THE STRATIFIED OCEAN MODEL WITH ADAPTIVE REFINEMENT (SOMAR)

Edward Santilli

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Physics.

Chapel Hill
2015

Approved by:

Alberto Scotti

Charles Evans

Fabian Heitsch

Reyco Henning

Sutanu Sarkar

ABSTRACT

Edward Santilli: The Stratified Ocean Model with Adaptive Refinement (SOMAR)
(Under the direction of Alberto Scotti)

A computational framework for the evolution of non-hydrostatic, baroclinic flows encountered in regional and coastal ocean simulations is presented, which combines the flexibility of Adaptive Mesh Refinement (AMR) with a suite of numerical tools specifically developed to deal with the high degree of anisotropy of oceanic flows and their attendant numerical challenges. This framework introduces a semi-implicit update of the terms that give rise to buoyancy oscillations, which permits a stable integration of the Navier-Stokes equations when a background density stratification is present. The *lepticity* of each grid in the AMR hierarchy, which serves as a useful metric for anisotropy, is used to select one of several different efficient Poisson-solving techniques. In this way, we compute the pressure over the entire set of AMR grids without resorting to the hydrostatic approximation, which can degrade the structure of internal waves whose dynamics may have large-scale significance. We apply the modeling framework to three test cases, for which numerical or analytical solutions are known that can be used to benchmark the results. In all the cases considered, the model achieves an excellent degree of congruence with the benchmark, while at the same time achieving a substantial reduction of the computational resources needed.

ACKNOWLEDGEMENTS

This work was supported by the ONR under grants N00014-05-1-0361 and N00014-09-1-0288 and by the NSF under grants OCE-0825997 and OCE-0729636. Computer resources were provided by the Information Technology Services Research Computing group.

I would like to personally thank Sutanu Sarkar, Vamsi Chalamalla, Masoud Jalali, and Narsimha Rapaka of the UCSD School of Engineering for their correspondence and valuable insight throughout the development and testing of SOMAR.

Finally, and most of all, I would like to express gratitude towards my advisor, Alberto Scotti, whose direction, patience, and deep understanding of all things numerical and physical made this work possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

AMR	Adaptive Mesh Refinement
BiCGStab	Stabilized Bi-Conjugate Gradient Method
CPU	Central Processing Unit
DNS	Direct Numerical Simulation
GCM	Global Circulation Model
GFD	Geophysical Fluid Dynamics
KdV	Korteweg-de Vries
LES	Large Eddy Simulation
LHS	Left-Hand Side
MG	Multigrid
MITgcm	Massachussetts Institute of Technology General Circulation Model
MPI	Message Passing Interface
NLIW	Nonlinear Internal Wave
NSE	Navier-Stokes Equations
PDE	Partial Differential Equation
PETSc	Portable, Extensible Toolkit for Scientific Computation
PPM	Piecewise Parabolic Method
RHS	Right-Hand Side
RMS	Root-Mean-Square
ROMS	Regional Ocean Modeling System
SUNTANS	The Stanford Unstructured-Grid, Nonhydrostatic, Parallel Coastal Ocean Model
UNC	University of North Carolina
b_T	The total buoyancy (reduced gravity)
$\bar{b}$	The background buoyancy profile (This is a function of z alone.)
b	The buoyancy deviation
u^α	Cartesian component of the 3-dimensional vector $\vec{u}$

w	Vertical Cartesian velocity component
u^i	Curvilinear component of the 3-dimensional vector $\vec{u}$
$u^{n,i}$	u^i at time t^n (n will never be used as an index.)
$u^{\star,i}$	An unprojected velocity component
u^i_{AD}	The time and face centered, projected advecting velocity
u^i_H	Interpolation of u^i_{AD} to all cell faces
ν, κ	Kinematic viscosity and scalar diffusion parameters
$\{x, y, z\}$ or $\{x^\alpha\}$	Cartesian coordinates
$\{\xi, \eta, \zeta\}$ or $\{\xi^i\}$	General curvilinear coordinates
g_{ij}, g^{ij}	The metric tensor and its inverse
J	The Jacobian determinant
CC	Cell centered.
FC	Face centered.
MAC	Marker-and-cell scheme that describes the exact, discrete projection method
ϕ	Scalar used during the MAC projection
π^l	Pressure computed during the level l, CC velocity projection
$\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}$	Averaging operator that sends CC data to FC data
$\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}$	Averaging operator that sends FC data to CC data
ϵ	Average rate of dissipation of turbulence kinetic energy per unit mass
ϵ_{ijk}	The totally anti-symmetric permutation symbol
ε_{ijk}	The covariant Levi-Civita tensor ($J\epsilon_{ijk}$)
λ	The grid lepticity, or leptic ratio

CHAPTER 1: Introduction

Section 1.1: Motivation

The numerical simulation of geophysical and astrophysical flows presents several challenges. One major difficulty is that the flows are often characterized by a marked anisotropy due to a combination of geometrical and dynamical constraints. In geophysical flows, geometrical constraints cause the horizontal scales to be much larger than the vertical scale, while dynamical constraints, such as gravity, rotation, and magnetism if the fluid is conducting, introduce a preferred direction, which severely constrains the flow at very large scales [4, 5, 6]. In the following, we will concentrate on oceanic applications, though the techniques presented in this manuscript could also be applied to atmospheric and, to a lesser extent, astrophysical flows. In the ocean, temporal scales range from the millennial time scale of the Meridional Overturning Circulation [7] to seconds in the case of small scale turbulence [8], though the type of processes for which we are developing the model discussed in this manuscript span the smaller range from days (internal tides) to seconds (turbulence).

Clearly, a unified modeling framework is out of reach for the foreseeable future. Rather, a set of tools has emerged since the pioneering work of Bryan in the late 1960's [9]. By the early 2000s, a well established suite of models existed covering either the very large scales or the very small scales. For the case of large scales, provided the horizontal scale of the motion remains much larger than the local depth, the pressure is found to satisfy to leading order the hydrostatic approximation. This greatly simplifies the equations, and around this core approximation a number of General Circulation Models (GCMs) have been developed differing among themselves based on the choice of dependent and independent variables, the type of discretization, and so on. For more details, and also for a historical perspective, the reader is referred to Haidvogel and Beckmann [6]. Due to the nonlinear nature of the underlying dynamics, the unresolved scales do have a nontrivial effect on the resolved scales. Their effect is usually included with parameterization schemes which range in complexity from simple constant "eddy" viscosity and diffusion, to more sophisticated closure schemes [10]. At the opposite end of much smaller scales, the hydrostatic approximation obviously breaks down, but the flow is also much less anisotropic and rotation is negligible. This dynamical regime can be efficiently dealt with Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) [11, 12], which developed rather independently and parallel to the development of the large scale GCMs.

At the same time, progress in observational techniques uncovered a host of processes that occur on

Study site	Code	Lepticity	Reference
South China Sea	SUNTANS	7	Zhang, *et. al.* [16]
Monterey Bay	SUNTANS	> 4	Kang, *et. al.* [17]
Monterey Bay	SUNTANS	> 14	Jachec, *et. al.* [18]
Luzon Strait	MITgcm	3.1	Guo, *et. al.* [19]
Andaman Sea	MITgcm	3.3	Vlasenko, *et. al.* [20]
Gibraltar Strait	MITgcm	0.7	Vlasenko, *et. al.* [21]
Cayuga Lake	MITgcm	1.2	Dorostkar, *et. al.* [22]

Table 1.1: Three-dimensional nonhydrostatic simulations. The Monterey Bay simulations where not designed with NLIWs in mind, but are included as they include the effect of complex topography on internal waves. To properly account for the physical (as opposed to numerical) sources of dispersion in the propagation of NLIWs, the grid lepticity must be less than one. Vitousek and Fringer [23] define the lepticity as the ratio of the horizontal grid spacing to the depth of the internal interface. Table adapted from Dorostkar [22].

intermediate scales – between the large scales covered by GCMs and the small scales covered by DNS/LES. Of particular interest here, since it provided the initial motivation to develop the modeling framework discussed in the rest of this paper, was the discovery that nonlinear internal waves are an ubiquitous feature of the coastal environment [13]. The dynamics of these waves are still characterized by a marked anisotropy, but the hydrostatic approximation needs to be abandoned and replaced by a full three-dimensional Poisson problem to determine the pressure. To fill the gap, new GCMs were developed that included the possibility of calculating the pressure from the full incompressibility condition, rather than relying on hydrostasy. Among the most widely used we find the MITgcm [14] and SUNTANS [15], which differ on the way the domain is discretized, but essentially solve the same equations. These models are very general, in that they can be run in a GCM mode (hydrostatic) all the way down to DNS. However, such generality comes at a price. The time required to solve the Poisson problem relative to the overall time per time step increases rapidly as the anisotropy of the domain increases, to the point where the solution of the Poisson problem becomes the bottleneck. As a result, most of the published results which involve the models run in their non-hydrostatic mode over high-aspect ratio have geophysical scales that restrict the dynamics to two-dimensional vertical sections with some notable exceptions (see table 1.1). The difficulties arise from a combination of two factors: The number of grid points increases substantially relative to a hydrostatic simulation and the convergence rate of standard iterative Poisson solvers decreases with increasing aspect ratio. To get an understanding of the first issue, in a GCM the number of grid points along a horizontal direction is $O(L_x/H)$, where L_x is the extent of the domain in that direction and H the local depth. To properly describe the physics of nonlinear internal waves, the horizontal grid size has to be an order of magnitude smaller than the total depth, i.e., $\Delta x \sim H/10$ [23]. For a three dimensional simulation, the number of grid points thus increases by at least an order of magnitude, but since the grid cannot always be aligned with the direction of propagation, in practice the increase can be as large as two orders of magnitude. Note that nonlinear internal waves and related

phenomena are usually highly localized, and in an appropriate sense not far from being hydrostatic (though the departure is essential to their dynamics). Improvement on the current status thus revolves in developing a framework that takes advantage of these properties to effectively manage the numerical resources and improve the convergence rate of the iterative solver in high aspect ratio conditions.

A description of such framework is the object of the remainder of this thesis. It combines the flexibility of Adaptive Mesh Refinement (AMR) [24] to effectively husband finite numerical resources with the Leptic Poisson Solver [25] to bring the convergence rate of the Poisson solver within acceptable limits.

Section 1.2: Equations of Motion

Before we delve into numerical algorithms, it is worthwhile to define the equations of motion that we wish to solve. A description will be presented in terms of the underlying physical principles, accounting for and justifying the approximations we use along the way.

1.2.1: The continuum hypothesis

The smallest length scales of dynamical importance in a fluid are the Kolmogorov scale, where molecular viscosity converts kinetic energy into heat, and the Batchelor scale, where diffusive time scales rival inertial time scales [26]. In the ocean, these scales can be as small as $\mathcal{O}(1\text{mm})$ [27, Sec. 2.2]. Far smaller, is the mean-free-path of the individual molecules of the fluid, which at standard temperature and pressure is less than $\mathcal{O}(1\text{nm})$ [28, Sec. 1.2.1]. With 6 orders of magnitude between these scales, we can, to a great degree of accuracy, ignore the molecular structure of the ocean. In doing so, we assume that the fluid is a continuum rather than a collection of discrete particles. This *continuum hypothesis* enables us to convert integral formulations of the dynamics into differential formulations. In this section, we will methodically derive the differential equations of motion that contain the physics of our ocean model.

Central to the idea of the continuum hypothesis is that of a *fluid parcel*. A fluid parcel is a volume of fluid that is smaller than any dynamically important feature, but large enough neglect its molecular structure. Fluid parcels can be tracked as they are forced to move about the domain, and although the parcel may become distorted over time, no fluid is allowed to pass through its boundary. In other words, by definition of the parcel's boundary, the total mass of a fluid parcel is a constant in time. In what follows, we will identify an arbitrary fluid parcel in the domain of interest by $\mathcal{V}(t)$, and its boundary by $\partial\mathcal{V}(t)$.

1.2.2: The Navier-Stokes Equations

Strictly speaking, the Navier-Stokes equations are a system of equations that dictate the evolution of momentum for a fluid whose stresses can be expressed as $-g^{ij}P + \mu(\nabla^i u^j + \nabla^j u^i) + \lambda g^{ij}\nabla \cdot \vec{u}$, where μ and λ parameterize the stresses due to shear and compression of a fluid parcel [29, Sec. 38]. When dealing with a stratified fluid, the momentum equation is joined with an evolution equation for density as well. For our purposes, the Navier-Stokes equations will refer to the complete system of equations that are solved to determine the state of the fluid. In this section, we will therefore derive both a mass transport equation and a momentum conservation law. In subsequent sections, we will focus on applying and justifying approximations that tailor the Navier-Stokes equations to our stratified oceanic needs.

Conservation of Mass

We begin by defining the mass of a fluid parcel, $\mathcal{V}(t)$, in terms of the fluid's density field,

$$m = \int_{\mathcal{V}(t)} \rho(\vec{x},t)\, dV. \tag{1.1}$$

In general, the total mass of a fluid need not be conserved. In the ocean, sunlight penetrates approximately 200 meters into the surface, changing the chemical and biological makeup of the fluid [30, 31]. Evaporation and precipitation, which tend to be locally out of balance, can add or remove mass to the surface, resulting in horizontal displacements of fluid to restore balance with the atmospheric pressure [32, Sec. 9.9, 9.14, 11.14]. Additionally, evaporation plays a significant role in removing heat energy and increasing salt concentrations at the ocean's surface [32, Sec. 2.6, 2.7]. For our purposes, we will assume that the mass inside the domain is, in fact, conserved and that any mediating factors can be handled by an appropriate equation of state or through a judicious choice of boundary conditions. To conserve total mass, we can fix the parcel boundary in time and assume that any change in local mass can only arise through a mass flux though the boundary $\partial\mathcal{V}(t_0)$. That is,

$$\int_{\mathcal{V}(t_0)} \frac{\partial \rho(\vec{x},t)}{\partial t}\, dV = -\oint_{\partial\mathcal{V}(t_0)} [\vec{u}(\vec{x},t)\,\rho(\vec{x},t)] \cdot d\vec{A} \tag{1.2a}$$

$$= -\int_{\mathcal{V}(t_0)} \nabla \cdot [\vec{u}(\vec{x},t)\,\rho(\vec{x},t)]\, dV, \tag{1.2b}$$

where we used Gauss' law and defined $\vec{u}(\vec{x},t)$ is the velocity of the fluid at each point in the domain. Since $\mathcal{V}(t_0)$ is arbitrary, we can equate the integrands to obtain the differential form of our conservation law,

$$\frac{\partial \rho(\vec{x},t)}{\partial t} + \nabla \cdot [\vec{u}(\vec{x},t)\,\rho(\vec{x},t)] = 0. \tag{1.3}$$

This formula describes the kinematics of the density field imposed by a physical conservation of mass.

