respectively.

The element–local equations for δp^{k+1}, which complete the DG discretization of (2.59) are unchanged from §2.3.2, and these are

$$\left(\mathbf{q}_{\delta p}^{k+1}, \boldsymbol{\theta}\right)_K - \left(\nabla \delta p^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle \delta p^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K} = \left\langle \lambda_{\delta p}^{k+1}, \hat{\mathbf{n}} \cdot \boldsymbol{\theta} \right\rangle_{\partial K} , \tag{B.4}$$

$$- \left(\nabla \cdot \mathbf{q}_{\delta p}^{k+1}, \theta\right)_K + \left\langle \tau_p \delta p^{k+1}, \theta \right\rangle_{\partial K} = \left\langle \tau_p \lambda_{\delta p}^{k+1}, \theta \right\rangle_{\partial K} - \left(\frac{\nabla \cdot \bar{\mathbf{v}}^{k+1}}{a\Delta t}, \theta \right)_K - \left\langle \frac{(\bar{\boldsymbol{\lambda}}^{k+1} - \bar{\mathbf{v}}^{k+1}) \cdot \hat{\mathbf{n}}}{a\Delta t}, \theta \right\rangle_{\partial K} , \tag{B.5}$$

where we have used $\widehat{\bar{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}$ as defined by (2.83), and $\tau_p = \frac{1}{a\Delta t \tau} \frac{[\![p\hat{\mathbf{n}}]\!]}{[\![\delta p \hat{\mathbf{n}}]\!]}$ is the HDG stability parameter for the pressure–correction (see §2.3.4). The global flux–conservation equation for $\lambda_{\delta p}^{k+1}$ is also unchanged,

$$\left\langle \left[\!\left[\widehat{\mathbf{q}_{\delta p}}^{k+1} \cdot \hat{\mathbf{n}} \right]\!\right], \theta_\varepsilon \right\rangle_\varepsilon = \langle g_{N_p}, \theta_\varepsilon \rangle_\varepsilon$$

$$\Rightarrow \left\langle \left[\!\left[\mathbf{q}_{\delta p}{}^{k+1} \cdot \hat{\mathbf{n}} - \tau_p \left(\delta p^{k+1} - \lambda_{\delta p}^{k+1} \right) \right]\!\right], \theta_\varepsilon \right\rangle_\varepsilon = \langle g_{N_p}, \theta_\varepsilon \rangle_\varepsilon , \tag{B.6}$$

$$\lambda_{\delta p}|_{\varepsilon_D^\partial}^{k+1} = g_{D_p} \tag{B.7}$$

where g_{D_P} and g_{N_P} are the values of Dirichlet and Neumann boundary conditions for the pressure–correction, respectively (and these are often zero Neumann, §2.2.3).

The velocity correction equation (2.60) remains unchanged, but now we use the

pressure–gradient correction equation instead (2.61),

$$\mathbf{v}^{k+1} = \bar{\mathbf{v}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}^{k+1},$$
$$\mathbf{q}_p^{k+1} = \mathbf{q}_p^k + \delta \mathbf{q}_p^{k+1} - \mathbf{r}_{\text{cor}}^{k+1},$$

where the rotational correction term $\mathbf{r}_{\text{cor}}^{k+1} = \nabla \frac{1}{\text{Re}} \cdot \nabla(a\Delta t \mathbf{q}_{\delta p}^{k+1})$ is discretized as

$$\begin{aligned} \left((\text{Re})\mathbf{Q}_{\text{cor}}^{k+1}, \Theta\right)_K &= \left(\nabla \mathbf{v}_{\text{cor}}^{k+1}, \Theta\right)_K + \left\langle \widehat{\mathbf{v}}_{\text{cor}}^{k+1} - \mathbf{v}_{\text{cor}}^{k+1}, \hat{\mathbf{n}} \cdot \Theta \right\rangle_{\partial K}, \\ \left(\mathbf{r}_{\text{cor}}^{k+1}, \boldsymbol{\theta}\right)_K &= \left(\nabla \cdot \mathbf{Q}_{\text{cor}}^{k+1}, \boldsymbol{\theta}\right)_K + \left\langle (\widehat{\mathbf{Q}}_{\text{cor}}^{k+1} - \mathbf{Q}_{\text{cor}}^{k+1}) \cdot \hat{\mathbf{n}}, \boldsymbol{\theta} \right\rangle_{\partial K}, \end{aligned} \tag{B.8}$$

with

$$\mathbf{v}_{\text{cor}}^{k+1} = -a\Delta t \mathbf{q}_{\delta p}^{k+1}, \tag{B.9}$$
$$\mathbf{Q}_{\text{cor}} = \frac{1}{\text{Re}} \nabla \mathbf{v}_{\text{cor}}, \tag{B.10}$$

since $-a\Delta t \mathbf{q}_{\delta p}^{k+1}$ is the velocity-corrector.

The correction on the velocity edge–space also remains unchanged from §2.3.2

$$\boldsymbol{\lambda}^{k+1} = \widehat{\bar{\mathbf{v}}}_{\star}^{k+1} + \widehat{\mathbf{v}}_{\text{cor}}^{k+1} \tag{B.11}$$
$$= \bar{\boldsymbol{\lambda}}^{k+1} - a\Delta t \widehat{\mathbf{q}_{\delta p}}^{k+1} + a\Delta t \tau_p \left(\delta p^{k+1} - \widehat{\delta p}^{k+1}\right) \hat{\mathbf{n}}, \tag{B.12}$$

where (2.73) was obtained by evaluating (2.60) on the edge-space, using the definition of $\widehat{\mathbf{v}}_{\text{cor}}^{k+1} = -a\Delta t \widehat{\mathbf{q}_{\delta p}}^{k+1}$ and assuming that (2.83) was used for $\widehat{\bar{\mathbf{v}}}_{\star}^{k+1}$.

To complete the DG discretization, we needed to define the fluxes, $\widehat{\bar{\mathbf{v}}}^{k+1}$, $\widehat{\bar{\mathbf{v}}}_{\star}^{k+1}$, $\widehat{\delta p}^{k+1}$, $\widehat{\bar{\mathbf{Q}}}^{k+1}$, $\widehat{p}^k$, and $\widehat{\mathbf{q}_{\delta p}}^{k+1}$. We also have yet to define the fluxes for $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$ and $\widehat{\mathbf{Q}}_{\text{cor}}^{k+1}$.

Now, our flux definitions for $\widehat{\bar{\mathbf{v}}}^{k+1}$ in (2.58) at time $k+1$, are:

$$\widehat{\bar{\mathbf{v}}}^{k+1} = \begin{cases} \mathsf{P}\mathbf{g}_D, & \text{on } \varepsilon^{\partial} \\ \bar{\boldsymbol{\lambda}}^{k+1}, & \text{on } \varepsilon^{\circ} \end{cases} \tag{B.13}$$

$$\widehat{\bar{\mathbf{Q}}}^{k+1} = \bar{\mathbf{Q}}^{k+1} - \tau \left(\bar{\mathbf{v}}^{k+1} - \widehat{\bar{\mathbf{v}}}^{k+1}\right) \hat{\mathbf{n}}, \tag{B.14}$$

where P is the L^2 projection of the boundary condition $\mathbf{g}_D$ into the same space as $\boldsymbol{\theta}_\varepsilon$. In the same equation, we are now implicitly using

$$\widehat{p}^k = \lambda_p^k. \tag{B.15}$$

The fluxes for the pressure–correction equation, (2.59), are unchanged, and expressed as:

$$\widehat{\delta p}^{k+1} = \begin{cases} \mathsf{P} g_{D_p}, & \text{on } \varepsilon^\partial \\ \lambda_{\delta p}^{k+1}, & \text{on } \varepsilon^\circ \end{cases} \tag{B.16}$$

$$\widehat{\mathbf{q}_{\delta p}}^{k+1} = \mathbf{q}_{\delta p}{}^{k+1} - \tau_p \left(\delta p^{k+1} - \widehat{\delta p}^{k+1} \right) \hat{\mathbf{n}}, \tag{B.17}$$

with

$$\widehat{\widehat{\mathbf{v}}}_\star^{k+1} = \bar{\boldsymbol{\lambda}}^{k+1}. \tag{B.18}$$