It should be pointed out that since eq. 1.2 came from setting the time derivative of eq. 1.1 to zero, we have the geometric identity

$$\frac{dQ(t)}{dt} = \frac{d}{dt}\int_{\mathcal{V}(t)} q(\vec{x},t)\,dV = \int_{\mathcal{V}(t)} \left\{ \frac{\partial q(\vec{x},t)}{\partial t} + \nabla\cdot\left[\vec{u}(\vec{x},t)\,q(\vec{x},t)\right] \right\} dV, \tag{1.4}$$

which applies to any scalar, $Q(t)$, and its associated density, $q(\vec{x},t)$, over any volume $\mathcal{V}(t)$ at any time t. Although this identity is simply a multi-dimensional version of the Leibniz integal rule [33], it has been given several names throughout the literature and has several distinct derivations. In many derivations, including one put forth by Osbourne Reynolds [34], the boundary, $\partial\mathcal{V}(t)$, deforms in such a way as to keep Q constant. In that case, the velocity field, $\vec{u}(\vec{x},t)$, is not only interpreted as the fluid velocity, but also as a vector that describes this deformation of the boundary. The so-called Reynold's Transport Theorem, eq. 1.4, is in fact a very general geometric identity that is applicable to any scalar and its associated density and is independent of the choice and evolution of $\mathcal{V}(t)$.

As an interesting note, we could have arrived at eq. 1.4 with a bit less effort by computing the time derivative of $Q(t)$ with a Lie derivative applied to the form $q(\vec{x},t)\,dV$ [35, Sec. 4.3]. However, this technology was not available in Reynolds' time, and currently, the machinery of exterior differential forms has not yet reached widespread use.

Conservation of Momentum

To conserve momentum, we tally all of the forces acting on a fluid parcel and apply Newton's second law. We split the forces into a vector that describes the forces acting on the entire fluid parcel, $\vec{F}_{\text{body}}(\vec{x},t)$, and a tensor that describes the forces acting on the boundary of a fluid parcel, $\overleftrightarrow{F}_{\text{surf}}(\vec{x},t)$. Newton's second law then takes the form

$$\frac{d}{dt}\int_{\mathcal{V}(t)} \rho\vec{u}\,dV = \int_{\mathcal{V}(t)} \vec{F}_{\text{body}}\,dV + \oint_{\partial V(t)} \overleftrightarrow{F}_{\text{surf}}\cdot d\vec{A}.$$

From now on, the function parameters, $(\vec{x},t)$, will not be explicitly identified unless needed for clarity. Also, as is necessary in continuum mechanics, the forces in the intergrands are actually forces per unit volume.

As described in section 1.2.2, we can apply the Reynolds Transport Theorem, eq. 1.4 to the time derivative,

$$\int_{\mathcal{V}(t)} \left\{ \frac{\partial(\rho\vec{u})}{\partial t} + \nabla\cdot(\rho\vec{u}\vec{u}) \right\} dV = \int_{\mathcal{V}(t)} \vec{F}_{\text{body}}\,dV + \oint_{\partial V(t)} \overleftrightarrow{F}_{\text{surf}}\cdot d\vec{A}.$$

By expanding the derivatives on the LHS and applying the conservation of mass equation 1.3, we get

$$\int_{\mathcal{V}(t)} \rho \frac{D\vec{u}}{Dt}\, dV = \int_{\mathcal{V}(t)} \vec{F}_{\text{body}}\, dV + \oint_{\partial V(t)} \overleftrightarrow{F}_{\text{surf}} \cdot d\vec{A},$$

where the *material derivative* or *Lagrangian derivative*, D/Dt, is a time derivative operator that follows the fluid parcel as it measures the rate of change of a quantity. It is defined by

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla.$$

We will identify the body force with the gravitational force, $-\rho g \hat{z}$, plus a yet unidentified external force, $\vec{F}_{\text{ext}}$. The gravitational force serves to define the negative vertical direction, $-\hat{z}$. Also, we can apply Gauss' Law to the surface term and, as usual, equate the integrands. This produces a rather general momentum conservation law in its differential form,

$$\rho \frac{D\vec{u}}{Dt} = -\rho g \hat{z} + \vec{F}_{\text{ext}} + \nabla \cdot \overleftrightarrow{F}_{\text{surf}}\,.$$

The surface forces can be split into an isotropic tensor, $-P\overleftrightarrow{1}$, and its deviation, $\overleftrightarrow{\tau}$. This casts the equations into the following familiar form,

$$\rho \frac{D\vec{u}}{Dt} = -\nabla P + \nabla \cdot \overleftrightarrow{\tau} - \rho g \hat{z} + \vec{F}_{\text{ext}}. \tag{1.5}$$

As long as the fluid velocities are much smaller than the RMS velocities of the molecular motion, which is true in the ocean to a very high degree, then the fluid remains quasi-static and P can be identified with the thermodynamic pressure [36, Sec. 4.5]. The tensor $\overleftrightarrow{\tau}$ is called the *deviatoric stress tensor*. By considering the torques on a fluid parcel, it can be shown that this tensor is symmetric. Also, by requiring Galilean invariance, this tensor cannot depend on the fluid velocity, $\vec{u}$, directly but can depend on the velocity gradients, $\nabla\vec{u}$. In general, the relation between $\overleftrightarrow{\tau}$ and $\nabla\vec{u}$ can be quite complex, however for a *Newtonian fluid*, the relationship is assumed linear. Furthermore, since pure rotation of a parcel imposes no stress, we can require $\overleftrightarrow{\tau}$ to depend only on the symmetric part of $\nabla\vec{u}$, that is, $\overleftrightarrow{S} = (\nabla\vec{u} + (\nabla\vec{u})^T)/2$. This symmetric tensor is called the *strain rate tensor* and describes the deformation of the fluid parcel [27, Ch. 17].

If the fluid reacts to stresses isotropically, then the fluid itself is said to be isotropic and the relation between the deviatoric stress tensor and the strain rate tensor becomes remarkably simple [36, Sec. 4.5],

$$\overleftrightarrow{\tau} = 2\mu\, \overleftrightarrow{S} + \lambda (\nabla \cdot \vec{u})\, \overleftrightarrow{1}\,. \tag{1.6}$$

This relation has only two parameters that depend on the thermodynamic state, μ and λ. Note that this form of $\overleftrightarrow{\tau}$ only depends on the velocity through $\overleftrightarrow{S}$ since $\nabla \cdot \vec{u} = \mathrm{Tr}[\overleftrightarrow{S}]$.

1.2.3: The Incompressibility Condition

Consider the conservation of mass equation, 1.3. An application of the product rule for derivatives and a bit of algebra casts the law into a different form,

$$\nabla \cdot \vec{u} = -\frac{1}{\rho}\frac{D\rho}{Dt}. \tag{1.7}$$

In general, eq. 1.7 will not be zero since the density of a fluid parcel is locked into an equation of state with other thermodynamic quantities. However, if the flow is not turbulent, then the irreversible conversion of kinetic energy into heat is negligible and we can assume the flow is isentropic. In that case, we can consider the density, ρ, the total pressure, P, and the entropy, s, as our thermodynamic triplet and associate changes in density (or volume) of our fluid parcel with changes in pressure alone since $ds = 0$ [37, Sec. 2].

For an isentropic flow, the speed of sound in the fluid is given by [37, Sec. 63]

$$c^2 = \left(\frac{\partial P}{\partial \rho}\right)_s.$$

This converts eq. 1.7 into

$$\nabla \cdot \vec{u} = -\frac{1}{\rho c^2}\left(\frac{DP}{Dt}\right)_s. \tag{1.8}$$

We can now non-dimensionalize this equation via the transformations [36, Sec. 4.11]

$$x^\star = \frac{x}{l}, \quad t^\star = \frac{Ut}{l}, \quad \vec{u}^\star = \frac{\vec{u}}{U}, \quad \rho^\star = \frac{\rho}{\rho_0}, \quad P^\star = \frac{P - P_{\mathrm{atm}}}{\rho_0 U^2},$$

where l and U are length and velocity scales representative of the flow, ρ_0 is a density scale representative of the fluid, and P_{atm} is the pressure at the surface of the ocean. The conservation of mass equation 1.8 now becomes

$$\nabla^\star \cdot \vec{u}^\star = -M^2 \frac{1}{\rho^\star}\left(\frac{DP^\star}{Dt^\star}\right)_s,$$

where the Mach number is defined by $M = U/c$. In sea water, c is approximately 1500m/s [38] whereas the advection speeds are typically $\mathcal{O}(10\mathrm{cm/s})$ and in certain exceptional cases, such as those found in the Gulf Stream, are as large as $\mathcal{O}(1\mathrm{m/s})$ [39]. Therefore, all of the flows considered in this manuscript are, to an exceptional extent, low-mach flows, and $\nabla \cdot \vec{u} = \mathcal{O}(M^2) << 1$.

If we explicitly write out eq. 1.7 in Cartesian coordinates, we see that

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \mathcal{O}(M^2).$$

So, to approximately one part in a million, the 3 terms of the divergence balance each other. In other words, although each term of the divergence is typically significantly larger than $\mathcal{O}(M^2)$, these potentially large terms result in a small sum. In the ocean, this means that the horizontal convergence or divergence of fluid is primarily balanced by a net upwelling or downwelling, rather than a density change. Thus, for the low-Mach flows of the ocean, we impose the *incompressibility condition*,

$$\nabla \cdot \vec{u} = 0. \tag{1.9}$$

Note that this does not mean that the density of a fluid parcel does not change. It simply expresses the fact that density changes are governed by dynamics that do not significantly effect the convergence or divergence of fluid. The dynamics of the density field are primarily derived from the fluid's equation of state. In particular, the relation among density and temperature will be discussed in section 1.2.5.

1.2.4: The Hydrostatic Pressure

If we combine eqs. 1.5, 1.6, and 1.9, we get

$$\rho \frac{D\vec{u}}{Dt} = -\nabla P - \rho g \hat{z} + \nabla \cdot (\mu \nabla \vec{u}) + \vec{F}_{\text{ext}}.$$

Note that the incompressibility condition simplifies the stress tensor to contain just a single thermodynamic parameter, the molecular viscosity, μ. If we divide both sides by a reference density, ρ_0, that is representative of the background stratification[1], we obtain

$$\frac{\rho}{\rho_0} \frac{D\vec{u}}{Dt} = -\frac{1}{\rho_0} \nabla P - \frac{\rho g}{\rho_0} \hat{z} + \nabla \cdot (\nu \nabla \vec{u}) + \frac{1}{\rho_0} \vec{F}_{\text{ext}},$$

where the new quantity, $\nu = \mu / \rho_0$, is the kinematic viscosity.

Using the background stratification, $\bar{\rho}(z)$, and the pressure at the surface of the ocean, P_{atm}, we can define the hydrostatic pressure,

$$\bar{P} = P_{\text{atm}} + g \int_z^0 \bar{\rho}(z')\, dz',$$

[1] Reasonable choices of ρ_0 include the domain average of $\bar{\rho}(z)$ or its minimum or maximum value.

and perform the following algebraic maneuver,

$$\frac{\rho}{\rho_0}\frac{D\vec{u}}{Dt} = -\frac{1}{\rho_0}\nabla\left(P-\bar{P}\right) - \frac{(\rho-\bar{\rho})g}{\rho_0}\hat{z} + \nabla\cdot(\nu\nabla\vec{u}) + \frac{1}{\rho_0}\vec{F}_{\text{ext}}.$$

We can now define the *buoyancy deviation*, b, the specific hydrostatic pressure deviation, p, and the specific external force, f_{ext},

$$b = g\frac{\rho-\bar{\rho}}{\rho_0}, \qquad p = \frac{P-\bar{P}}{\rho_0}, \qquad \vec{f}_{\text{ext}} = \frac{\vec{F}_{\text{ext}}}{\rho_0}.$$

With these definitions, the conservation of momentum equations become

$$\frac{\rho}{\rho_0}\frac{D\vec{u}}{Dt} = -\nabla p - b\hat{z} + \nabla\cdot(\nu\nabla\vec{u}) + \vec{f}_{\text{ext}}. \tag{1.10}$$

In the ocean, since the ratio of horizontal to vertical length scales are approximately $1000:1$, it is often assumed that the horizontal wavenumbers, k, are much smaller than the vertical wavenumbers, m. Through the forthcoming derivation of relation 1.23 in section 1.3.3, it can be seen that in the limit $k/m \to 0$, we can approximate Dw/Dt to be zero. This is called the *hydrostatic approximation* and arises from a geometric constraint on the flow. Dynamically, this approximation applied to the stratified ocean assumes that the flow is very nearly in the plane of the background isopycnals (surfaces of constant density). With the vertical acceleration assumed negligible in the momentum equation 1.10, the only source of diapycnal motion comes from the incompressibility equation 1.9. While many regional and global ocean models assume hydrostasy [13], the work presented in this document will take strides to avoid it. As we shall see, the dynamics of internal waves are non-hydrostatic, but highly localized. Therefore, while the large majority of the ocean is very nearly hydrostatic, there are small, localized patches that are non-hydrostatic. This fact will be of importance in what follows.

1.2.5: The Boussinesq Approximation

If we neglect salinity and assume a linear equation of state, $\rho = \rho_0[1+\alpha(T-T_0)]$, then the conservation of momentum equation 1.10 becomes

$$[1+\alpha(T-T_0)]\frac{D\vec{u}}{Dt} = -\nabla p - b\hat{z} + \nabla\cdot(\nu\nabla\vec{u}) + \vec{f}_{\text{ext}}.$$

In the ocean, the volumetric thermal expansion coefficient, α, is approximately $2.1\times10^{-4}\text{K}^{-1}$ and $\alpha(T-T_0) << 1$. We can therefore approximate $\rho/\rho_0 \approx 1$ in fluid acceleration term. In doing so, we impart the

Boussinesq approximation on the conservation of momentum equation,

$$\frac{D\vec{u}}{Dt} = -\nabla p - b\hat{z} + \nabla \cdot (\nu \nabla \vec{u}) + \vec{f}_{\text{ext}}. \tag{1.11}$$

Although it is tempting to make a similar simplification to the density field that is buried inside the buoyancy, b, we cannot do so. As stated in section 1.2.4, the majority of the ocean is nearly in hydrostatic balance, which places an exacting importance on the buoyant forcing. Regardless of whether or not the Boussinesq approximation is assumed, the the dynamical forcing terms on the RHS of the z-component of this equation are very nearly in balance throughout the ocean and none of these terms can be removed or simplified [40, Sec. 3.7].

Our linear equation of state has further consequences due to the fact that the temperature of a fluid parcel, T, obeys the heat equation,

$$\frac{DT}{Dt} = \kappa \nabla^2 T,$$

where κ is the thermal diffusivity of the fluid. Cushman-Roisin and Beckers show that this heat equation is a result of Fourier's law of heat conduction and the incompressibility condition applied to the first law of thermodynamics [40, Sec. 3.4–3.7]. By applying the material derivative to the linear equation of state, we get

$$\frac{D\rho}{Dt} = \rho_0 \alpha \frac{DT}{Dt} = \rho_0 \alpha \kappa \nabla^2 T = \kappa \nabla^2 \rho.$$

That is,

$$\frac{D\rho}{Dt} = \kappa \nabla^2 \rho. \tag{1.12}$$

Given the incompressibility condition, eq. 1.9, the Boussinesq approximation invalidates conservation of mass, eq. 1.7, in favor of conservation of volume. In other words, eq. 1.9 still holds, but eq. 1.7 does not. It has been replaced with eq. 1.12 which allows for fictitious mass fluxes to serve as a proxy for thermal diffussion [27, Sec. 2.4.3]. Since sea water is very nearly incompressible, this is a reasonable approximation for much of the dynamics of the ocean. This removes the dynamically unimportant acoustic waves from the model, allowing for a large increase of the maximum stable time step in our numerical integration of the Navier-Stokes equations [41, pp. 335–336]. On the other hand, if we wish to model the fluid's thermodynamics, we would be at a loss. Assuming the conservation of volume spoils even the most basic thermodynamic principles, such as $dW = PdV$.

Throughout the remainder of this document, we will consider an alternate form of this equation that

involves buoyancy rather than density. The total buoyancy and background buoyancy are defined via

$$b_T = g\frac{\rho - \rho_0}{\rho_0} \quad \text{and} \quad \bar{b} = g\frac{\bar{\rho} - \rho_0}{\rho_0} \tag{1.13}$$

which is made consistent with our definition of the buoyancy deviation through $b_T = \bar{b} + b$, and we must keep in mind that the background buoyancy profile is a function of z alone. In terms of the total buoyancy, the equation of state reads $b_T = \alpha(T - T_0)$ and eq. 1.12 becomes

$$\frac{Db_T}{Dt} = \kappa\nabla^2 b_T.$$

While we do wish to diffuse the dynamic buoyancy deviation, b, we do not wish to do the same to the background buoyancy, $\bar{b}$, since this profile is often maintained by unmodeled phenomena. To this end, we manually remove the term that diffuses the background buoyancy, leaving

$$\frac{Db}{Dt} + w\frac{\partial \bar{b}}{\partial z} = \kappa\nabla^2 b. \tag{1.14}$$

If the background stratification is stable, then heavy fluid lies beneath lighter fluid and $d\bar{b}/dz < 0$. To emphasize this, we define $N = (-d\bar{b}/dz)^{1/2}$ and cast the buoyancy evolution equation into its final form,

$$\frac{Db}{Dt} = N^2 w + \kappa\nabla^2 b. \tag{1.15}$$

1.2.6: The Coriolis Effect

In Geophysical Fluid Dynamics (GFD), it is customary to denote the local eastward, northward, and upward unit vectors as $\hat{x}$, $\hat{y}$, and $\hat{z}$, respectively [40, Sec. 2.5]. If we take the time derivative of any vector, $\vec{Q}$, in this rotating basis, then we must account for the explicit time dependence of $\vec{Q}$ in the rotating frame and the change in the basis vectors' orientation. Both of these terms are neatly derived and packaged in a formula from Marion and Thornton [42, p. 390]

$$\left(\frac{d\vec{Q}}{dt}\right)_{\text{fixed}} = \left(\frac{d\vec{Q}}{dt}\right)_{\text{rotating}} + \vec{\Omega} \times \vec{Q},$$

where the subscripts denote which frame of reference the derivative is measured and $\vec{\Omega}$ is the rotation vector. This formula, applied to a fluid parcel's position on the Earth, $\vec{x}$, provides a connection between the fluid

velocity as measured by an inertial observer and an observer on the surface of the rotating Earth,

$$\vec{u}_{\text{fixed}} = \vec{u}_{\text{rotating}} + \vec{\Omega} \times \vec{x}.$$

We can now differentiate again to obtain an acceleration as measured by an inertial observer,

$$\begin{aligned}
\vec{a}_{\text{fixed}} &= \left(\frac{d}{dt}\right)_{\text{fixed}} \vec{u}_{\text{fixed}} \\
&= \left(\frac{d}{dt}\right)_{\text{fixed}} \left[\vec{u}_{\text{rotating}} + \vec{\Omega} \times \vec{x}\right] \\
&= \left[\left(\frac{d}{dt}\right)_{\text{rotating}} + \vec{\Omega}\times\right] \left[\vec{u}_{\text{rotating}} + \vec{\Omega} \times \vec{x}\right] \\
&= \vec{a}_{\text{rotating}} + 2\vec{\Omega} \times \vec{u}_{\text{rotating}} + \vec{\Omega} \times \vec{\Omega} \times \vec{x}.
\end{aligned}$$

The $2\vec{\Omega} \times \vec{u}_{\text{rotating}}$ term is the Coriolis acceleration term. The $\vec{\Omega} \times \vec{\Omega} \times \vec{x}$ term is a centrifugal acceleration term and can be reduced to $-\Omega^2 R\cos(\Theta)\hat{z}$, where R is the fluid parcel's distance to the center of the Earth and Θ is the local latitude. Since the centrigugal acceleration is oriented downward, we simply redefine the local gravitational acceleration constant, $g - \Omega^2 R\cos(\Theta) \to g$, and neglect the explicit addition of this term in the dynamical equations. At 45° N latitude, this redefinition alters the value of g by approximately 0.2%. Alternatively, if we express the centrifugal acceleration as the gradient of a potential by assuming $\cos(\Theta)$ is uniform throughout the domain of interest, we can simply push the entire forcing term into the yet undetermined pressure.

Our momentum equation 1.11, can now be expressed in the Earth's rotating frame by the addition of a Coriolis term,

$$\frac{D\vec{u}}{Dt} + 2\vec{\Omega} \times \vec{u} = -\nabla p - b\hat{z} + \nabla \cdot (\nu \nabla \vec{u}) + \vec{f}_{\text{ext}}. \tag{1.16}$$

In the ocean, the vertical velocity, w, is typically much smaller than the horizontal velocities, u and v, due to geometric considerations. This simplifies the Coriolis term in the momentum equation since

$$\begin{aligned}
2\vec{\Omega} \times \vec{u} &= 2\Omega[(w\cos\Theta - v\sin\Theta)\hat{x} + u\sin\Theta\hat{y} - u\cos\Theta\hat{z}] \\
&\approx 2\Omega[-v\sin\Theta\hat{x} + u\sin\Theta\hat{y} - u\cos\Theta\hat{z}] \\
&= -fv\hat{x} + fu\hat{y} - 2\Omega\cos\Theta u\hat{z},
\end{aligned}$$

where the *Coriolis parameter* is defined as $f = 2\Omega\sin\Theta$. On Earth, f ranges from $+1.45 \times 10^{-4}\text{s}^{-1}$ at the North pole to zero at the equator to $-1.45 \times 10^{-4}\text{s}^{-1}$ at the South pole. We will often restrict ourselves

to regions of the ocean in which the latitude does not vary much. In those cases, we employ the *f-plane approximation* by assuming that the Coriolis parameter is uniform throughout the domain of interest. As a final simplification, the z-component of the Coriolis acceleration, $-2\Omega\cos\Theta u$, is often assumed negligible compared to the pressure gradient term, leaving

$$2\vec{\Omega}\times\vec{u}\approx -fv\hat{x}+fu\hat{y}.$$

For a complete description of the scales of oceanic motion and the simplification of the Coriolis terms, see Cushman-Roisin and Beckers [40, Sec. 4.3].

1.2.7: Non-Orthogonal Coordinate Systems

Equations 1.16, 1.15, and 1.9 constitute the Incompressible Navier-Stokes equations under the Boussinesq approximation,

$$\frac{D\vec{u}}{Dt}+2\vec{\Omega}\times\vec{u}=-\nabla p-b\hat{z}+\nabla\cdot(\nu\nabla\vec{u})+\vec{f}_{\text{ext}} \tag{1.17a}$$

$$\frac{Db}{Dt}=N^2w+\kappa\nabla^2 b \tag{1.17b}$$

$$\nabla\cdot\vec{u}=0 \tag{1.17c}$$

We wish to formulate eqs. 1.17 in general curvilinear coordinates without the use of Christoffel symbols since in 3D, they are 18 independent functions that are expensive to compute and store.

To proceed, we will use index notation and sum over repeated indices. Greek indices will refer to Cartesian components of vectors and tensors, while latin indices will refer to curvilinear components. $\{x^\alpha\}$ are the Cartesian coordinates and $\{\xi^i\}$ are the curvilinear coordinates. The metric tensor is defined via the Jacobian elements,

$$g_{ij}=\sum_\alpha \frac{\partial x^\alpha}{\partial\xi^i}\frac{\partial x^\alpha}{\partial\xi^j},$$

the inverse metric tensor is denoted g^{ij}, and the Jacobian determinant is $J=\sqrt{\det(g_{ij})}$. The generalized gradient and divergence operators are

$$\nabla^i f=g^{ij}\partial_j f \quad\text{and}\quad \nabla_i u^i=\frac{1}{J}\partial_i J u^j, \tag{1.18}$$

where ∂_i is shorthand for partial derivatives with respect to curvilinear coordinates [35, Ch. 2]. These derivatives are imposed on everything to their right, $\partial_i fg=g\partial_i f+f\partial_i g$. The Laplacian operator can be

generalized by contracting the covariant divergence and covariant gradient operators. When operating on a scalar, the Laplacian is expressed as

$$\nabla^2 = \frac{1}{J}\partial_i J g^{ij} \partial_j.$$

When applied to the curvilinear components of a vector or tensor, this operator becomes more complicated and involves Christoffel symbols. For this reason, we will always apply the Laplacian to scalars and Cartesian components of vectors and tensors. For a complete description of the notation used here and a thorough explanation of Riemannian geometry, see Lovelock and Rund [43].

The incompressibility condition is generalized by applying the covariant divergence of eqs. 1.18 to the curvilinear velocity components,

$$\partial_i J u^i = 0,$$

where we have multiplied both sides by J to simplify the equation. The buoyancy equation can be converted similarly,

$$\frac{\partial b}{\partial t} + u^i \partial_i b = N^2 w + \kappa \nabla^2 b,$$

where the advection term utilizes the convenient transformation

$$u^\beta \frac{\partial}{\partial x^\beta} = u^i \partial_i.$$

The same simple transformation could have been applied to the advective term involving the background buoyancy, but since the $\bar{b}$ profile is not dynamic, and it is a function of z alone, it is convenient to compute this term in Cartesian coordinates. To finalize the covariant buoyancy equation, we cast the advective term into a conservative form by use of the incompressibility condition. The resulting form,

$$\frac{\partial b}{\partial t} + \frac{1}{J}\partial_i J u^i b = N^2 w + \kappa \nabla^2 b,$$

is better suited to the numerical schemes that will be presented in chapter 4.

To generalize the momentum equation, we will again convert the derivative operators using eqs. 1.18, but we will apply them to the Cartesian components of the velocity vector to avoid Christoffel symbols. These updated components can then be used to generate the curvilinear velocity vector via $u^i = u^\alpha(\partial \xi^i / \partial x^\alpha)$ if needed. The α-component of the momentum equation becomes

$$\frac{\partial u^\alpha}{\partial t} + u^i \partial_i u^\alpha + 2\varepsilon^\alpha_{ij}\Omega^i u^j = -g^{\alpha i}\partial_i p - b\hat{z}^\alpha + \nu \nabla^2 u^\alpha + f^\alpha_{\text{ext}}. \tag{1.19}$$

Note that with ν removed from the Laplacian, we are assuming a uniform viscosity. Also note that, by definition, $g^{\alpha i} = (\partial \xi^\alpha / \partial x^j) g^{ji}$. As in the buoyancy equation, we wish to cast the advective term into a conservative form. The resulting set of equations is

$$\frac{\partial u^\alpha}{\partial t} + \frac{1}{J}\partial_i J u^i u^\alpha + 2\varepsilon^\alpha_{ij}\Omega^i u^j = -g^{\alpha i}\partial_i p - b\hat{z}^\alpha + \nu\nabla^2 u^\alpha + f^\alpha_{\text{ext}} \tag{1.20a}$$

$$\frac{\partial b}{\partial t} + \frac{1}{J}\partial_i J u^i b = N^2 w + \kappa\nabla^2 b \tag{1.20b}$$

$$\partial_i J u^i = 0. \tag{1.20c}$$

The f-plane approximation is recovered by replacing $2\varepsilon^\alpha_{ij}\Omega^i u^j$ with $f\varepsilon^\alpha_{ij}\frac{\partial \xi^i}{\partial z}u^j$.

Section 1.3: Physics Of A Stratified Fluid

1.3.1: Buoyancy Oscillations

Consider a fluid parcel removed vertically a small distance δz from its hydrostatic position, z_0. The parcel then has a buoyancy of $\bar{b}(z_0)$ and is surrounded by fluid whose buoyancy is $\bar{b}(z_0 + \delta z)$. If we choose $\rho_0 = \bar{\rho}(z_0)$, then, by definition,

$$\bar{b}(z_0) = g\frac{\bar{\rho}(z_0) - \rho_0}{\rho_0} = 0.$$

We can use this in a Taylor expansion of $\bar{b}(z_0 + \delta z)$,

$$\bar{b}(z_0 + \delta z) = \delta z \left.\frac{d\bar{b}}{dz}\right|_{z_0} + \mathcal{O}\left(\delta z^2\right).$$

If the fluid is stably stratified, then $d\bar{b}/dz < 0$. To emphasize this, we will define

$$N^2 = -\left.\frac{d\bar{b}}{dz}\right|_{z_0}.$$

Now, by construction, our fluid parcel of zero buoyancy has navigated into a region whose ambient buoyancy is approximately $-N^2\delta z$. Newton's second law provides an equation of motion for the displaced parcel,

$$\frac{d^2\delta z}{dt^2} = -N^2\delta z.$$

If the background density decreases with depth, then $d\bar{b}/dz > 0$, N is imaginary, and the solutions become real-valued exponentials. This exhibits the *Rayleigh-Taylor instability.* On the other hand, if the background density increases with depth, then the stratification is stable and we recover oscillatory solutions of the form

$\delta z \sim e^{\pm iNt}$ with N a real-valued angular frequency of the motion called the *Brunt-Väisälä frequency*. These *buoyancy oscillations* are at the heart of internal wave modeling. As we will see in section 1.3.3, this simple harmonic motion provides the smallest timescale of dynamical importance in laminar regimes of a stratified ocean.