The form of the fluxes $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$ and $\widehat{\mathbf{Q}}_{\text{cor}}^{k+1}$ are explained in §B.2. While there is only one sensible choice for $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$, there are various choices for $\widehat{\mathbf{Q}}_{\text{cor}}^{k+1}$. The different choices for $\widehat{\mathbf{Q}}_{\text{cor}}^{k+1}$ are given in §B.2, and here we report the flux that we have used:

$$\begin{aligned} \widehat{\mathbf{v}}_{\text{cor}}^{k+1} &= -a\Delta t \left\{ \left\{ \widehat{\mathbf{q}_{\delta p}}^{k+1} \right\} \right\} \\ &= -a\Delta t \left\{ \{ \mathbf{q}_{\delta p}^{k+1} \} \right\} + a\Delta t \tau_p \left[\left[\delta p^{k+1} \hat{\mathbf{n}} \right] \right] \end{aligned} \tag{B.19}$$

$$\widehat{\mathbf{Q}}_{\text{cor}}^{k+1} = \mathbf{Q}_{\text{cor}}^{k+1} - \frac{\tau}{2} \left[\left[\mathbf{v}_{\text{cor}}^{k+1} \hat{\mathbf{n}} \right] \right]. \tag{B.20}$$

This completes the specification of the numerical scheme used in this Appendix. In summary, the most significant difference between this scheme and that of §2.3.2 is the choice of the pressure flux.

B.2 Consistency of the gradient of the rotational velocity correction: formal justification

In this section we explain how to apply the rotational correction to the projection method when using the hybrid discontinuous Galerkin method with the fluxes reported in Appendix B. We begin by describing the problem to gain an understanding of the issue, then we present different choices for the fluxes, and we show why calculating the pressure–correction directly does not avoid the problem.

The main issue here is that the rotational (velocity-based) correction of the pressure is in reality an algebraic equation update that should be directly consistent with the pressure update. This quantity should be explicitly calculable from existing quantities, but due to the implicit nature of HDG methods, a globally coupled solve is required for a fully consistent scheme. The rotational correction originates from the diffusion operator applied to the non–divergent component of the predictor velocity, or $-a\Delta t\mathbf{q}_{\delta p}^{k+1}$ (see §2.2.1). As such, the same diffusion operator used in the velocity predictor equation, (B.1), including the same flux definitions, and stability parameters (that is, τ) should be used. However, these fluxes are not well–defined, and the problem of constructing consistent fluxes with known quantities becomes clear when we examine the form of the HDG fluxes as expressed by element–local quantities.

The problem: In §2.3.3 we remarked that the corrector velocity used is single–valued only in the normal direction. To see from where the discontinuity arises, consider the velocity correction on the $\bullet^+$ element:

$$\widehat{\mathbf{v}}_{\mathrm{cor}}^{+}/a\Delta t = \left\{-\mathbf{q}_{\delta p}{}^{+} + \tau_p\left(\delta p^+ - \lambda_{\delta p}\right)\hat{\mathbf{n}}^+\right\}.$$

This can be re-written in terms of element–local quantities only by substituting for

$\lambda_{\delta p}$

$$\begin{aligned}
\widehat{\mathbf{v}}^{+}_{\text{cor}}/a\Delta t &= \left\{-{\mathbf{q}_{\delta p}}^{+}\right\} + \tau_p \left(\delta p^{+} - \{\{\delta p\}\} + \frac{1}{2\tau_p}\left[\!\left[\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\right]\!\right]\right)\hat{\mathbf{n}}^{+} \\
&= \left\{-({\mathbf{q}_{\delta p}}^{+}\cdot\hat{\mathbf{n}}^{+})\hat{\mathbf{n}}^{+} - ({\mathbf{q}_{\delta p}}^{+}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+}\right\} + \frac{1}{2}\left[\!\left[\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\right]\!\right]\hat{\mathbf{n}}^{+} + \tau_p\left(\delta p^{+} - \{\{\delta p\}\}\right)\hat{\mathbf{n}}^{+} \\
&= -\{\{\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\}\}\,\hat{\mathbf{n}}^{+} + \frac{\tau_p}{2}\left[\!\left[\delta p\hat{\mathbf{n}}^{+}\right]\!\right] - (\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+}. \qquad \text{(B.21)}
\end{aligned}$$

Now, since $(\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+} \neq (\mathbf{q}^{-}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+}$, this flux is not single–valued. The equation (B.21), shows that the edge-correction fluxes on either side of an edge differ only by the tangential components of pressure-correction gradient.

The situation, however, is exacerbated for the diffusive fluxes. To see this, we proceed as above:

$$\begin{aligned}
\widehat{\mathbf{Q}}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} &= \mathbf{Q}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} - \tau(\mathbf{v}^{+}_{\text{cor}} - \widehat{\mathbf{v}}^{+}_{\text{cor}}) \\
&= \mathbf{Q}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} - a\Delta t\tau\left(-\mathbf{q}^{+}_{\delta p} + \{\{\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\}\}\,\hat{\mathbf{n}}^{+} - \frac{\tau_p}{2}\left[\!\left[\delta p\hat{\mathbf{n}}\right]\!\right] + (\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+}\right) \\
&= \mathbf{Q}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} - a\Delta t\tau\left(-(\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{n}}^{+})\hat{\mathbf{n}}^{+} - (\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+} + \{\{\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\}\}\,\hat{\mathbf{n}}^{+} - \frac{\tau_p}{2}\left[\!\left[\delta p\hat{\mathbf{n}}\right]\!\right] + (\mathbf{q}^{+}_{\delta p}\cdot\hat{\mathbf{t}}^{+})\hat{\mathbf{t}}^{+}\right) \\
&= \mathbf{Q}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} - a\Delta t\tau\left(-\frac{1}{2}\left[\!\left[\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\right]\!\right]\hat{\mathbf{n}}^{+} - \frac{\tau_p}{2}\left[\!\left[\delta p\hat{\mathbf{n}}\right]\!\right]\right) \qquad \text{(B.22)}
\end{aligned}$$

Now, since $\mathbf{Q}^{+}_{\text{cor}}\cdot\hat{\mathbf{n}}^{+} \neq \mathbf{Q}^{-}_{\text{cor}}\cdot\hat{\mathbf{n}}^{-}$, this flux is not single–valued in the normal or tangential directions. The problem, then, is that $\mathbf{Q}_{\text{cor}}$ can not be directly related to $(\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}})\hat{\mathbf{n}}$ and $\delta p\hat{\mathbf{n}}$ defining a new flux that is single valued and consistent with the HDG fluxes chosen for the velocity predictor and pressure corrector equations. Unfortunately, an exact solution requires the solution of a globally coupled equation (§2.5). Thus, we explore approximate solutions.