1.3.2: The Internal Wave Equation

Buoyancy oscillations occur when a stably stratified fluid is perturbed from its rest hydrostatic state. Consider a continuously stratified fluid whose rest state is characterized by $\vec{u} = 0$, $\rho = \bar{\rho}$, and $P = \bar{P}$. As usual, $\bar{\rho}$ is the background density profile and $\bar{P}$ is its associated hydrostatic pressure defined by $\partial \bar{P}/\partial z = -\bar{\rho} g$. Deviations from this rest state are described in Cartesian coordinates under the Boussinesq approximation by eqs. 1.17. If we ignore viscous, diffusive, and nonlinear effects, these equations become

$$\frac{\partial u}{\partial t} - fv = -\frac{\partial p}{\partial x} \tag{1.21a}$$

$$\frac{\partial v}{\partial t} + fu = -\frac{\partial p}{\partial y} \tag{1.21b}$$

$$\chi \frac{\partial w}{\partial t} = -\frac{\partial p}{\partial z} - b \tag{1.21c}$$

$$\frac{\partial b}{\partial t} = N^2 w \tag{1.21d}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \tag{1.21e}$$

where χ is set to 1 for non-hydrostatic flows and 0 for hydrostatic flows.

Buoyancy oscillations can be recovered by considering the particular non-hydrostatic solutions with $u = v = \partial p/\partial z = 0$. These solutions must satisfy

$$\frac{\partial w}{\partial z} = 0$$

$$\frac{\partial^2 w}{\partial t^2} + N^2 w = 0$$

$$\frac{\partial^2 b}{\partial t^2} + N^2 b = 0.$$

Of significant importance is the fact that both p and w are constant throughout vertical water columns. This fact will crop up again when considering how our adaptive grids will be structured in the vicinity of an interfacial wave (eg. a depression in the pycnocline).

Returning to the linearized equations 1.21, we can apply $\partial/\partial x$ and $\partial/\partial y$ to the u and v evolution

equations, then add to obtain

$$\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)=f\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)-\nabla_h^2 p.$$

Application of the incompressibility condition produces

$$-\frac{\partial}{\partial t}\frac{\partial w}{\partial z}=f\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)-\nabla_h^2 p.$$

We must now return to equations 1.21 to produce a vorticity equation. Applying derivatives and subtracting yields

$$\frac{\partial}{\partial t}\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)=f\frac{\partial w}{\partial z}.$$

These equations can be combined to eliminate the vorticity

$$\left(\frac{\partial^2}{\partial t^2}+f^2\right)\frac{\partial w}{\partial z}=\frac{\partial}{\partial t}\nabla_h^2 p.$$

Finally, we can apply $\partial/\partial z$ and substitute the z-component of the linearized momentum equation and linearized buoyancy equation to arrive at a Sobolev equation [44], called the *internal wave equation*,

$$\left(\chi\nabla_h^2+\frac{\partial^2}{\partial z^2}\right)\frac{\partial^2 w}{\partial t^2}+\left(N^2\nabla_h^2+f^2\frac{\partial^2}{\partial z^2}\right)w=0. \tag{1.22}$$

1.3.3: Dispersion

Equation 1.22 is a hyperbolic linear PDE, which means we can expect traveling solutions of the form $e^{i(\vec{K}\cdot\vec{x}-\omega t)}$. In this section, we will restrict ourselves to two spatial dimensions without loss of generality. Upon substitution into eq. 1.22, we find the *dispersion relation* among the temporal frequencies, ω, and the spatial frequencies, $\vec{K}=(k,m)$,

$$\omega^2=\frac{N^2k^2+f^2m^2}{\chi k^2+m^2}. \tag{1.23}$$

This relation provides important information about the direction and speeds of phase lines and energy transport, how harmonic forcing will influence the fluid, and how the hydrostatic approximation impacts the dynamics. In what follows, we will restrict ourselves to the non-rotating, $f=0$ limit.

Non-hydrostatic case ($\chi = 1$)

If we define α to be the angle between the vertical and the direction of phase propagation, $\hat{K}$, then this relation is simply expressed as

$$\omega = \pm N \sin\alpha. \tag{1.24}$$

Therefore, given a particular stratification, N, and a particular forcing frequency, ω, all internal waves will have lines of constant phase that propagate at angles of $\pm\alpha$ with the vertical. Furthermore, since $\sin\alpha \leq 1$, the forcing frequency must be less than or equal to the frequency of buoyancy oscillations, N, if internal waves are to form. If $\omega > N$, the energy given to the system will disturb the fluid locally and produce turbulence, but the energy will not be carried away by waves [40, Sec. 13.2]. Thus, the Brunt-Väisälä frequency represents the highest frequency of wave-like motion in the ocean.

Given a dispersion relation, we can gather useful information about the speeds of the Fourier modes that construct the internal waves. The phase speed in the x-direction is given by following a portion of the wave that is at constant phase, ϕ_0, along the x-axis. That is, given the phase function, $\phi(\vec{x},t) = \vec{K}\cdot\vec{x} - \omega t$, we can choose a particular value, $\phi(\vec{x},t) = \phi_0$, and measure the position of this phase value as it moves along the x-axis. This produces $\phi_0 = kx - \omega t$, or $x(t) = (\omega/k)t - \phi_0/k$. The phase speed along the x-axis is simply the coefficient of t. When we perform the same analysis along the z-axis and along the direction of phase propagation, $\hat{K}$, we recover the following formulas

$$c_p^{(x)} = \frac{\omega}{k}, \qquad c_p^{(z)} = \frac{\omega}{m}, \qquad c_p = \frac{\omega}{K}.$$

It is extremely important to notice that $c_p^{(x)}$ and $c_p^{(z)}$ are not the x and z components of the phase velocity, $c_p\hat{K}$. This fact is emphasized in the limit of purely horizontal wave motion, $\alpha \to \pm 90°$. If we let $k \to K$ and $m \to 0$, then $c_p^{(x)} \to c_p$ while $c_p^{(z)} \to \pm\infty$.

Using our dispersion relation, eq. 1.24, we find the phase speeds for internal waves,

$$c_p^{(x)} = \pm\frac{N}{K}, \qquad c_p^{(z)} = \pm\frac{N}{K}\tan\alpha, \qquad c_p = \pm\frac{N}{K}\sin\alpha.$$

In the limit $\alpha \to 0$, only $c_p^{(x)}$ has a non-zero value, but since the x-direction is tangent to the surfaces of constant phase, $c_p^{(x)}$ is irrelevant. In this degenerate limit, there is no wave motion at all, in accordance with the fact that the forcing frequency, $\omega = \pm N\sin\alpha$, must also approach zero.

Next, we can compute the group velocity of our internal waves via $\vec{c_g} = \nabla_{\vec{K}}\omega$. The group velocity, which

dictates the direction of energy transport, has components

$$c_g^{(x)} = \pm\frac{N}{K}\cos^2\alpha, \qquad c_g^{(z)} = \mp\frac{N}{K}\sin\alpha\cos\alpha.$$

By computing the scalar product of $\vec{c}_p$ and $\vec{c}_g$, we find

$$\begin{aligned}\vec{c}_p\cdot\vec{c}_g &= c_p\hat{K}\cdot\vec{c}_g \\ &= \left[\pm\frac{N}{K}(\sin\alpha,\cos\alpha)\right]\cdot\left[\pm\frac{N}{K}\cos\alpha(\cos\alpha,-\sin\alpha)\right] \\ &= 0.\end{aligned}$$

Thus, the phase and group velocities of internal waves are orthogonal to each other.

This phenomenon is illustrated by vertically oscillating a bobber in a vertically, linearly stratified fluid. We choose the ratio of the forcing frequency to the Brunt-Väisäilä frequency to be $\omega/N = \sqrt{2}$ so that the phase and group velocities travel at $\pm 45°$ angles to the vertical. Figures 1.1 and 1.2 show a simulated shadowgraph[2] of a 2D region of fluid with periodic BCs. These figures were generated by numerically integrating the linearized equations 1.21 with $f = 0$ and $\chi = 1$. The fluid is initially at rest and the disturbance can be seen to travel outward from the bobber, illustrating the direction of each wave's group velocity. This is shown in figure 1.1, where the four slides depict the wave structure at the start of four different forcing periods. The direction of each wave's phase velocity can be seen in figure 1.2, where each slide represents a different portion of a single forcing period. By comparing these two figures, we see that the phase and group velocities of the internal waves are, in fact, orthogonal. This internal wave pattern is known as St. Andrew's Cross. It was first demonstrated in the lab and visualized through a schlieren system by Mowbray and Rarity [47].

Hydrostatic case ($\chi = 0$)

With $\chi = 1$ and $f \to 0$, our dispersion relation is $\omega = \pm N\sin\alpha$. However, if we invoke the hydrostatic approximation by fixing $\chi = 0$, the dispersion relation becomes $\omega = \pm N\tan\alpha$. Therefore, we drastically alter the dispersive properies of the internal waves whose phase propagation is not nearly vertical. To quantify how the waves are affected, we can compute the hydrostatic phase and group velocities in terms of their non-hydrostatic counterparts.

$$c_{p,\mathrm{hs}}^{(\bullet)} = c_p^{(\bullet)}\sec\alpha, \qquad c_{g,\mathrm{hs}}^{(\bullet)} = c_g^{(\bullet)}\sec^3\alpha.$$

[2]The shadowgraph technique was first used by Robert Hooke and documented in his historic book, *Micrographia* [45]. The method, and it's more sophisticated relative, synthetic schlieren, is still in wide use today and has evolved considerably since its first appearance in 1665 [46].

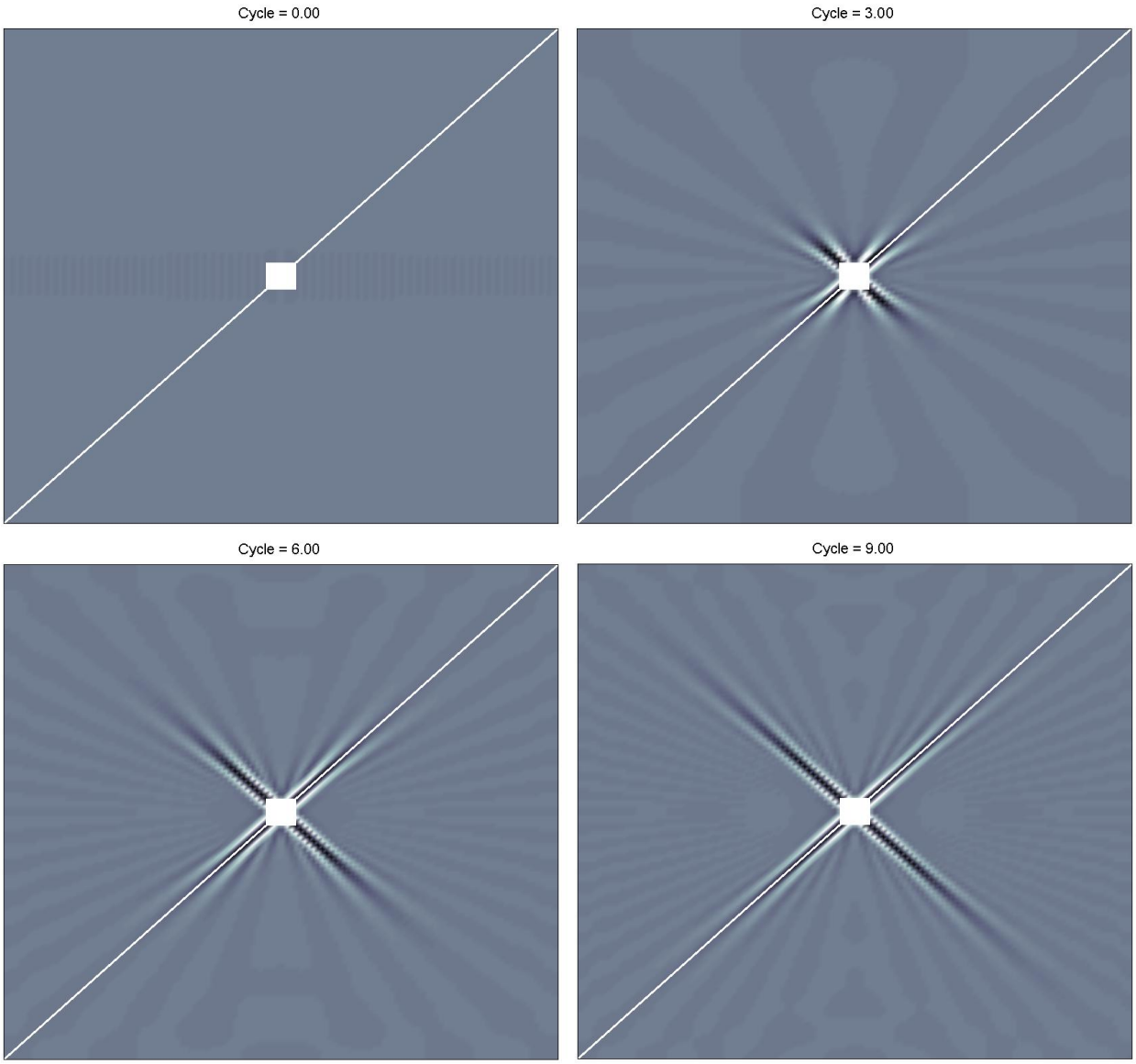


Figure 1.1: A simulated shadowgraph of St. Andrew's Cross, illustrating the direction of each wave's group velocity. Each slide represents the start of a forcing period (cycle). The solid white square indicates the location of the bobber and the diagonal white line is provided for reference.

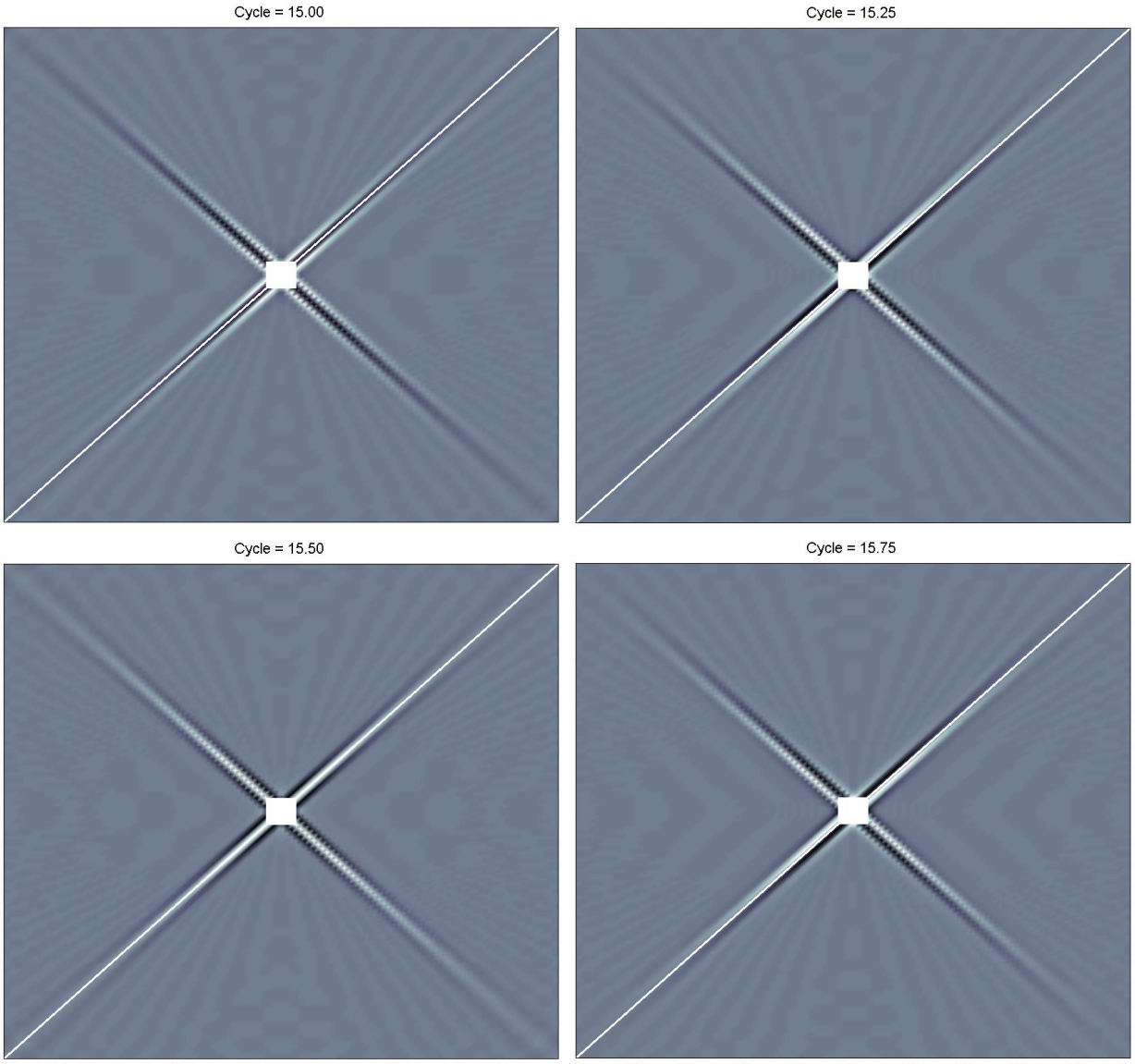


Figure 1.2: A simulated shadowgraph of St. Andrew's Cross, illustrating the direction of each wave's phase velocity. Each slide represents a different portion of a single forcing period (cycle). The solid white square indicates the location of the bobber and the diagonal white line is provided for reference.

The ($\bullet$) superscripts indicate that the formula holds for speeds in any direction. The secant factors again indicate that the only internal waves that are faithfully represented by the hydrostatic approximation are those waves with $\alpha \approx 0$. As mentioned in section 1.2.4, the hydrostatic approximation assumes that the flow is very nearly in the plane of the isopycnals. Since phase and group velocities are orthogonal to each other, the internal waves at $\alpha = 0$ have group velocities directed horizontally. That is, the hydrostatic approximation is only accurate if the energy transport is very nearly directed along horizontal isopycnal surfaces. This is an extrememly important fact about internal waves – oceanic internal waves are not hydrostatic.

CHAPTER 2: Finite Volume Discretization in Curvilinear Coordinates

Our curvilinear formulations of various differential operators follow those derived in most books on Riemannian geometry. But in order to describe our discretization of these operators, we first need to get some notational issues out of the way.

1. We choose to index the cell-centers with integers and the face-centers with half-integers.

2. To adhere to convention, this section will label cell indices with i, j, k. All other latin characters are reserved for curvilinear component indices.

3. We will frequently encounter the product Jg^{mn}. For brevity, we will provide just one set of indices for both quantities so that $(Jg^{mn})_{ijk}$ represents $J_{ijk}\, g^{mn}_{ijk}$.

4. In 3D, we will denote the set of grid nodes by $\mathcal{G}$, the grid's edges in the l-direction by $\mathcal{G}^l$, the grid's faces parallel to the lm-plane by $\mathcal{G}^{lm}$, and the grids cell-centers by $\mathcal{G}^{lmn}$. Often, we will also use a dual notation. The grid's cell-centers can be denoted $\mathcal{G}'$, the grid's faces normal to the l-direction by $\mathcal{G}'_l$, the grid's edges normal to the lm-plane by $\mathcal{G}'_{lm}$, and the grid's nodes by $\mathcal{G}'_{lmn}$. For an illustration of these centerings, see table 2.1. In 2D, this set of symbols reduces to $\mathcal{G}, \mathcal{G}^l, \mathcal{G}^{lm}$ and $\mathcal{G}', \mathcal{G}'_l, \mathcal{G}'_{lm}$.

In 3D, the basic operators of vector calculus, written in a covariant (coordinate-independent) way, is summarized by [27, Ch. 21]

Operator	domain $\to$ range	formula
Gradient	scalars $\to$ vectors	$(\nabla\phi)^m = g^{mn}\partial_n\phi$
Curl	vectors $\to$ vectors	$(\nabla \times \vec{u})^m = \varepsilon^{mpq}\partial_p u_q$
Divergence	vectors $\to$ scalars	$\nabla \cdot \vec{u} = \frac{1}{J}\partial_m J u^m$
Laplacian	scalars $\to$ scalars	$\nabla^2\phi = \frac{1}{J}\partial_m J g^{mn}\partial_n\phi.$

Note that the curl uses the covariant components of $\vec{v}$, which are generated by applying the metric tensor, $u_m = g_{mn}u^n$. In the SOMAR code, a design choice was made to redefine the gradient to produce fluxes rather than vectors. This means that once the above formula for the gradient is used, the result is multiplied by J. The divergence is also redefined to operate on fluxes, which assumes that we do not need to multiply

by J before computing partial derivatives. The Laplacian is simply the contraction of the divergence and gradient and is not effected by these alterations.

Our discretization of geometric quantities and differential operators must be *freestream preserving*, meaning they cannot impose a bias on the direction of simple flows based on the direction of the grid lines. Also, when enforcing the incompressibility condition, we will require our velocity vector to be expressed as the sum of an irrotational component and a solenoidal component, both of which are defined using our discrete operators. These issues will be the focus of this chapter.

Section 2.1: Freestream preserving metric discretization

The choice of discretization of the Jacobian elements and metric tensor significantly impacts the long-term accuracy and stability of our numerical solutions. To illustrate this, consider the velocity advection equation,

$$\frac{\partial u^\alpha}{\partial t} + \frac{1}{J}\partial_m J u^m u^\alpha = 0 \tag{2.1}$$

in a 2D domain with periodic BCs. We will set the Cartesian components of the initial velocity to $(U, 0)$, where U is uniform and constant. The single non-trivial component of the evolution equation becomes

$$0 = \partial_m J u^m U = U^2 \partial_m J \frac{\partial \xi^m}{\partial x},$$

or simply

$$0 = \partial_m J \frac{\partial \xi^m}{\partial x}. \tag{2.2}$$

This equation is no longer a dynamical evolution equation, but a requirement that must be satisfied by the Jacobian elements. If this equation does not hold, then a uniform velocity field will not remain so because the construction of the curvilinear grid will impose a bias on the dynamics, resulting in inaccuracies and potential instabilities that can grow in time. Note that if we were to choose an initial velocity whose Cartesian components read $(0, V)$, then we would have arrived at a similar restriction,

$$0 = \partial_m J \frac{\partial \xi^m}{\partial y}. \tag{2.3}$$

We can combine eqs. 2.2 and 2.3 into the more consise requirement,

$$0 = \partial_m J \frac{\partial \xi^m}{\partial x^\alpha} \quad \forall \alpha. \tag{2.4}$$

A discretization of the Jacobian elements and operators that respect eqs. 2.4 exactly is called a *freestream-preserving discretization.*

Equations 2.4 will be satisfied if

$$\begin{aligned} J\frac{\partial \xi}{\partial x^\alpha} &= \quad \partial_\eta F_\alpha \\ J\frac{\partial \eta}{\partial x^\alpha} &= -\partial_\xi F_\alpha, \end{aligned}$$

for some set of functions, F_α. To arrive at these functions, let us explicitly write out the Jacobian and inverse Jacobian elements,

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}, \qquad [J]^{-1} = \frac{1}{J}\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{bmatrix}, \qquad J = \det([J]).$$

Now, we requre the inverse Jacobian elements of this coordinate mapping to be identically equal to the Jacobian elements of the inverse coordinate mapping. That is,

$$[J]^{-1} = \frac{1}{J}\begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial x}{\partial \eta} \\ -\frac{\partial y}{\partial \xi} & \frac{\partial x}{\partial \xi} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix}.$$

This produces useful relations among the various Jacobian elements. In particular, we have the following four relations,

$$\begin{aligned} J\frac{\partial \xi}{\partial x} &= \quad \frac{\partial y}{\partial \eta} & J\frac{\partial \xi}{\partial y} &= -\frac{\partial x}{\partial \eta} \\ J\frac{\partial \eta}{\partial x} &= -\frac{\partial y}{\partial \xi} & J\frac{\partial \eta}{\partial y} &= \quad \frac{\partial x}{\partial \xi} \end{aligned} \tag{2.5}$$

which provides the two functions, $F_x = y$ and $F_y = -x$, that were needed to satisfy eqs. 2.4. We must now investigate how to compute $\partial x^\alpha / \partial \xi^m$.

Suppose we evaluate the Cartesian coordinate functions, $x^\alpha = x^\alpha(\vec{\xi})$, at each node in $\mathcal{G}$. We can then compute the exact average of the partial derivatives $\partial_m x^\alpha$ over each edge in $\mathcal{G}^m$ via a standard second-order, centered, finite difference stencil [48, Ch. 4]. Using the identities 2.5, it is now just a matter of flipping signs to generate the exact average of $J\partial \xi^m / \partial x^\alpha$ over each edge in $\mathcal{G}'_m$. With this construction, eq. 2.4 is satisfied

as follows

$$\begin{aligned}
\left[\partial_m J\frac{\partial \xi^m}{\partial x}\right]_{i,j} &= \left[\partial_\xi J\frac{\partial \xi}{\partial x}\right]_{i,j} + \left[\partial_\eta J\frac{\partial \eta}{\partial x}\right]_{i,j} \\
&= \frac{1}{\Delta\xi}\left(\left[J\frac{\partial \xi}{\partial x}\right]_{i+1/2,j} - \left[J\frac{\partial \xi}{\partial x}\right]_{i-1/2,j}\right) + \frac{1}{\Delta\eta}\left(\left[J\frac{\partial \eta}{\partial x}\right]_{i,j+1/2} - \left[J\frac{\partial \eta}{\partial x}\right]_{i,j-1/2}\right) \\
&= \frac{1}{\Delta\xi}\left(\left[\frac{\partial y}{\partial \eta}\right]_{i+1/2,j} - \left[\frac{\partial y}{\partial \eta}\right]_{i-1/2,j}\right) - \frac{1}{\Delta\eta}\left(\left[\frac{\partial y}{\partial \xi}\right]_{i,j+1/2} - \left[\frac{\partial y}{\partial \xi}\right]_{i,j-1/2}\right) \\
&= \frac{1}{\Delta\xi}\left(\left[\frac{y_{i+1/2,j+1/2} - y_{i+1/2,j-1/2}}{\Delta\eta}\right] - \left[\frac{y_{i-1/2,j+1/2} - y_{i-1/2,j-1/2}}{\Delta\eta}\right]\right) \\
&\quad - \frac{1}{\Delta\eta}\left(\left[\frac{y_{i+1/2,j+1/2} - y_{i-1/2,j+1/2}}{\Delta\xi}\right] - \left[\frac{y_{i+1/2,j-1/2} - y_{i-1/2,j-1/2}}{\Delta\xi}\right]\right) \\
&= 0.
\end{aligned}$$

Interpereted as a finite volume discretization, we have constructed the cell averages of $\partial_m J\partial\xi^m/\partial x$ to be identically zero. A similar result holds for the cell averages of $\partial_m J\partial\xi^m/\partial y$. This necessitates a specific discretization of the divergence, which will be discussed in section 2.2.

In 3D, the construction is similar, but a bit more care needs to be taken. Eqs. 2.5 must be expressed as a curl,

$$J\frac{\partial \xi}{\partial x} = \frac{1}{2}\left\{\partial_\eta\left(y\partial_\zeta z - z\partial_\zeta y\right) - \partial_\zeta\left(y\partial_\eta z - z\partial_\eta y\right)\right\} \tag{2.6}$$

$$J\frac{\partial \eta}{\partial x} = \frac{1}{2}\left\{\partial_\zeta\left(y\partial_\xi z - z\partial_\xi y\right) - \partial_\xi\left(y\partial_\zeta z - z\partial_\zeta y\right)\right\} \tag{2.7}$$

$$J\frac{\partial \zeta}{\partial x} = \frac{1}{2}\left\{\partial_\xi\left(y\partial_\eta z - z\partial_\eta y\right) - \partial_\eta\left(y\partial_\xi z - z\partial_\xi y\right)\right\}, \tag{2.8}$$

so that divergences are identically zero. The remaining six formulas result from cyclic permutation of x, y, and z. Notice that we also chose a gauge that is symmetric in all of the coordinates. Next, we must construct a discretization of these formulas.

We again evaluate x^α at all nodes in $\mathcal{G}$, followed by the computation of averages of partial derivatives, $\partial_m x^\alpha$ over all edges in $\mathcal{G}^m$ in a manner analogous to the 2D setup. Now, using the 3D identities 2.6, we generate the averages of $J\partial\xi^m/\partial x^\alpha$ over each face in $\mathcal{G}'_m$. With this construction, the 3D discrete divergence will be identically zero. Table 2.1 illustrates the centerings of the various quantities.