Flux choices: Let us first discuss the choices for $\widehat{\mathbf{v}}_{\text{cor}}$. We can choose to have either a continuous flux that is single-valued on the HDG edge-space, but a discontinuous flux is also possible. For a continuous $\widehat{\mathbf{v}}_{\text{cor}}$, the hybrid discontinuous Galerkin flux has to be of the form

$$\widehat{\mathbf{v}}_{\text{cor}}/a\Delta t = -\{\{\mathbf{q}_{\delta p}\}\} - b\left[\!\left[\mathbf{q}_{\delta p}\cdot\hat{\mathbf{t}}\right]\!\right]\hat{\mathbf{t}} + \tau_p\left[\!\left[\delta p\hat{\mathbf{n}}\right]\!\right],$$

which is written to resemble an advective flux. From this it becomes more clear that

we could use a purely central, purely upwind, or mixed scheme if we choose

$$
\begin{aligned}
b &= 0, \\
b &= \text{sign}\left(-\mathbf{q}_{\delta p} \cdot \hat{\mathbf{n}} + \tau_p \left(\delta p - \lambda_{\delta p}\right)\right), \\
b &= f\left(\bar{\mathbf{v}}, \mathbf{q}_{\delta p}, \delta p, \lambda_{\delta p}, \tau_p, \hat{\mathbf{n}}\right),
\end{aligned}
$$

respectively. Since the equation in question is of a diffusive nature, we choose to use a central scheme, as there is no preferred direction of propagation for information. This leads to the flux:

$$
\begin{aligned}
\widehat{\mathbf{v}}_{\text{cor}}/a\Delta t &= -\left\{\left\{\mathbf{q}_{\delta p} \cdot \hat{\mathbf{n}}\right\}\right\} \hat{\mathbf{n}} + \tau_p \left[\!\left[\delta p \hat{\mathbf{n}}\right]\!\right] - \left\{\left\{\mathbf{q}_{\delta p} \cdot \hat{\mathbf{t}}\right\}\right\} \hat{\mathbf{t}} \\
&= -\left\{\left\{\mathbf{q}_{\delta p}\right\}\right\} + \tau_p \left[\!\left[\delta p \hat{\mathbf{n}}\right]\!\right]
\end{aligned}
$$

Also, the central scheme most closely resembles the expected hybrid discontinuous Galerkin flux and numerical experimentation seemed to indicate that this choice gives the most accurate and stable solution.

For a discontinuous $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$, the flux can be defined local to each element. In that case, a sensible choice is

$$
\widehat{\mathbf{v}}_{\text{cor}}/a\Delta t = -\mathbf{q}_{\delta p} + \tau_p \left(\delta p - \lambda_{\delta p}\right) \hat{\mathbf{n}},
$$

however other choices also exist. This flux is discontinuous because on the $\bullet^+$ element the $(\mathbf{q}_{\delta p}^+ \cdot \hat{\mathbf{t}}^+)\hat{\mathbf{t}}^+$ term from (B.21) is used, while on the $\bullet^-$ element the $(\mathbf{q}_{\delta p}^- \cdot \hat{\mathbf{t}}^-)\hat{\mathbf{t}}^-$ term is used (as in (B.21)). Our tests showed that this discontinuous flux representation was less stable than a continuous choice.

As a final note, we could add a tangential component to the correction. For example, we could have

$$
\widehat{\mathbf{v}}_{\text{cor}}^+ = -\mathbf{q}_{\delta p}{}^+ + \tau_p \left(\delta p^+ - \lambda_{\delta p}^{k+1}\right) \hat{\mathbf{n}} + \tau_p' \left(\delta p^+ - \lambda_{\delta p}\right) \hat{\mathbf{t}}.
$$

This will modify (B.21) to give:

$$\boldsymbol{\lambda}_{\text{cor}}/a\Delta t = -\left\{\left\{\mathbf{q}_{\delta p}\cdot\hat{\mathbf{n}}\right\}\right\}\hat{\mathbf{n}}^{+} + \tau_p\left[\!\left[\delta p\hat{\mathbf{n}}^{+}\right]\!\right] - \left\{\left\{\mathbf{q}_{\delta p}\cdot\hat{\mathbf{t}}\right\}\right\}\hat{\mathbf{t}}^{+} + \tau_p^{'}\left[\!\left[\delta p\hat{\mathbf{t}}^{+}\right]\!\right]$$

This form will not affect the solution of any global equation, the discrete divergence, or the advection operator, since the tangential terms will be zero when the flux is dotted with the normal. It will only affect the rotational correction of the pressure which is used as a predictor for the pressure at the next time-step.

Next, we have to consider the flux choices for $\widehat{\mathbf{Q}}_{\text{cor}}^{k+1}$. The problem in this case is different from $\widehat{\mathbf{v}}_{\text{cor}}^{k+1}$, where we at least know that the normal component of the flux is continuous for all reasonable choices. The diffusive flux, $\mathbf{Q}_{\text{cor}}^{k+1}$, however, is not expected to be continuous, since we never solved an equation to preserve the flux of, what is effectively, the third derivative of the pressure. As such, enforcing a continuous flux may not be required or desired. With this in mind, we could simply apply the hybrid discontinuous fluxes with local values in each element

$$\widehat{\mathbf{Q}}_{\text{cor}}^{+} = \mathbf{Q}_{\text{cor}}^{+} - \tau\left(\mathbf{v}_{\text{cor}}^{+} - \widehat{\mathbf{v}}_{\text{cor}}\right)\hat{\mathbf{n}}.$$

The only problem with this flux is that the stabilization term will be different on the left and right sides of the element. As such, we propose a small modification

$$\widehat{\mathbf{Q}}_{\text{cor}}^{+} = \mathbf{Q}_{\text{cor}}^{+} - \frac{\tau}{2}\left[\!\left[\mathbf{v}_{\text{cor}}\hat{\mathbf{n}}\right]\!\right].$$

While this flux is still discontinuous, numerical experiments seem to indicate that it is slightly more stable compared to the previous choice.

Another strategy would be to define a completely continuous flux, which takes the usual form for diffusive discontinuous Galerkin fluxes, that is using

$$\widehat{\mathbf{Q}}_{\text{cor}} = \left\{\left\{\mathbf{Q}_{\text{cor}}\right\}\right\} - C_{11}\left[\!\left[\mathbf{v}_{\text{cor}}\hat{\mathbf{n}}\right]\!\right] + \mathbf{C}_{12}\left[\!\left[\mathbf{Q}_{\text{cor}}\cdot\hat{\mathbf{n}}\right]\!\right],$$

,

we have to define $\mathbf{C}_{12}$ and C_{22}. If we use the hybrid discontinuous Galerkin fluxes, $C_{11} = \tau$, $\mathbf{C}_{12} = 0$, but numerical experiments show this flux is much less stable than the discontinuous one above.

In summary, we argued that a wide range of fluxes could be used for calculating the rotational correction. None of them will be completely consistent with the hybrid discontinuous Galerkin method, unless a globally coupled equation is inverted. The approximate fluxes that we use attempt to remain somewhat compatible with hybrid discontinuous Galerkin, but are not continuous across elements. We have tried various different flux choices, but numerical experiments seemed to indicate that the best result is obtained if we solve the element-local equations for $\mathbf{r}_{\text{cor}}$ using the following fluxes (B.19)–(B.20)

$$\begin{aligned}
\widehat{\mathbf{v}}_{\text{cor}} &= -a\Delta t \{\{\mathbf{q}_{\delta p}\}\} + a\Delta t \tau_p [\![\delta p \hat{\mathbf{n}}]\!], \\
\widehat{\mathbf{Q}}^{+}_{\text{cor}} &= \mathbf{Q}^{+}_{\text{cor}} - \frac{\tau}{2} [\![\mathbf{v}_{\text{cor}} \hat{\mathbf{n}}]\!].
\end{aligned}$$

Direct calculation of pressure: Since the rotational correction term is difficult to compute, why not solve for the true pressure correction directly? Unfortunately, this strategy leads to a problem with the velocity correction. Consider the pressure gradient correction equation, and take its divergence:

$$\begin{aligned}
\mathbf{q}_p^{k+1} &= \mathbf{q}_p^k + \mathbf{q}_{\delta p}^{k+1} - \nabla \frac{1}{\text{Re}} \cdot \nabla a\Delta t \mathbf{q}_{\delta p}^{k+1}, \\
\Rightarrow \nabla \cdot (\mathbf{q}_p^{k+1} - \mathbf{q}_p^k) &= \nabla \cdot \mathbf{q}_{\delta p}^{k+1} - \nabla \cdot \nabla \frac{1}{\text{Re}} \cdot \nabla a\Delta t {\mathbf{q}_{\delta p}}^{k+1}, \\
&= \nabla \cdot \mathbf{q}_{\delta p}^{k+1} + \nabla \cdot \nabla \frac{1}{\text{Re}} \cdot \nabla \bar{\mathbf{v}}^{k+1}, \\
&= \frac{1}{a\Delta t} \nabla \cdot \bar{\mathbf{v}}^{k+1} + \nabla \frac{1}{\text{Re}} \cdot \nabla (\nabla \cdot \bar{\mathbf{v}}^{k+1}),
\end{aligned}$$

where the third line is explained by $\nabla \cdot \mathbf{v}^{k+1} = 0$, $\nabla \cdot \bar{\mathbf{v}} = -a\Delta t \nabla \cdot \mathbf{q}_{\delta p}$. Having solved this equation, the pressure is corrected as

$$\mathbf{q}_p^{k+1} = (\mathbf{q}_p^{k+1} - \mathbf{q}_p^k) + \mathbf{q}_p^k.$$

The velocity, correction, unfortunately, requires the solution of an inverse problem

$$\begin{aligned}\mathbf{v}^{k+1} &= \bar{\mathbf{v}}^{k+1} - a\Delta t \mathbf{q}_{\delta p}^{k+1} \\ &= \bar{\mathbf{v}}^{k+1} - a\Delta t \left[\mathbf{I} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla a \Delta t\right]^{-1} (\mathbf{q}_p^{k+1} - \mathbf{q}_p^k),\end{aligned}$$

since $\mathbf{q}_p^{k+1} - \mathbf{q}_p^k = \left[\mathbf{I} - \nabla \frac{1}{\mathrm{Re}} \cdot \nabla a \Delta t\right] \mathbf{q}_{\delta p}^{k+1}$.

Hence, we prefer solving for $\mathbf{q}_{\delta p}$ with fluxes (B.19), (B.20), which only explicit solves.

Bibliography

Agarwal, A. (2009). Statistical Field Estimation and Scale Estimation for Complex Coastal Regions and Archipelagos. S.M. thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering.

Agarwal, A. and Lermusiaux, P. F. J. (2010). Statistical Field Estimation for Complex Coastal Regions and Archipelagos. Ocean Modeling, In prep.

Ahnert, T. and Bärwolff, G. (Under review. 2013). Numerical comparison of hybridized discontinuous galerkin and finite volume methods for incompressible flow. Int. J. Numer. Meth. Fluids.

Anderson, D., McGillicuddy, D., Townsend, D., and Turner, J. (2005). The ecology and oceanography of toxic Alexandrium fundyense blooms in the Gulf of Maine - Preface. Deep-Sea Research Part II-Topical Studies in Oceanography, 52(19-21):2365–2368.

Ascher, U., Ruuth, S., and Spiteri, R. (1997). Implicitexplicit RungeKutta methods for time-dependent partial differential equations. Appl. Numer. Math., 25:151 – 167.

Baptista, A. and Zhang, Y.-L. (2008). SELFE: A semi-implicit Eulerian-Lagrangian finite-element model for cross-scale ocean circulation. Ocean Modelling, 21(3-4):71–96.

Barter, G. E. and Darmofal, D. L. (2007). Shock capturing with higher-order, pde-based artificial viscosity. AIAA paper, 3823:2007.

Barter, G. E. and Darmofal, D. L. (2010). Shock capturing with pde-based artificial viscosity for dgfem: Part i. formulation. Journal of Computational Physics, 229(5):1810–1827.

Beck, M. (2009). Grand Challenges for the Future for Environmental Modeling. White Paper NFS Award# 0630367, NSF's Environmental Observatories Initiatives, Warnell School of Forestry and Natural Resources, University of Georgia, Athens, Georgia 30602-2152.

Bernard, P. E., Remacle, J. F., and Legat, V. (2009). Boundary discretization for high-order discontinuous Galerkin computations of tidal flows around shallow water islands. International Journal For Numerical Methods in Fluids, 59(5):535–557.

Besiktepe, S. T., Lermusiaux, P. F. J., and Robinson, A. R. (2003). Coupled physical and biogeochemical data-driven simulations of Massachusetts Bay in late summer: real-time and postcruise data assimilation. Journal of Marine Systems, 40:171–212.

Blaise, S., Comblen, R., Legat, V., Remacle, J.-F., Deleersnijder, E., and Lambrechts, J. (2010). A discontinuous finite element baroclinic marine model on unstructured prismatic meshes. Ocean Dynamics, 60(6):1371–1393.

Blossey, P. N. and Durran, D. R. (2008). Selective monotonicity preservation in scalar advection. Journal of Computational Physics, 227(10):5160–5183.

Bourgault, D. and Kelley, D. E. (2004). A laterally averaged nonhydrostatic ocean model. Journal of Atmospheric and Oceanic Technology, 21:1910–1924.

Brown, D. L., Cortez, R., and Minion, M. L. (2001). Accurate projection methods for the incompressible navier–stokes equations. Journal of Computational Physics, 168(2):464–499.

Bryan, K. (1969). A Numerical Method for the Study of the Circulation of the World Ocean. Journal of Computational Physics, 4:347–376.

Budgell, W. P., Oliveira, A., and Skogen, M. D. (2007). Scalar advection schemes for ocean modelling on unstructured triangular grids. Ocean Dynamics, 57(4-5):339–361.

Burchard, H., Deleersnijder, E., and Meister, A. (2005). Application of modified Patankar schemes to stiff biogeochemical models for the water column. Ocean Dynamics, 55:326–337.

Burton, L. J. (2009). Modeling Coupled Physics and Biology in Ocean Straits with Application to the San Bernardino Strait in the Philippine Archipelago. S.M. thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering.

Casulli, V. (1999). A semi-implicit finite difference method for non-hydrostatic, free-surface flows. International Journal for Numerical Methods in Fluids, 30(4):425–440.

Casulli, V. and Zanolli, P. (2002). Semi-implicit numerical modeling of nonhydrostatic free-surface flows for environmental problems. Mathematical and computer modelling, 36(9):1131–1149.

Cebeci, T., Shao, J. P., Kafyeke, F., and Laurendeau, E. (2005). Computational Fluid Dynamics for Engineers. Springer, New York, NY.

Chapra, S. C. and Canale, R. P. (2006). Numerical Methods for Engineers. McGraw-Hill Higher Education, Boston, MA, 5^{th} edition.

Chapra, S. C. and Canale, R. P. (2010). Numerical methods for engineers. McGraw-Hill Higher Education.

Chen, C., Huang, H., Beardsley, R. C., Liu, H., Xu, Q., and Cowles, G. (2007). A finite volume numerical approach for coastal ocean circulation studies: Comparisons with finite difference models. Journal of Geophysical Research: Oceans (1978–2012), 112(C3).

Chen, C., Zhao, L., Cowles, G., and Rothschild, B. (2008). Critical issues for circulation modeling of narragansett bay and mount hope bay. Science for Ecosystem-Based Management: Narragansett Bay in the 21st Century, pages 281–300. Desbonnet, A CostaPierce, BA Symposium on State of Science Knowledge of Nutrients in Narragansett Bay NOV, 2004 Block Isl, RI.

Chorin, A. (1968). Numerical solution of the NavierStokes equations. Math. Comput., 22:745–762.

Chung, T. J. (2002). Computational Fluid Dynamics. Cambridge University Press, New York, NY.

Cockburn, B. and Gopalakrishnan, J. (2009). The derivation of hybridizable discontinuous galerkin methods for stokes flow. SIAM J. Numer. Anal., 47(2):1092–1125.

Cockburn, B., Gopalakrishnan, J., and Lazarov, R. (2009a). Unified hybridization of discontinuous galerkin, mixed, and continuous galerkin methods for second order elliptic problems. SIAM J. Numer. Anal., 47(2):1319–1365.

Cockburn, B., Gopalakrishnan, J., Nguyen, N., Peraire, J., and Sayas, F.-J. (2011). Analysis of hdg methods for stokes flow. Mathematics of Computation, 80(274):723–760.

Cockburn, B., Guzmán, J., and Wang, H. (2009b). Superconvergent discontinuous galerkin methods for second-order elliptic problems. Mathematics of Computation, 78(265):1–24.

Cockburn, B., Hou, S., and Shu, C.-W. (1990). The runge-kutta local projection discontinuous galerkin finite element method for conservation laws. iv. the multidimensional case. Mathematics of Computation, 54(190):545–581.