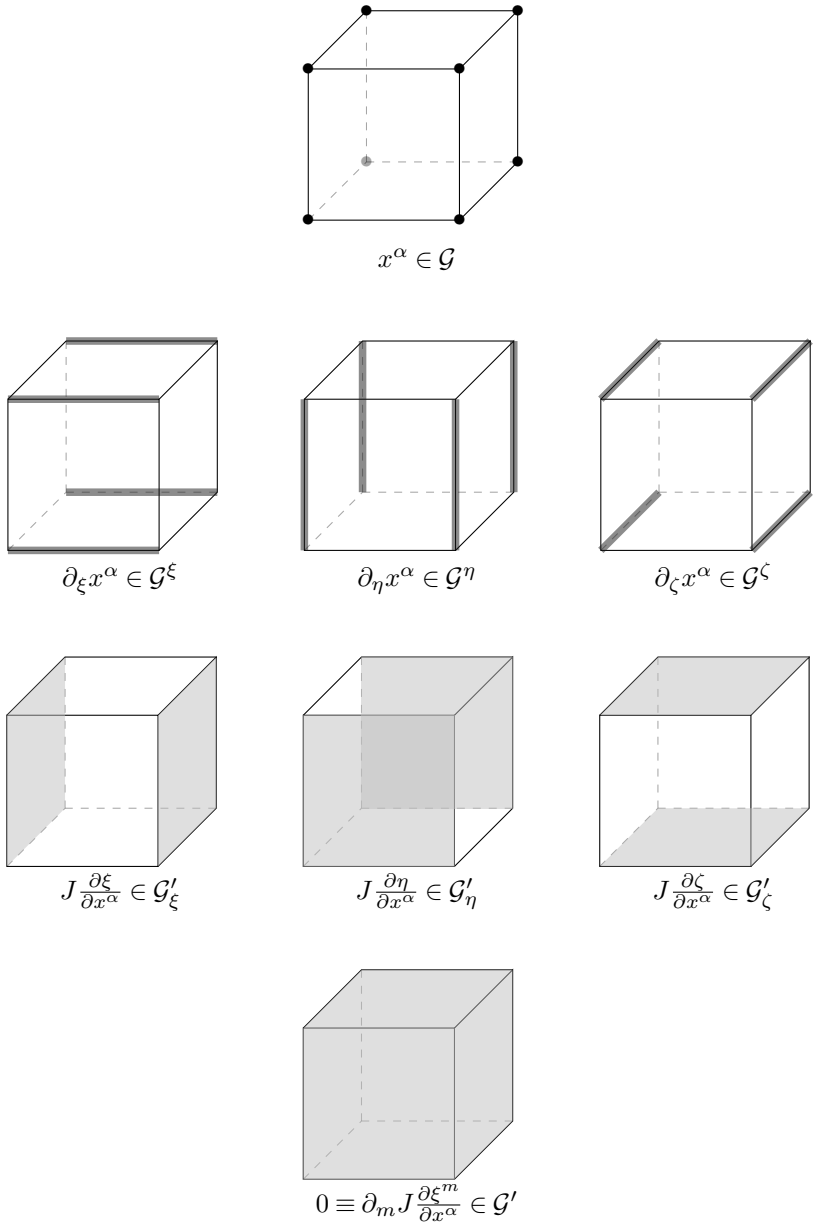


Table 2.1: The locations of various geometric quantities in a grid cell.

With this construction, eq. 2.4 is satisfied as follows

$$\begin{aligned}
\left[\partial_m J\frac{\partial \xi^m}{\partial x}\right]_{i,j,k} &= \left[\partial_\xi J\frac{\partial \xi}{\partial x}\right]_{i,j,k} + \left[\partial_\eta J\frac{\partial \eta}{\partial x}\right]_{i,j,k} + \left[\partial_\zeta J\frac{\partial \zeta}{\partial x}\right]_{i,j,k} \\
&= \frac{1}{\Delta\xi}\left(\left[J\frac{\partial \xi}{\partial x}\right]_{i+1/2,j,k} - \left[J\frac{\partial \xi}{\partial x}\right]_{i-1/2,j,k}\right) \\
&+ \frac{1}{\Delta\eta}\left(\left[J\frac{\partial \eta}{\partial x}\right]_{i,j+1/2,k} - \left[J\frac{\partial \eta}{\partial x}\right]_{i,j-1/2,k}\right) \\
&+ \frac{1}{\Delta\zeta}\left(\left[J\frac{\partial \zeta}{\partial x}\right]_{i,j,k+1/2} - \left[J\frac{\partial \zeta}{\partial x}\right]_{i,j,k-1/2}\right).
\end{aligned}$$

We define

$$A = y\partial_\xi z - z\partial_\xi y$$

$$B = y\partial_\eta z - z\partial_\eta y$$

$$C = y\partial_\zeta z - z\partial_\zeta y$$

so that

$$\begin{aligned}
\left[\partial_m J\frac{\partial \xi^m}{\partial x}\right]_{i,j,k} = &\quad \frac{1}{2\Delta\xi}\left(\left[\partial_\eta C - \partial_\zeta B\right]_{i+1/2,j,k} - \left[\partial_\eta C - \partial_\zeta B\right]_{i-1/2,j,k}\right) \\
&+ \frac{1}{2\Delta\eta}\left(\left[\partial_\zeta A - \partial_\xi C\right]_{i,j+1/2,k} - \left[\partial_\zeta A - \partial_\xi C\right]_{i,j-1/2,k}\right) \\
&+ \frac{1}{2\Delta\zeta}\left(\left[\partial_\xi B - \partial_\eta A\right]_{i,j,k+1/2} - \left[\partial_\xi B - \partial_\eta A\right]_{i,j,k-1/2}\right) \\
= &\quad \frac{1}{2\Delta\xi}\left(\left[\frac{C_{i+1/2,j+1/2,k} - C_{i+1/2,j-1/2,k}}{\Delta\eta} - \frac{B_{i+1/2,j,k+1/2} - B_{i+1/2,j,k-1/2}}{\Delta\zeta}\right]\right. \\
&\qquad \left. - \left[\frac{C_{i-1/2,j+1/2,k} - C_{i-1/2,j-1/2,k}}{\Delta\eta} - \frac{B_{i-1/2,j,k+1/2} - B_{i-1/2,j,k-1/2}}{\Delta\zeta}\right]\right) \\
&+ \frac{1}{2\Delta\eta}\left(\left[\frac{A_{i,j+1/2,k+1/2} - A_{i,j+1/2,k-1/2}}{\Delta\zeta} - \frac{C_{i+1/2,j+1/2,k} - C_{i-1/2,j+1/2,k}}{\Delta\xi}\right]\right. \\
&\qquad \left. - \left[\frac{A_{i,j-1/2,k+1/2} - A_{i,j-1/2,k-1/2}}{\Delta\zeta} - \frac{C_{i+1/2,j-1/2,k} - C_{i-1/2,j-1/2,k}}{\Delta\xi}\right]\right) \\
&+ \frac{1}{2\Delta\zeta}\left(\left[\frac{B_{i+1/2,j,k+1/2} - B_{i-1/2,j,k+1/2}}{\Delta\xi} - \frac{A_{i,j+1/2,k+1/2} - A_{i,j-1/2,k+1/2}}{\Delta\eta}\right]\right. \\
&\qquad \left. - \left[\frac{B_{i+1/2,j,k-1/2} - B_{i-1/2,j,k-1/2}}{\Delta\xi} - \frac{A_{i,j+1/2,k-1/2} - A_{i,j-1/2,k-1/2}}{\Delta\eta}\right]\right) \\
\equiv &\; 0.
\end{aligned}$$

Note that this is completely independent of the discretization of A, B, and C. The cell-averaged divergences of $J\partial\xi^m/\partial y$ and $J\partial\xi^m/\partial y$ can be made identically zero by similar construction.

We must now discretize J so that we can obtain the inverse Jacobian elements, $\partial\xi^m/\partial x^\alpha$. We are free to choose any useful, face-averaged discretization of J without sacrificing freestream preservation as long as we define the inverse Jacobian determinant via $1/J$. That is, the discretized versions of $1/J$ and J must be exact multiplicative inverses of one another.

To proceed, we consider the primary utility of J – to compute volume integrals. $J\,d\xi\,d\eta\,d\zeta = dx\,dy\,dz$ is the volume element in the curvilinear coordinate system and $J\,\Delta\xi\,\Delta\eta\,\Delta\zeta$ is the volume of a cell. To maintain conservation laws, all cell-averaged quantities must be refined and coarsened using operators that weight the values in each cell by J. Also, the values of J on the coarse grid must be identical to the (unweighted)

averages of the overlying values of J on the finer grids. However, J itself may be computed using any second order stencil that is convenient. Therefore, we simply compute the cell-centered determinant of the Jacobian matrix from the nodal coordinate data via common second order derivative stencils. Whenever the face-centered Jacobian is needed, we use an average of the neighboring cell-centered values.

Once J is known, the inverse Jacobian matrix elements also become available. From this, we can compute the metric tensor and its inverse,

$$g_{mn} = \sum_{\alpha} \frac{\partial x^{\alpha}}{\partial \xi^{m}} \frac{\partial x^{\alpha}}{\partial \xi^{n}} \qquad \text{and} \qquad g^{mn} = \sum_{\alpha} \frac{\partial \xi^{m}}{\partial x^{\alpha}} \frac{\partial \xi^{n}}{\partial x^{\alpha}}.$$

The metric tensor itself turns out to not be of dynamical interest, whereas the inverse metric tensor can be found throughout eqs. 1.20, both implicitly and explicitly. As it turns out, g^{mn} will only be needed at cell faces, $\mathcal{G}'_m$. Therefore, the off-diagonal entries require inverse Jacobian elements that must be computed using special derivative stencils.

Section 2.2: Divergence

For eq. 2.4 to be upheld, the divergence operator must be a mappping from $\mathcal{G}'_m$ to $\mathcal{G}'$, applying itself to face-averaged quantities to produce cell-averaged quantities. We define the discrete divergence on a single AMR level as

$$\left[\frac{1}{J}\partial_m F^m\right]_{i,j,k} \approx \left[\mathbf{D}\vec{F}\right]_{i,j,k} \tag{2.9}$$

$$= \frac{1}{J_{i,j,k}} \left(\frac{F^{\xi}_{i+1/2,j,k} - F^{\xi}_{i-1/2,j,k}}{\Delta\xi} + \frac{F^{\eta}_{i,j+1/2,k} - F^{\eta}_{i,j-1/2,k}}{\Delta\eta} + \frac{F^{\zeta}_{i,j,k+1/2} - F^{\zeta}_{i,j,k-1/2}}{\Delta\zeta} \right). \tag{2.10}$$

If we compute the divergence on a coarse AMR level, l, and then compute the divergence on a finer AMR level, $l+1$, we must correct the coarse-level divergence to account for the fine-level information since the fine-level information is assumed to be more accurate. This correction is provided in two steps. First we average the results of the fine-level divergence down to the coarser-level. Our averaging operator is made conservative by weighting the fine-level data by J^{l+1}, averaging down, then dividing the coarse-level results by the sum of the J^{l+1} weightings. Second, we correct the coarse-level divergence in those cells that are on the coarse side of the coarse-fine (CF) interface. For example, if the fine-level fluxes are known at the upper

ξ boundary of a coarse-level cell, then the correction to that cell's divergence is

$$\mathbf{D}_R \vec{F}^l_{i,j,k} = \pm \frac{1}{J_{i,j,k}} \left(\frac{\sum_I F_I^{\xi,l+1} - F^{\xi,l}_{i+1/2,j,k}}{\Delta \xi} \right), \tag{2.11}$$

where I represents the indices of the faces that hold the fine-level fluxes and l or $l+1$ denotes the data's AMR level. Corrections on the lower cell boundaries use the minus sign, while corrections on the upper cell boundaries use the plus sign. Since $\vec{F}$ is a flux, it is assumed to be scaled by J due to our redefinition of the divergence operator. These corrections are accumulated over the domain then applied by

$$\mathbf{D}\vec{F}^l \leftarrow \mathbf{D}\vec{F}^l + \mathbf{D}_R F^l. \tag{2.12}$$

The correction term, $\mathbf{D}_R \vec{F}^l$ is called the *reflux divergence* and the act of correcting the coarse-level divergence with fine-level fluxes via eq. 2.12 is called *explicit refluxing*.

If we are refluxing a diffused quantity, the we must diffuse the correction before applying it to $\mathbf{D}\vec{F}^l$. We perform the *implicit refluxing* using a backwards Euler timestep. For example, suppose $\vec{F}^l$ is a buoyancy flux and κ is its (thermal) diffusivity. Then we solve

$$(1 - \Delta t^l \kappa \nabla^2) \delta \vec{F}^l = \Delta t^l \mathbf{D}_R \vec{F}^l,$$

using homogeneous BCs at the domain boundaries. If the coarse level, l, has a CF interface with another coarser level, $l-1$, then we assume that $\delta \vec{F}^{l-1} = 0$ and use quadratic interpolation to fill the CF interface ghosts of $\delta \vec{F}^l$. The diffused update is then applied by

$$\mathbf{D}\vec{F}^l \leftarrow \mathbf{D}\vec{F}^l + \delta F^l. \tag{2.13}$$

Implicit refluxing not only promotes accuracy at the CF interfaces, but also ensures the stability of hyperbolic conservation laws.

Finally, we point out that while refluxing was developed to conserve fluid properties during hyperbolic updates, it turns out to be absolutely crucial to the convergence of multi-level elliptic and parabolic solvers. For a thorough analysis of refluxing and its effects on hyperbolic and elliptic updates, the reader is referred to the works of Dan Martin [49, 50] and Ann Almgren [51].

Section 2.3: Gradient

The discretized gradient must take objects in $\mathcal{G}'$ to $\mathcal{G}'_m$. Since the gradient's input is cell-centered, we must first fill ghost cells by applying the problem's Dirichlet, periodic, and CF interface BCs. Dirichlet conditions are applied by linear extrapolation,

$$\phi_{\mathrm{G}} = 2B - \phi_{\mathrm{N}}, \tag{2.14}$$

where B is the boundary value, ϕ_{G} is the ghost cell value being set, and ϕ_{N} is its nearest neighbor inside the domain. Periodic conditions are applied by a direct copy of data into the ghost cells on opposite ends of the domain, usually requiring an *exchange* (communication across processors). CF interface BCs are a special form of Dirichlet BCs. These BCs fill fine-level ghosts through quadratic interpolations of coarse-level data and avoid the use of data that is covered by a finer level (called *invalid* data). Neumann BCs are set after the gradient has been calculated.

The ξ-component of the discrete gradient operator is

$$\begin{aligned}
\left[Jg^{\xi m}\partial_m\phi\right]_{i+1/2,j,k} &\approx [\mathbf{G}\phi]^{\xi}_{i+1/2,j,k} \\
&= (Jg^{\xi\xi})_{i+1/2,j,k}\left(\frac{\phi_{i+1,j,k}-\phi_{i,j,k}}{\Delta\xi}\right) \\
&+ (Jg^{\xi\eta})_{i+1/2,j,k}\left(\frac{\tilde{\phi}_{i+1,j+1,k}-\tilde{\phi}_{i+1,j-1,k}+\tilde{\phi}_{i,j+1,k}-\tilde{\phi}_{i,j-1,k}}{4\Delta\eta}\right) \\
&+ (Jg^{\xi\zeta})_{i+1/2,j,k}\left(\frac{\tilde{\phi}_{i+1,j,k+1}-\tilde{\phi}_{i+1,j,k-1}+\tilde{\phi}_{i,j,k+1}-\tilde{\phi}_{i,j,k-1}}{4\Delta\zeta}\right).
\end{aligned}$$

To avoid conflicts at corner ghost cells by potentially competing BCs, we create $\tilde{\phi}$, a copy of ϕ with all boundary ghosts linearly extrapolated. We assume that ϕ is a smooth function and that the direction of extrapolation is not an issue. This method of extrapolating ghost data to compute boundary derivatives is identical to using biased finite differencing stencils.