Cockburn, B., Lin, S.-Y., and Shu, C.-W. (1989). Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws iii: one-dimensional systems. Journal of Computational Physics, 84(1):90–113.

Cockburn, B. and Shu, C. (1998a). The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. SIAM Journal of Numerical Analysis, 35(6):2440–2463.

Cockburn, B. and Shu, C.-W. (1989). Tvb runge-kutta local projection discontinuous galerkin finite element method for conservation laws. ii. general framework. Mathematics of Computation, 52(186):411–435.

Cockburn, B. and Shu, C.-W. (1998b). The runge–kutta discontinuous galerkin method for conservation laws v: multidimensional systems. Journal of Computational Physics, 141(2):199–224.

Cockburn, B. and Shu, C.-W. (2001). Runge–kutta discontinuous galerkin methods for convection-dominated problems. Journal of scientific computing, 16(3):173–261.

Cockburn, B. (1998). An introduction to the discontinuous Galerkin method for convection dominated flows. In: Quarteroni, A. (Ed.), Advanced Numerical Approximation of Nonlinear Hyperbolic Equations. Springer-Verlag, New York, NY.

Comblen, R., Blaise, S., Legat, V., Remacle, J.-F., Deleersnijder, E., and Lambrechts, J. (2010a). A discontinuous finite element baroclinic marine model on unstructured prismatic meshes. part ii: implicit/explicit time discretization. Ocean Dynamics, 60:1395–1414.

Comblen, R., Lambrechts, J., Remacle, J.-F., and Legat, V. (2010b). Practical evaluation of five partly discontinuous finite element pairs for the non-conservative shallow water equations. International Journal for Numerical Methods in Fluids, 63(6):701–724.

Cotter, C. and Ham, D. (2011). Numerical wave propagation for the triangular¡ i¿ p¡/i¿ 1¡ sub¿ dg¡/sub¿–¡ i¿ p¡/i¿ 2 finite element pair. Journal of Computational Physics, 230(8):2806–2820.

Cushman-Roisin, B. and Beckers, J. (2011). Introduction to Geophysical Fluid Dynamics: Physical and Numerical Aspects. International Geophysics. Elsevier Science.

Davies, D. R., Wilson, C. R., and Kramer, S. C. (2011). Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics. Geochemistry, Geophysics, Geosystems, 12(6).

de Brye, B., de Brauwere, A., Gourgue, O., Delhez, E. J., and Deleersnijder, E. (2012). Water renewal timescales in the scheldt estuary. Journal of Marine Systems, 94:74–86.

Dedner, A., Klöfkorn, R., Nolte, M., and Ohlberger, M. (2012). Dune-fem: a general purpose discretization toolbox for parallel and adaptive scientific computing. In Advances in DUNE, pages 17–31. Springer.

Deleersnijder, E., Legat, V., and Lermusiaux, P. F. (2010). Multi-scale modelling of coastal, shelf and global ocean dynamics. Ocean Dynamics, 60(6):1357–1359.

Deleersnijder, E. and Lermusiaux, P. F. J. (2008). Multi-scale Modeling: Nested Grid and Unstructured Grid Approaches. Ocean Dynamics, 58(5–6):335–336.

Denaro, F. M. (2003). On the application of the helmholtz–hodge decomposition in projection methods for incompressible flows with general boundary conditions. International Journal for Numerical Methods in Fluids, 43(1):43–69.

Dolbow, J., Khaleel, M., and Mitchell, J. (2004). Multiscale Mathematics Initiative: A Roadmap. Technical Report PNNL-14966, U.S. Department of Energy, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

Doney, S. C., Lima, I., Moore, J. K., Lindsay, K., Behrenfeld, M. J., Mahowald, T. K. W. N., Glovera, D. M., and Takahashi, T. (2009). Skill metrics for confronting global upper ocean ecosystem-biogeochemistry models against field and remote sensing data . Journal of Marine Systems, 76(1-2):95–112.

Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 10(2):112–122.

Dukowicz, J. and Smith, R. (1994). Implicit free-surface method for the Bryan-Cox-Semtner ocean model. Journal of Geophysical Research, 99(C4):7991–8014.

Durran, D. R. (1999). Numerical Methods for Wave Equations in Geophisical Fluid Dynamics/c Dale R. Durran, volume 32. Springer Verlag.

Ernsdorf, T., Schröder, D., Adams, S., Heinemann, G., Timmermann, R., and Danilov, S. (2011). Impact of atmospheric forcing data on simulations of the laptev sea polynya dynamics using the sea-ice ocean model fesom. Journal of Geophysical Research: Oceans (1978–2012), 116(C12).

Evans, M. A., MacIntyre, S., and Kling, G. W. (2008). Internal wave effects on photosynthesis: Experiments, theory, and modeling. Limnology and Oceanography, 53(1):339.

Farrell, P. (2011). The addition of fields on different meshes. Journal of Computational Physics, 230(9):3265–3269.

Farrell, P. and Maddison, J. (2011). Conservative interpolation between volume meshes by local galerkin projection. Computer Methods in Applied Mechanics and Engineering, 200(1):89–100.

Farrell, P., Micheletti, S., and Perotto, S. (2011). An anisotropic zienkiewicz–zhu-type error estimator for 3d applications. International journal for numerical methods in engineering, 85(6):671–692.

Fasham, M. J. R., Ducklow, H. W., and McKelvie, S. M. (1990). A nitrogen-based model of plankton dynamics in the oceanic mixed layer. Journal of Marine Research, 48:591–639(49).

Fennel, W. and Neumann, T. (2004). Introduction to the modelling of marine ecosystems, volume 72 of Elsevier Oceanography Series. Elsevier, New York, NY.

Ferziger, J. H. and Peric, M. (2002). Computational Methods for Fluid Dynamics. Springer, New York, NY, 3^{rd} edition.

Flierl, G. and McGillicuddy, D. J. (2002). Mesoscale and Submesoscale Physcial-Biological Interactions. The Sea, 12:1–74.

Fringer, O., Gerritsen, M., and Street, R. (2006). An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. Ocean Modelling, 14(3):139–173.

Garrett, C. and Kunze, E. (2007). Internal tide generation in the deep ocean. Annu. Rev. Fluid Mech., 39:57–87.

Gaxiola-Castro, G., Álvarez-Borrego, S., Nájera-Martínez, S., and Zirino, A. R. (2002). Internal waves effect on the gulf of california phytoplankton. Ciencias Marinas, 28(3):297–309.

Gerkema, T. and Zimmerman., J. (2008). An introduction to internal waves. Royal NIOZ, Texel. 207pp.

Geuzaine, C. and Remacle, J. F. (2009). Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. International Journal for Numerical Methods in Engineering, 79(11):1309–1331.

Girault, V. and Raviart, P.-A. (1986). Numerical solution of the stokes problem in the primitive variables. In Finite Element Methods for Navier-Stokes Equations, volume 5 of Springer Series in Computational Mathematics, pages 112–192. Springer Berlin Heidelberg.

Gresho, P. and Sani, R. (1987). On pressure boundary conditions for the incompressible NavierStokes equations. Int. J. Numer. Methods Fluids, 7:1111–1145.

Griffies, S., Adcroft, A., Banks, H., Boninng, C., Chassignet, E., Danabasoglu, G., Danilov, S., Deleersnijder, E., Drange, H., England, M., et al. (2010). Problems and prospects in large-scale ocean circulation models. In OceanObs' 09 Conference: Sustained Ocean Observations and Information for Society, volume 2, pages 1–24.

Griffies, S., Böning, C., Bryan, F., Chassignet, E., Gerdes, R., Hasumi, H., Hirst, A., Treguier, A.-M., and Webb, D. (2000). Developments in ocean climate modelling. Ocean Modelling, 2:123–192.

Guermond, J., Minev, P., and Shen, J. (2006). An overview of projection methods for incompressible flows. Comput. Methods Appl. Mech. Engrg., 195:6011–6045.

Haidvogel, D. B. and Beckmann, A. (1999). Numerical ocean circulation modeling. London : Imperial College Press, River Edge, NJ : Distributed by World Scientific Pub.