Neumann boundary conditions do not dictate the values of the partial derivatives, $\partial_\xi\phi$, as they do in Cartesian coordinates, but rather dictate the values of the fluxes, $Jg^{\xi m}\partial_m\phi$. Using these BCs to generate ghost data has its drawbacks. It requires the solution to several elliptic equations each time we want to fill ghosts, which serves as a severe bottleneck in our iterative schemes. It is also difficult to fill ghosts in such a way that upon applying the gradient, we retrieve the boundary fluxes exactly, which represents a stability issue. For these reasons, we take the simpler, economical, and more stable approach of setting the boundary flux values directly on the boundary faces after computing the gradient in the interior.

Once the ξ-component of the gradient is computed, we must repeat the process for the η and ζ components. This does not require any additional communication, ghost fills, or extrapolations, provided $\tilde{\phi}$ remains in memory.

Section 2.4: Operator adjoints

Consider an arbitrary (but sufficiently smooth) scalar field, ϕ, and flux field, $\vec{F}$, with a vanishing outward normal flux, $\vec{F}\cdot\hat{n}$, at the domain boundaries. Analytically, it is straightforward to show

$$\begin{aligned}
\left\langle \phi, \nabla\cdot\vec{F} \right\rangle &= \int_{\mathcal{V}} \phi \nabla\cdot\vec{F}\, dV \\
&= \int_{\mathcal{V}} \nabla\cdot\left(\phi\vec{F}\right) dV - \int_{\mathcal{V}} (\nabla\phi)\cdot\vec{F}\, dV \\
&= \oint_{\partial\mathcal{V}} \phi\vec{F}\cdot\hat{n}\, dS - \int_{\mathcal{V}} (\nabla\phi)\cdot\vec{F}\, dV \\
&= 0 - \int_{\mathcal{V}} (\nabla\phi)\cdot\vec{F}\, dV \\
&= \left\langle -\nabla\phi, \vec{F} \right\rangle .
\end{aligned}$$

The gradient operator is therefore the negative adjoint of the divergence operator. We wish this to be the case in the discrete sense as well, where the inner product is a sum rather than an integral. This requirement can be written as $\mathbf{D} \equiv -\mathbf{G}^T$.

Consider the discrete analog of $\left\langle \phi, \nabla\cdot\vec{F} \right\rangle$,

$$\begin{aligned}
\left\langle \phi, \mathbf{D}\vec{F} \right\rangle &= \sum_{\mathcal{G}'} \phi_{i,j,k} \left(\mathbf{D}\vec{F}\right)_{i,j,k} J_{i,j,k}\, \Delta\xi\, \Delta\eta\, \Delta\zeta \\
&= \Big([F^\xi \text{ terms}] + [F^\eta \text{ terms}] + [F^\zeta \text{ terms}] \Big)\, \Delta\xi\, \Delta\eta\, \Delta\zeta .
\end{aligned}$$

Let us just focus on the F^ξ terms. If the boundaries are at infinity, the we can write

$$\begin{aligned}
[F^\xi \text{ terms}] &= \cdots + \frac{\phi_{i-1,j,k} F^\xi_{i-1/2,j,k}}{\Delta\xi} - \frac{\phi_{i,j,k} F^\xi_{i-1/2,j,k}}{\Delta\xi} + \frac{\phi_{i,j,k} F^\xi_{i+1/2,j,k}}{\Delta\xi} - \frac{\phi_{i+1,j,k} F^\xi_{i+1/2,j,k}}{\Delta\xi} + \cdots \\
&= \cdots + \frac{\phi_{i-1,j,k} - \phi_{i,j,k}}{\Delta\xi} F^\xi_{i-1/2,j,k} + \frac{\phi_{i,j,k} - \phi_{i+1,j,k}}{\Delta\xi} F^\xi_{i+1/2,j,k} + \cdots \\
&= \sum_{\mathcal{G}'_\xi} (-\mathbf{G}\phi)_\xi\, F^\xi .
\end{aligned}$$

Similar formulas hold for the F^η and F^ζ terms. Therefore,

$$\begin{aligned}\left\langle \phi, \mathbf{D}\vec{F} \right\rangle &= \left(\sum_{\mathcal{G}'_\xi} (-\mathbf{G}\phi)_\xi F^\xi + \sum_{\mathcal{G}'_\eta} (-\mathbf{G}\phi)_\eta F^\eta + \sum_{\mathcal{G}'_\zeta} (-\mathbf{G}\phi)_\zeta F^\zeta \right) \Delta\xi\,\Delta\eta\,\Delta\zeta \\ &= \sum_{\text{all faces}} (-\mathbf{G}\phi)\cdot\vec{F}\,\Delta\xi\,\Delta\eta\,\Delta\zeta \\ &= \left\langle -\mathbf{G}\phi, \vec{F} \right\rangle,\end{aligned}$$

and we have recovered the desired result. Note that this holds true irrespective of the discretization of the non-orthogonal terms in the gradient ($g^{\xi\eta}\partial_\eta$, $g^{\xi\zeta}\partial_\zeta$, etc.), which is beneficial since we (1) opted to compute these non-orthogonal terms using biased stencils at the domain boundaries and (2) required special derivative stencils to construct the off-diagonal elements of the inverse metric tensor.

Now, suppose we are at the upper ξ boundary. We can write out the discretizations of both $<\phi, \mathbf{D}\vec{F}>$ and $<-\mathbf{G}\phi, \vec{F}>$ explicitly. In just one dimension with $\Delta\xi$ scaled to 1, we have

$$\begin{array}{lcccccccc} \left\langle \phi, \mathbf{D}\vec{F} \right\rangle & = & \cdots & + & \phi_{i-2}\left(F^\xi_{i-3/2} - F^\xi_{i-5/2}\right) & + & \phi_{i-1}\left(F^\xi_{i-1/2} - F^\xi_{i-3/2}\right) & + & \phi_i\left(F^\xi_{i+1/2} - F^\xi_{i-1/2}\right) \\ \left\langle -\mathbf{G}\phi, \vec{F} \right\rangle & = & \cdots & - & \left(\phi_{i-1} - \phi_{i-2}\right)F^\xi_{i-3/2} & - & \left(\phi_i - \phi_{i-1}\right)F^\xi_{i-1/2} & - & \left(\phi_{i+1} - \phi_i\right)F^\xi_{i+1/2}. \end{array}$$

Subtracting these two, we do not immediately get zero, but instead find

$$\left\langle \phi, \mathbf{D}\vec{F} \right\rangle - \left\langle -\mathbf{G}\phi, \vec{F} \right\rangle = \phi_{i+1} F^\xi_{i+1/2}.$$

If the outward normal flux at the boundary is not zero, then the boundary term does not go to zero, but depends on the BCs of ϕ and how the ghost extrapolation is implemented. On the other hand, if the outward normal flux does equal zero at the boundary, then by definition $F^\xi_{i+1/2} = 0$ and we again recover $\mathbf{D} \equiv -\mathbf{G}^T$. Whenever this result is needed, we will require $\vec{F}\cdot\hat{n} = 0$ at the boundaries. This result also holds in higher dimensions.

Unfortunately, the results of this section do not prove true for composite (multi-level) operators at CF interfaces. This is largely due to the gradient's ghost interpolation stencils and, to a much lesser extent, the divergence's refluxing mechanism. At the moment, no solution has been found to correct or circumvent these issues. If the dynamics at the CF interface are observed to be sacrificed by a composite projection of the velocity field (see sec. 4.4), then further research into mimetic interpolation stencils would be warranted.

Section 2.5: The discrete Hodge-Helmholtz decomposition

The Hodge-Helmholtz decomposition of a vector field, $\vec{u}^\star$, is

$$\vec{u}^\star = \nabla \times \vec{A} + \nabla\phi.$$

For our purposes, we do not wish to construct a discrete curl, $\mathbf{C}$, that mimics the properties of its continuous counterpart, $\mathbf{DC} \equiv 0$, $\mathbf{CG} \equiv 0$, and $\mathbf{C} \equiv -\mathbf{C}^T$. To do so would potentially require a major overhaul of our divergence and gradient discretizations, which would in turn effect the freestream preserving construction of our metric elements. Instead, we rely on an alternate form of the decomposition,

$$\begin{aligned} \vec{u}^\star &= \vec{u} + \nabla\phi \\ \nabla \cdot \vec{u} &= 0, \end{aligned} \tag{2.15}$$

where $\vec{u}$ is by definition a soleniodal, or divergence-free vector field. It is worth pointing out that the only reason we do not need a mimetic $\mathbf{C}$ is because the curl does not show up in our formulation of the dynamical equations 1.20.

The discrete version of eq. 2.15 is

$$\begin{aligned} \vec{u}^\star &= \vec{u} + \mathbf{G}\phi \\ \mathbf{D}\vec{u} &= 0, \end{aligned} \tag{2.16}$$

where $\vec{u}$ is now understood to be defined on the grid's face-centers and weighted by J. This equation mimics all of the necessary properties of the continuous Hodge-Helmholtz decomposition. First, $\vec{u}$ and $\mathbf{G}\phi$ belong to orthogonal vector spaces since

$$\langle \mathbf{G}\phi, \vec{u} \rangle = -\langle \phi, \mathbf{D}\vec{u} \rangle = 0,$$

meaning the solenoidal and irrotational terms of the decomposition can be thought of as orthogonal components of $\vec{u}^\star$. Second, the energy put into a system by a timestepping algorithm can be written as

$$\frac{1}{2}\langle \vec{u}^\star, \vec{u}^\star \rangle = \frac{1}{2}\langle \vec{u}, \vec{u} \rangle + \frac{1}{2}\langle \mathbf{G}\phi, \mathbf{G}\phi \rangle .$$

Therefore, if we remove the irrotational component, $\mathbf{G}\phi$, we will not remove or alter the component of the energy that owes itself to the dynamical velocity. Third, we can remove the irrotational component of $\vec{u}^\star$ by solving Poisson's equation,

$$\begin{aligned} \mathbf{DG}\phi &= \mathbf{D}\vec{u}^\star \\ \vec{u} &= \vec{u}^\star - \mathbf{G}\phi. \end{aligned}$$

For brevity, this operation is called a *velocity projection*.

The idempotent operator $\mathbf{P} = 1 - \mathbf{G}(\mathbf{DG})^{-1}\mathbf{D}$ is a projection operator. It exacly removes the irrotational component of vectors in the staggered space $\mathcal{G}'_m$. All cell-centered quantities (buoyancy and momentum) will be advected using an *advecting velocity*, $u_{\mathrm{AD}} \in \mathcal{G}'_m$, that has been projected in this exact manner. However, since our advection scheme is a finite volume method, we must update a cell-averaged velocity. (Since cell-averaged quantities are also cell-centered quantities to second order in the grid spacing, we will simply call the cell-averaged velocity the CC velocity.) This requires non-staggered versions of the gradient and divergence operators. We will call these the CC gradient and CC divergence operators. To construct these operators, we introduce two averaging operators that change the centering of grid data, $\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}$ and $\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}$. These are defined componentwise via

$$\begin{aligned}
\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}[u]_{i,j,k} &= \frac{1}{2}\left(u_{i+1/2,j,k} - u_{i-1/2,j,k}\right) \\
\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}[v]_{i,j,k} &= \frac{1}{2}\left(v_{i,j+1/2,k} - v_{i,j-1/2,k}\right) \\
\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}[w]_{i,j,k} &= \frac{1}{2}\left(w_{i,j,k+1/2} - w_{i,j,k-1/2}\right)
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[u]_{i+1/2,j,k} &= \frac{1}{2}\left(u_{i+1,j,k} - u_{i-1,j,k}\right) \\
\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[v]_{i,j+1/2,k} &= \frac{1}{2}\left(v_{i,j+1,k} - v_{i,j-1,k}\right) \\
\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[w]_{i,j,k+1/2} &= \frac{1}{2}\left(w_{i,j,k+1} - w_{i,j,k-1}\right),
\end{aligned}$$

where $\vec{u} = u\hat{\xi} + v\hat{\eta} + w\hat{\zeta}$ and the centerings are implicitly declared by the cell/face indices. With these averaging operators, the CC gradient of a CC scalar, ϕ, and the CC divergence of a CC velocity, $\vec{u}$ are defined by

$$\begin{aligned}
\mathbf{G}^{\mathrm{CC}}\phi &= \mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}[\mathbf{G}\phi] \\
\mathbf{D}^{\mathrm{CC}}\vec{u} &= \mathbf{D}\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[\vec{u}].
\end{aligned}$$

Our CC projector is now defined by $\mathbf{P}^{\mathrm{CC}} = 1 - \mathbf{G}^{\mathrm{CC}}(\mathbf{DG})^{-1}\mathbf{D}^{\mathrm{CC}}$. This is not idempontent since $(\mathbf{DG})^{-1}$ is not the inverse of $\mathbf{D}^{CC}\mathbf{G}^{CC}$. The CC projector is therefore only an approximate projector and does not completely remove the irrotational component of a CC vector field in one application. It has been shown that the composite operator $(\mathbf{P}^{CC})^n$ tends towards becoming an exact projector in the limit $n \to \infty$ and

that this limit is well defined and stable, but only one application of $\mathbf{P}^{CC}$ is sufficient to produce a stable timestepping algorithm [49].

Upon application of the CC projector to a velocity field, we see that by definition of $\mathbf{D}^{\mathrm{CC}}$,

$$\begin{aligned}\mathbf{P}^{\mathrm{CC}}\vec{u} &= \vec{u} - \mathbf{G}^{\mathrm{CC}}(\mathbf{DG})^{-1}\mathbf{D}^{\mathrm{CC}}\,\vec{u} \\ &= \vec{u} - (\text{various operations})\,\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[\vec{u}].\end{aligned}$$

In words, when we project a CC velocity field, $\vec{u}$, we are actually removing the irrotational component of the filtered velocity, $\mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}[\vec{u}]$. The filtered modes that cannot be removed from the velocity are of the highest frequency supported by the grid and appear as checkerboards or stripes of covergent/divergent fluid. Attempts at filtering these modes with a Shapiro filter [52] and a compact, 4th-order filter derived by Lele [53] have proven to be an unstable detriment to the dynamics. A truly mimetic discretization may prove to be a more fruitful option, since it would preclude the need for FC and CC sets of velocities while reducing the stencil size of the Laplacian operator. But these efforts have not yet been explored for use in SOMAR. We find that simply providing adequate resolution at the base level (the coarsest AMR level) reduces the high frequency modes provided to the CC projector in the first place.

As a final note, attempts have been made to construct an idempotent, exact CC projector, but without much success. The simplest approach would be to use all CC operators in the projector's construction, $\mathbf{P}^{\mathrm{CC}} = 1 - \mathbf{G}^{\mathrm{CC}}(\mathbf{D}^{\mathrm{CC}}\mathbf{G}^{\mathrm{CC}})^{-1}\mathbf{D}^{\mathrm{CC}}$. This, however, checkerboards the grid due to the wide stencil of the CC Laplacian, $\mathbf{D}^{\mathrm{CC}}\mathbf{G}^{\mathrm{CC}}$, again promoting errors of the highest frequencies supported by the grid.