Haley, P. J. and Lermusiaux, P. F. J. (2010a). Multiscale two-way embedding schemes for free-surface primitive equations in the Multidisciplinary Simulation, Estimation and Assimilation System. Ocean dynamics, 60(6):1497–1537.

Haley, Jr., P. J. and Lermusiaux, P. F. J. (2010b). Multiscale two-way embedding schemes for free-surface primitive-equations in the Multidisciplinary Simulation, Estimation and Assimilation System (MSEAS). Ocean Dynamics, 60:1497–1537.

Haley, P. J. J., Lermusiaux, P. F. J., Robinson, A. R., Leslie, W. G., Logutov, O., Cossarini, G., Liang, X. S., Moreno, P., Ramp, S. R., Doyle, J. D., Bellingham, J., Chavez, F., and Johnston, S. (2009). Forecasting and Reanalysis in the Monterey Bay/California Current Region for the Autonomous Ocean Sampling Network-II Experiment. Special issue on AOSN-II, Deep Sea Research Part II: Topical Studies in Oceanography, 56:127–148.

Hanert, E., Le Roux, D. Y., Legat, V., and Deleersnijder, E. (2004). Advection schemes for unstructured grid ocean modelling. Ocean Modelling, 7(1-2):39–58.

Härtel, C., Meiburg, E., and Necker, F. (2000). Analysis and direct numerical simulation of the flow at a gravity-current head. Part 1. Flow topology and front speed for slip and no-slip boundaries. Journal of Fluid Mechanics, 418:189–212.

Hecht, M. W., Holland, W. R., and Rasch, P. J. (1995). Upwind-Weighted Advection Schemes for Ocean Tracer Transport - an Evaluation in a Passive Tracer Context . Journal Of Geophysical Research-Oceans, 100(C10):20763–20778.

Hesthaven, J. and Kirby, R. (2008). Filtering in Legendre spectral methods. Mathematics of Computation, 77(263):1425–1452.

Hesthaven, J. and Warburton, T. (2008). Nodal Discontinuous Galerkin Methods, volume 54 of Texts in Applied Mathematics. Springer, New York, NY.

Hofmann, E. E. and Friedrichs, M. A. M. (2002). Predictive modeling for marine ecosystems. The Sea, 12:537–565.

Hofmann, E. E. and Lascara, C. M. (1998). Overview of interdisciplinary modelling for marine ecosystems. The Sea, 10:507–540. Brink, K. H. and A. R. Robinson Eds.

Holloway, G. (1984). Effects of velocity fluctuations on vertical distributions of phytoplankton. Journal of marine research, 42(3):559–571.

Hoteit, H., Ackerer, P., Mos, R., Erhel, J., and Philippe, B. (2004). New two-dimensional slope limiters for discontinuous Galerkin methods on arbitrary meshes. International Journal for Numerical Methods in Engineering, 61(14):25662593.

Huerta, A., Casoni, E., and Peraire, J. (2012). A simple shock-capturing technique for high-order discontinuous Galerkin methods. Int. J. Numer. Meth. Fluids, 69(10):1614–1632.

Huynh, L., Nguyen, N., Peraire, J., and Khoo, B. (2013). A high-order hybridizable discontinuous galerkin method for elliptic interface problems. International Journal for Numerical Methods in Engineering, 93(2):183–200.

Iskandarani, M., Levin, J. C., Choi, B. J., and Haidvogel, D. B. (2005). Comparison of advection schemes for high-order h-p finite element and finite volume methods. Ocean Modelling, 10(1-2):233–252.

Jachec, S. M., Fringer, O. B., Street, R. L., and Gerritsen, M. G. (2007). Effects of grid resolution on the simulation of internal tides. International Journal of Offshore and Polar Engineering, 17(2):105–111.

Ji, R., Davis, C., Chen, C., and Beardsley, R. (2008). Influence of local and external processes on the annual nitrogen cycle and primary productivity on Georges Bank: A 3-D biological-physical modeling study. Journal of Marine Systems, 73(1-2):31 – 47.

Kärnä, T., Legat, V., and Deleersnijder, E. (2012a). A baroclinic discontinuous galerkin finite element model for coastal flows. Ocean Modelling.

Kärnä, T., Legat, V., Deleersnijder, E., and Burchard, H. (2012b). Coupling of a discontinuous galerkin finite element marine model with a finite difference turbulence closure model. Ocean Modelling, 47:55–64.

Karniadakis, G. E. and Sherwin, S. J. (2005). Spectral/hp Element Methods for CFD. Oxford University Press, New York, NY, 2^{nd} edition.

Kelly, S. M., Nash, J. D., and Kunze, E. (2010). Internal-tide energy over topography. J. Geophys. Res., page doi:10.1029/2009JC005618.

Kennedy, C. and Carpenter, M. (2003). Additive RungeKutta schemes for convectiondiffusionreaction equations. Appl. Numer. Math., 44:139 – 181.

Kirby, R. M., Sherwin, S. J., and Cockburn, B. (2012). To cg or to hdg: a comparative study. Journal of Scientific Computing, 51(1):183–212.

Kramer, S., Cotter, C., and Pain, C. (2010). Solving the poisson equation on small aspect ratio domains using unstructured meshes. Ocean Modelling, 35(3):253–263.

Krivodonova, L. (2007). Limiters for high-order discontinuous Galerkin methods. Journal of Computational Physics, 226(1):276–296.

Kubatko, E. J., Bunya, S., Dawson, C., Westerink, J. J., and Mirabito, C. (2009). A Performance Comparison of Continuous and Discontinuous Finite Element Shallow Water Models. Journal of Scientific Computing, 40:315–339.

Kushnir, V., Tokarev, Y., Williams, R., Piontkovski, S., and Evstigneev, P. (1997). Spatial heterogeneity of the bioluminescence field of the tropical atlantic ocean and its relationship with internal waves. Marine Ecology-Progress Series, 160:1–11.

Lai, Z., Chen, C., Beardsley, R. C., Rothschild, B., and Tian, R. (2010). Impact of high-frequency nonlinear internal waves on plankton dynamics in massachusetts bay. Journal of Marine Research, 68(2):259.

Lalli, C. and Parsons, T. (1997). Biological Oceanography: An Introduction. Butterworth-Heinemann, 2^{nd} edition.

Lane, E. M. and Walters, R. A. (2009). Verification of RiCOM for Storm Surge Forecasting. Marine Geodesy, 32:118–132.

Langtangen, H. (2009). Python Scripting for Computational Science. Texts in Computational Science and Engineering. Springer.

Lennert-Cody, C. E. and Franks, P. J. (2002). Fluorescence patches in high-frequency internal waves. Marine Ecology Progress Series, 235:29–42.

Lermusiaux, P. F. J. (1997). Error subspace data assimilation methods for ocean field estimation: theory, validation and applications. PhD thesis, Harvard University.

Lermusiaux, P. F. J. (1999). Data Assimilation via Error Subspace Statistical Estimation, Part II: Mid-Atlantic Bight Shelfbreak Front Simulations, and ESSE Validation. Monthly Weather Review, 127(8):1408–1432.

Lermusiaux, P. F. J. (2001). Evolving the subspace of the three-dimensional multiscale ocean variability: Massachusetts bay. Journal of Marine Systems, 29:385–422.

Lermusiaux, P. F. J. (2006). Uncertainty Estimation and Prediction for Interdisciplinary Ocean Dynamics. Journal of Computational Physics, pages 176–199. Special issue on "Uncertainty Quantification." J. Glimm and G. Karniadakis, Eds.

Lermusiaux, P. F. J. (2007). Adaptive Modeling, Adaptive Data Assimilation and Adaptive Sampling. Physica D, 230:172–196. Special issue on "Mathematical Issues and Challenges in Data Assimilation for Geophysical Systems: Interdisciplinary Perspectives." C.K.R.T. Jones and K. Ide, Eds.

Lermusiaux, P. F. J., Haley, Jr., P. J., Leslie, W. G., and Logutov, O. G. (2009). Philippines Strait Dynamics Experiment - MSEAS Home Page. http://mseas.mit.edu/Research/Straits/.

Lermusiaux, P. F. J. and Xu, J. (2010). Coupled Ocean-Acoustic prediction of transmission loss in a continental shelfbreak region: predictive skill, uncertainty quantification and dynamical sensitivities. IEEE Transactions, Journal of Oceanic Engineering, In press.

Lévy, M., Estublier, A., and Madec, G. (2001). Choice of an Advection Scheme for Biogeochemical Models. Geophysical Research Letters, 28(19):3725–3728.

Liu, W., Chen, W., Cheng, R., and Hsu, M. (2008). Modelling the impact of wind stress and river discharge on Danshuei River plume. Applied Mathematical Modelling, 32(7):1255–1280.

Logutov, O. G. and Lermusiaux, P. F. J. (2008). Inverse Barotropic Tidal Estimation for Regional Ocean Applications . Ocean Modelling, 25:17–34.

Lomax, H., Pulliam, T. H., and Zingg, D. W. (2003). Fundamentals of Computational Fluid Dynamics (Scientific Computation). Springer, New York, NY.

Lynch, D. R., McGillicuddy, D. J., and Werner, F. E. (2009). Skill assessment for coupled biological/physical models of marine systems. Journal of Marine Systems, 76(1-2):1–3.

MacIntyre, S. and Jellison, R. (2001). Nutrient fluxes from upwelling and enhanced turbulence at the top of the pycnocline in mono lake, california. Hydrobiologia, 466(1-3):13–29.

Maddison, J., Cotter, C., and Farrell, P. (2011a). Geostrophic balance preserving interpolation in mesh adaptive linearised shallow-water ocean modelling. Ocean Modelling, 37(1):35–48.

Maddison, J., Marshall, D., Pain, C., and Piggott, M. (2011b). Accurate representation of geostrophic and hydrostatic balance in unstructured mesh finite element ocean modelling. Ocean Modelling, 39(3):248–261.

Markall, G. R., Ham, D. A., and Kelly, P. H. (2010a). Generating optimised finite element solvers for gpu architectures. In AIP Conference Proceedings, volume 1281, page 787.

Markall, G. R., Ham, D. A., and Kelly, P. H. (2010b). Towards generating optimised finite element solvers for gpus from high-level specifications. Procedia Computer Science, 1(1):1815–1823.

Marshall, J., Adcroft, A., Hill, C., Perelman, L., and Heisey, C. (1997). A finite-volume, incompressible navier stokes model for studies of the ocean on parallel computers. Journal of Geophysical Research: Oceans (1978–2012), 102(C3):5753–5766.

Mavriplis, C. A. (1989). Nonconforming Discretization and a Posteriori Error Estimators for Adaptive Spectral Element Techniques. Ph.D. Thesis, MIT.

McCalpin, J. D. (1995). A survey of memory bandwidth and machine balance in current high performance computers. IEEE TCCA Newsletter, pages 19–25.

Michoski, C., Mirabito, C., Dawson, C., Wirasaet, D., Kubatko, E., and Westerink, J. (2011). Adaptive hierarchic transformations for dynamically¡ i¿ p¡/i¿-enriched slope-limiting over discontinuous galerkin systems of generalized equations. Journal of Computational Physics, 230(22):8028–8056.

Mirabito, C., Ueckermann, M., Haley, P., Lermusiaux, P., and Jang, D. (2013). Parallelization of hybridizable discontinuous galerkin methods for a nonhydrostatic free surface primitive equation ocean model. PDF slides. SIAM CSE13 Conference, Boston.

MSEAS-Group (2010). Multidisciplinary Simulation, Estimation, and Assimilation Systems (http://mseas.mit.edu/ , http://mseas.mit.edu/codes). Reports in Ocean Science and Engineering 6, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Nguyen, N., Peraire, J., and Cockburn, B. (2009a). An implicit high-order hybridizable discontinuous Galerkin method for linear convection-diffusion equations. Journal of Computational Physics, 228(9):3232–3254.

Nguyen, N., Peraire, J., and Cockburn, B. (2009b). An implicit high-order hybridizable discontinuous galerkin method for nonlinear convection–diffusion equations. Journal of Computational Physics, 228(23):8841–8855.

Nguyen, N., Peraire, J., and Cockburn, B. (2010a). A hybridizable discontinuous galerkin method for stokes flow. Computer Methods in Applied Mechanics and Engineering, 199(9):582–597.

Nguyen, N., Peraire, J., and Cockburn, B. (2010b). A hybridizable discontinuous galerkin method for the incompressible navier-stokes equations. In Proceedings of the 48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida.

Nguyen, N., Peraire, J., and Cockburn, B. (2011). An implicit high-order hybridizable discontinuous galerkin method for the incompressible navier–stokes equations. Journal of Computational Physics, 230(4):1147–1170.

Nguyen, N., Persson, P.-O., and J., P. (2007). Rans solutions using high order discontinuous galerkin methods. AIAA Paper, 914.

Nguyen, N., Roca, X., Moro, D., and Peraire, J. (2013). A hybridized multiscale discontinuous galerkin method for compressible flows. AIAA Paper, 689.

Oreskes, N., Shrader-Frechette, K., Belitz, K., et al. (1994). Verification, validation, and confirmation of numerical models in the earth sciences. Science, 263(5147):641–646.

Oschlies, A. and Garcon, V. (1998). Eddy-induced enhancement of primary production in a model of the North Atlantic Ocean. Nature, 394:266–269.

Palma, M. E. S. (2012). Hybridizable discontinuous Galerkin method for curved domains. PhD thesis, University of Minnesota.

Pasquero, C., Bracco, A., and Provenzale, A. (2004). Coherent vortices, lagrangian particles and the marine ecosystem. In Jirka and Uijttewaal, editors, Shallow Flows, pages 399–415. Taylor and Francis Group.

Patterson, J. C. (1991). Modelling the effects of motion on primary production in the mixed layer of lakes. Aquatic Sciences, 53(2-3):218–238.

Pedlosky, J. (1987). Geophysical Fluid Dynamics. Springer, New York, NY, 2^{nd} edition.

Peraire, J., Nguyen, N., and Cockburn, B. (2010). A hybridizable discontinuous galerkin method for the compressible euler and navier-stokes equations. AIAA Paper, 363:2010.

Peraire, J., Nguyen, N., and Cockburn, B. (2011). An embedded discontinuous galerkin method for the compressible euler and navier-stokes equations. AIAA Paper, 3228.

Persson, P., Bonet, J., and Peraire, J. (2009). Discontinuous galerkin solution of the navier–stokes equations on deformable domains. Computer Methods in Applied Mechanics and Engineering, 198(17):1585–1595.

Persson, P.-O. and Peraire, J. (2006). Sub-cell shock capturing for Discontinuous Galerkin methods. AIAA paper, 112.

Persson, P. O. and Strang, G. (2004). A Simple Mesh Generator in MATLAB. SIAM Review, 46(2):329–345.

Qiu, J. and Shu, C.-W. (2005). Runge–kutta discontinuous galerkin method using weno limiters. SIAM Journal on Scientific Computing, 26(3):907–929.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing, 1(3):244–256.

Roache, P. J. (1998). Verification and validation in computational science and engineering. Hermosa Albuquerque.

Robinson, A. R., Haley Jr, P. J., Lermusiaux, P. F. J., and Leslie, W. G. (2002a). Predictive skill, predictive capability and predictability in ocean forecasting. In OCEANS'02 MTS/IEEE, volume 2, pages 787–794. IEEE.

Robinson, A. R. and Lermusiaux, P. F. J. (1999). Report of a Workshop on the Assimilation of Biological Data in Coupled Physical/Ecosystem Models. GLOBEC Special Contribution 3, Bologna, Italy.

Robinson, A. R., McCarthy, J. J., and Rothschild, B. J. (2002b). Biological-Physical Interactions in the Sea, volume 12 of The Sea. John Wiley & Sons, Inc., New York, NY.

Rothstein, L. M., Cullen, J. J., Abbott, M., Chassignet, E. P., Denman, K., Doney, S. C., Ducklow, H., Fennel, K., Follows, M., Haidvogel, D., Hoffman, E., Karl, D. M., Kindle, J., Lima, I., Maltrud, M., McClain, C., McGillicuddy, D. J., Olascoaga, M. J., Spitz, Y., Wiggert, J., and Yoder, J. (2006). Modeling Ocean Ecosystems: The Paradigm Program. Oceanography, 19(1):22–51.

Sangrà, P., Basterretxea, G., Pelegrí, J. L., Arístegui, J., et al. (2001). Chlorophyll increase due to internal waves in the shelf-break of gran canaria island (canary islands). Sci. Mar.

Sapsis, T. P. and Lermusiaux, P. F. J. (2009). Dynamically orthogonal field equations for continuous stochastic dynamical systems. Physica D, 238:2347–2360.

Sapsis, T. P. and Lermusiaux, P. F. J. (2012). Dynamical criteria for the evolution of the stochastic dimensionality in flows with uncertainty. Physica D: Nonlinear Phenomena, 241(1):60–76.

Schütz, J. and May, G. (2013). A hybrid mixed method for the compressible navier-stokes equations. Journal of Computational Physics.

Seny, B., Lambrechts, J., Comblen, R., Legat, V., and Remacle, J.-F. (2013). Multirate time stepping for accelerating explicit discontinuous galerkin computations with application to geophysical flows. International Journal for Numerical Methods in Fluids, 71(1):41–64.

Shah, S., Heemink, A., and Deleersnijder, E. (2011). Assessing lagrangian schemes for simulating diffusion on non-flat isopycnal surfaces. Ocean Modelling, 39(3):351–361.

Shirokoff, D. and Rosales, R. (2011). An efficient method for the incompressible NavierStokes equations on irregular domains with no-slip boundary conditions, high order up to the boundary. Journal of Computational Physics, 230(23):8619 – 8646.

Signell, R. (1989). Tidal dynamics and dispersion around coastal headlands. PhD thesis, WHOI/MIT Joint Program.

Slingo, J., Bates, K., Nikiforakis, N., Piggott, M., Roberts, M., Shaffrey, L., Stevens, I., Vidale, P. L., and Weller, H. (2009). Developing the next-generation climate system models: challenges and achievements. Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences, 367(1890):815–831. Discussion Meeting on the Environmental eScience Revolution APR 07-08, 2008 Royal Soc, London, ENGLAND.

Smith, W. H. and Sandwell, D. T. (1997). Global sea floor topography from satellite altimetry and ship depth soundings. Science, 277(5334):1956–1962.

Sondergaard, T. and Lermusiaux, P. F. J. (2011a). Data Assimilation with Gaussian Mixture Models using the Dynamically Orthogonal Field Equations. Part I: Theory and Scheme. Submitted to Mon. Wea. Rev.

Sondergaard, T. and Lermusiaux, P. F. J. (2011b). Data Assimilation with Gaussian Mixture Models using the Dynamically Orthogonal Field Equations. Part II: Applications. Submitted to Mon. Wea. Rev.

Spitz, Y., Allen, J., and Gan, J. (2005). Modeling of ecosystem processes on the oregon shelf during the 2001 summer upwelling. Journal of Geophysical Research, 110:C10S17.

Stow, C. A., Jolliff, J., McGillicuddy, D. J., Doney, S. C., Allen, J. I., Friedrichs, M. A. M., Rose, K. A., and Wallhead, P. (2009). Skill Assessment for Coupled Biological/Physical Models of Marine Systems. Journal of Marine Systems, 76(1-2):4–15.

Strang, G. and Fix, G. J. (1973). An analysis of the finite element method, volume 212. Prentice-Hall Englewood Cliffs, NJ.

Témam, R. (1969). Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (ii). Archive for Rational Mechanics and Analysis, 33:377–385.

Thomas, P. (1979). Geometric conservation law and its application to flow computations on moving grids. AIAA Journal, 17:1030–1037.

Timmermans, L., Minev, P., and Van De Vosse, F. (1996). An approximate projection scheme for incompressible flow using spectral elements. Int. J. Numer. Methods Fluids, 22:673–688.

Ueckermann, M. (2009). Towards Next Generation Ocean Models: Novel Discontinuous Galerkin Schemes for 2D unsteady biogeochemical models. Master's thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering.

Ueckermann, M. P. and Lermusiaux, P. F. J. (2010). High-order schemes for 2D unsteady biogeochemical ocean models. Ocean Dynamics, 60:1415–1445.

Ueckermann, M. P., Lermusiaux, P. F. J., and Sapsis, T. P. (2012a). Numerical schemes for dynamically orthogonal equations of stochastic fluid and ocean flows. Journal of Computational Physics.

Ueckermann, M. P., Lermusiaux, P. F. J., and Sapsis, T. P. (2012b). Numerical schemes for dynamically orthogonal equations of stochastic fluid and ocean flows. Journal of Computational Physics.

Umlauf, L. and Burchard, H. (2003). A generic length-scale equation for geophysical turbulence models. Journal of Marine Research, 61(2):235–265.

Umlauf, L. and Burchard, H. (2005). Second-order turbulence closure models for geophysical boundary layers. a review of recent work. Continental Shelf Research, 25(7):795–827.

Vincent, P. and Jameson, A. (2011). Facilitating the adoption of unstructured high-order methods amongst a wider community of fluid dynamicists. Mathematical Modelling of Natural Phenomena, 6(03):97–140.

Vlasenki, V., Stashchuck, N., and Hutter, K. (2005). Baroclinic Tides: Theoretical Modeling and Observational Evidence. Cambridge University Press. 351pp.

Walters, R. A., Gillibrand, P. A., Bell, R. G., and Lane, E. M. (2010). A study of tides and currents in cook strait, new zealand. Ocean Dynamics, 60(6):1559–1580.

Waluga, C. and Egger, H. (2012). An implementation of hybrid discontinuous galerkin methods in dune. In Advances in DUNE, pages 169–180. Springer.

Wang, Q., Danilov, S., and Schröter, J. (2008). Finite element ocean circulation model based on triangular prismatic elements, with application in studying the effect of topography representation. Journal of Geophysical Research: Oceans (1978–2012), 113(C5).

Wang, Q., Danilov, S., and Schröter, J. (2009). Bottom water formation in the southern Weddell Sea and the influence of submarine ridges: Idealized numerical simulations. Ocean Modelling, 28(1-3):50–59.

Warner, J. C., Sherwood, C. R., Arango, H. G., and Signell, R. P. (2005). Performance of four turbulence closure models implemented using a generic length scale method. Ocean Modelling, 8(1):81–113.

Westerink, J., Luettich, R. J., and Muccino, J. (1994). Modeling tides in the Wester North Atlantic using unstructured graded grids. Tellus, 46(A):178–199.

Westerink, J.J.and Luettich, R. J., Feyen, J.C.and Atkinson, J., Dawson, C., Powell, M., Dunion, J., Roberts, H., Kubatko, E., and Pourtaheri, H. (2007). A Basin to Channel Scale Unstructured Grid Hurricane Storm Surge Model as Implemented for Southern Louisiana. Monthly Weather Review, 136:833–864.

White, L., Legat, V., and Deleersnijder, E. (2008). Tracer conservation for three-dimensional, finite-element, free-surface, ocean modeling on moving prismatic meshes. Monthly weather review, 136(2):420–442.

Wunsch, C., Haidvogel, D. B., Iskandarani, M., and Hughes, R. (1997). Dynamics of the Long-Period Tides. Progress in Oceanography, 40:80–108.

Yang, D., Ye, H., and Wang, G. (2010). Impacts of internal waves on chlorophyll a distribution in the northern portion of the south china sea. Chinese Journal of Oceanology and Limnology, 28(5):1095.

Zhu, J., Qiu, J., Shu, C.-W., and Dumbser, M. (2008). Runge–kutta discontinuous galerkin method using weno limiters ii: unstructured meshes. Journal of Computational Physics, 227(9):4330–4353.