Section 2.6: Curl

The discretization of the curl has no significant dynamical impact due to its absence in the equations of motion 1.20. Primarily, we use the curl to compute the vorticity of the cell-centered velocity field for diagnostic purposes. The ζ-component of discrete, cell-centered curl, $\mathbf{C}^{\mathrm{CC}}$, is defined by

$$\left(\mathbf{C}^{\mathrm{CC}}\vec{u}\right)^{\zeta} = \frac{\mathscr{V}_{i+1,j,k} - \mathscr{V}_{i-1,j,k}}{2\Delta\xi} - \frac{\mathscr{U}_{i,j+1,k} - \mathscr{U}_{i,j-1,k}}{2\Delta\eta},$$

where $\mathscr{U}$ and $\mathscr{V}$ are two of the covariant velocity components (and not scaled by J). In 2D, this is the entire curl operator. In 3D, other components are defined similarly.

CHAPTER 3: The Leptic Iterative Method[1]

Section 3.1: Introduction

The solution of Poisson problems is often the single most expensive step in the numerical solution of partial differential equations (PDEs). For example, when solving the Navier-Stokes or Euler equations, the Poisson problem arises from the incompressibility condition [54, 55]. The particular solution strategy depends of course on a combination of factors, including the specific choice of the discretization and the type of boundary conditions. In simple geometries, very efficient schemes can be devised to reduce the effective dimensionality of the problem, such as using FFTs or cyclic reduction to partially or completely diagonalize the operator [55]. For more complex geometries or boundary conditions, the available choices to solve the discretized problem usually involves direct inversion for small size problems, while Krylov based iterative methods such as Conjugate Gradient or multigrid methods are used when the matrix problem is too large to be inverted exactly [55, 56, 57].

In this chapter, we are interested in Poisson problems characterized by a high level of anisotropy (to be precisely defined later). The source of anisotropy can be due to the highly flattened domains over which the solution is sought, as is the case for atmospheric, oceanic [55] and some astrophysical problems [58, p. 77]. However, the source of anisotropy could have a physical base, e.g. Non-Fickian diffusion problems where the flux is related to the gradient by an anisotropic tensor [59] and MHD problems where the thermal conductivity of plasmas is influenced by magnetic field lines [60]. Recently, research has been conducted to develop compound materials that could serve as a cloaking device [61, 62]. These *metamaterials* are designed to have specific anisotropic acoustic and electromagnetic properties that divert pressure and light waves around a region of space unscathed.

Anisotropy results in a spreading of the spectrum of the discretized operator, with severe consequences on the convergence rate. We illustrate this point with a simple Poisson problem

$$\nabla^2 \phi = \rho.$$

The RHS and boundary conditions are chosen randomly (but compatible, see below). The Laplacian operator

[1] This chapter previously appeared as an article in the Journal of Computational Physics. The original citation is as follows: Edward Santilli and Alberto Scotti. An efficient method for solving highly anisotropic elliptic equations. *J. Comput. Phys.*, 230(23):8342–8359, September 2011.

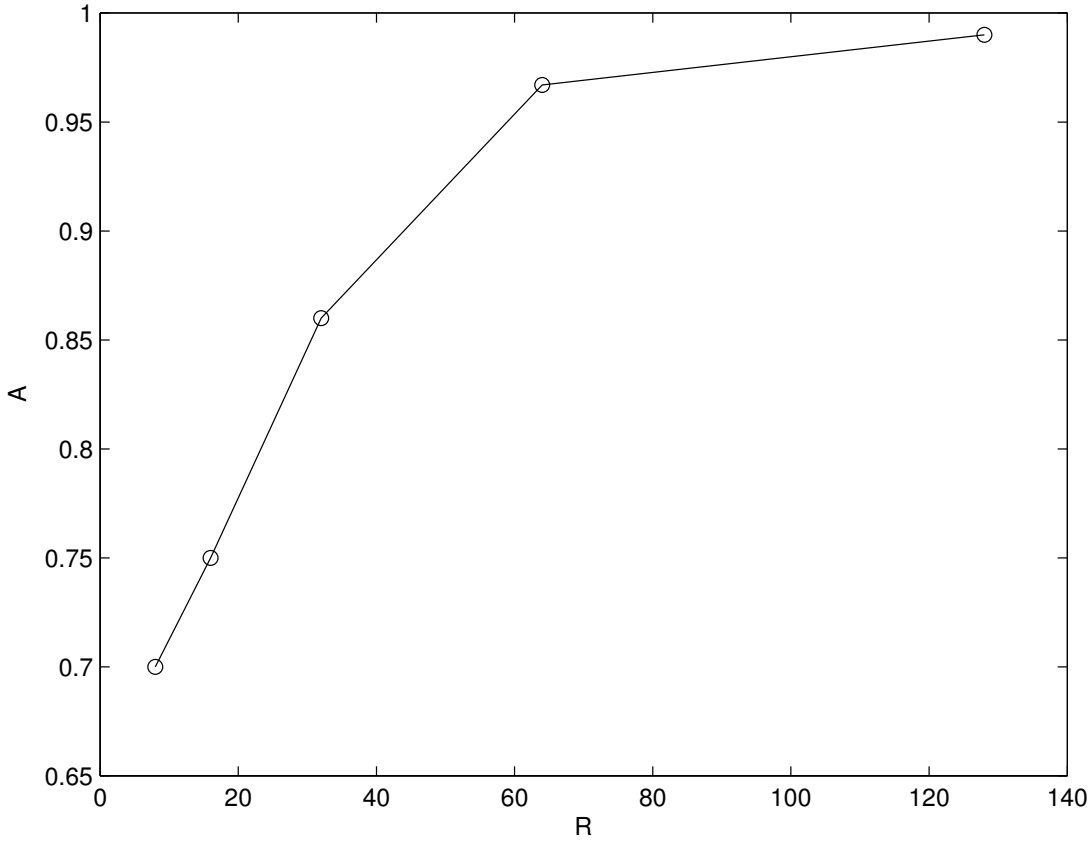


Figure 3.1: Attenuation factor as a function of domain aspect ratio for a Poisson problem solved used a standard multigrid scheme.

is discretized with a standard 7-point, second-order stencil, the domain is rectangular with dimensions $L \times L \times H$, the domain's aspect ratio is measured by $R = L/H$, and the discretization is chosen as $(\Delta x, \Delta y, \Delta z) = (H/2, H/2, H/16)$. For illustration purposes, the problem is solved using a 4-level multigrid scheme which employs line relaxation in the vertical (stiff) direction, as used by Armenio and Roman [63] to do a LES of a shallow coastal area. Figure 3.1 shows the attenuation factor $A = \|E_{n+1}\|/\|E_n\|$ as a function of the aspect ratio, where E_n is the residual error after n iterations. For moderate aspect-ratio domains, the convergence is satisfactory, but as R increases, we rapidly approach a point where the method becomes, for practical purposes, useless. Similar results (see below) hold for Krylov based methods.

There have been several attempts to produce valid and economical solution methods for strongly anisotropic elliptic equations. Perhaps most recently, P. Degond, *et. al.* discussed just such a method that can easily be extended to D-dimensional geometries [64, 65, 66]). Their method proves quite useful for problems that have homogeneous boundary conditions and anisotropies orthogonal to the remaining $(D-1)$-dimensional

physics. To appeal to the CFD interests of the authors, this paper provides a method that frees us from the restrictions of homogeneity and orthogonality. We will end up with an iterative scheme that splits the N-dimensional elliptic problem into $1+(D-1)$-dimensional elliptic problems. The 1-dimensional problem can be sent to a tridiagonal solver while the remaining $(D-1)$-dimensional elliptic problem can be sent to any existing algorithm the reader chooses. To further widen our audience, we will also analyze our algorithm's utility in preconditioning weakly anisotropic problems.

The method this chapter presents is based on a formal series solution of a Poisson problem derived by Scotti and Mitran [67], herein referred to as SM, that can be used to significantly speed up the convergence of traditional iterative schemes. SM introduced the concept of grid lepticity, λ, to describe the degree of anisotropy of a discretized domain and then sought a solution to the Poisson problem written as a power series in λ^{-1} – the leptic expansion. An apparent limitation of the leptic expansion is that it is very efficient only for lepticity larger than a critical value of order 1. SM were led to introduce the leptic expansion in order to provide the right amount of dispersion needed to balance nonlinear steepening of internal waves propagating in a shallow stratified ocean. For this limited purpose, SM showed that at most only three terms in the expansion are needed, and thus the lack of overall convergence was not a serious limitation. Here, we develop the method for the purpose of efficiently calculating solutions of a discretized Poisson problem. In our approach, the lepticity, which in SM's original formulation was related to the aspect ratio of the domain, becomes now a generic measure of anisotropy. The main result of this chapter is that for subcritical values of the lepticity, the leptic expansion can still be extremely valuable to dramatically increase the convergence rate of standard iterative schemes, as the numerical demonstrations of the method will show. The examples are coded using Matlab and Chombo's BoxTools library [68].

What makes the leptic expansion particularly attractive is that it can be parallelized in a very straightforward way, as long as the decomposition of the domain does not split along the stiff (vertical, in our examples) direction. For comparison, the parallel implementation of the Incomplete Cholesky Decomposition of a sparse matrix, which is used as a preconditioner for Conjugate Gradient schemes and yields very good convergence rates even at high levels of anisotropy [69], is a highly non-trivial task [70].

A known consequence of strong anisotropy is that standard centered differencing methods do not preserve monotonicity [60, 71]. The leptic expansion *per se* will not cure this problem if the underlying discretization employed does not preserve it in first place, though, since the expansion is discretization-independent, it could be possible to use a specialized discretization in conjuction with a suitably modified version of the expansion to preserve monotonicity. However, in the interest of simplicity, we will not pursue this argument any further. The non-monotonicity issue arises primarily when the coordinate system is not aligned with the direction of maximum anisotropy, a situation that we avoid here by using analytical methods to separate the

stiff direction from the remaining (horizontal) directions. When the time comes to employ a finite differencing method on the resulting horizontal Poisson problem, only a weak, coordinate (non-physical) anisotropy will remain.

Finally, it must be noted that the idea behind the leptic expansion can be traced as far back as the work of Boussinesq on waves [72]. What we have done here is to formulate it in a way suitable for numerical calculations.

The rest of the chapter is organized as follows: a discretization- and coordinate-independent version of the leptic expansion is introduced in Section 3.2. The reader who is not interested in the details may skip directly to Section 3.2.2, where a summary of the scheme is provided. Section 3.3 presents convergence estimates of the leptic expansion using Fourier analysis techniques [73]. This is where the leptic method's potential to generate initial guesses for conventional iterative schemes emerges. In Section 3.4, we consider some examples to illustrate how the leptic expansion can be used with conventional iterative schemes to create very efficient solvers. A final section summarizes the main results.

Section 3.2: Derivation

The problem we wish to solve is an anisotropic elliptic PDE with Neumann boundary conditions of the type that often arises when solving the incompressible Navier-Stokes equations [54, 55]. More precisely, we wish to solve the following equation

$$\begin{aligned} \partial_i \sigma^{ij} \partial_j \phi &= \rho \text{ in } \mathcal{V} \\ \sigma^{ij} \partial_j \phi &= u^{\star i} \text{ on } \partial\mathcal{V}, \end{aligned} \tag{3.1}$$

where σ^{ij} is a positive-definite, symmetric tensor field. The only restriction on ρ and $u^{\star i}$ is that they must be compatible with one another. That is, if we integrate eq. (3.1) over $\mathcal{V}$ and apply Stokes' theorem, we obtain an identity that must be obeyed by the sources,

$$\int_{\mathcal{V}} \rho d\mathcal{V} = \oint_{\partial\mathcal{V}} u^{\star i} d\mathcal{A}_i. \tag{3.2}$$

Also, in order to identify a unique solution to eq. (3.1), we will fix the volumetric average of ϕ to be zero. This does not limit the generality of the problem in any way, while allowing for several simplifications to be made later in our solution method.

In general, the domain can have any number of dimensions higher than 1, but without loss of generality we will restrict ourselves to the 3-dimensional case. We will also assume there is a small parameter, ε, that

we can use to identify terms of the field and operator in a formal perturbation expansion of the form

$$\begin{aligned} \phi &= \phi_0 + \varepsilon\phi_1 + \varepsilon^2\phi_2 + \varepsilon^3\phi_3 + \dots \\ \partial_i\sigma^{ij}\partial_j &= \partial_3\sigma^{33}\partial_3 + \varepsilon\left(\partial_m\sigma^{mn}\partial_n + \partial_m\sigma^{m3}\partial_3 + \partial_3\sigma^{3n}\partial_n\right). \end{aligned} \tag{3.3}$$

In section 3.2.4, we will show that if the anisotropy is due to the geometry and/or discretization of the domain, $\epsilon = H^2/\Delta x^2$, where H is the domain's length in the direction of the anisotropy and Δx is the minimum horizontal grid spacing. We should point out, however, that ϵ can more generally be used to identify small elements of the general, symmetric tensor, σ^{ij}. In this section, we will derive a method to solve eqs. (3.1) using the expansion (3.3).

3.2.1: The desired form of the expansion

We begin by plugging expansion (3.3) into the first of eqs. (3.1) and equating powers of ε.

$$\partial_3\sigma^{33}\partial_3\phi_0 = \rho \tag{3.4}$$

$$\partial_3\sigma^{33}\partial_3\phi_1 + \left(\partial_m\sigma^{m3}\partial_3 + \partial_3\sigma^{3n}\partial_n + \partial_m\sigma^{mn}\partial_n\right)\phi_0 = 0 \tag{3.5}$$

$$\partial_3\sigma^{33}\partial_3\phi_2 + \left(\partial_m\sigma^{m3}\partial_3 + \partial_3\sigma^{3n}\partial_n + \partial_m\sigma^{mn}\partial_n\right)\phi_1 = 0 \tag{3.6}$$

$$\text{etc}\dots$$

The first thing to notice is that eq. (3.4) with Neumann boundary conditions can only determine ϕ_0 up to an additive function of x and y. We will call the solution of eq. (3.4) $\phi_0^v(x,y,z)$ and the still undetermined function $\phi_0^h(x,y)$. We might as well preemptively write the fields at every order as

$$\phi_n(x,y,z) = \phi_n^v(x,y,z) + \phi_n^h(x,y)$$

so that equations (3.4)-(3.6) read

$$\partial_3\sigma^{33}\partial_3\phi_0^v = \rho \tag{3.7}$$

$$\partial_3\sigma^{33}\partial_3\phi_1^v + \partial_m\sigma^{m3}\partial_3\phi_0^v + \left(\partial_3\sigma^{3n}\partial_n + \partial_m\sigma^{mn}\partial_n\right)\phi_0 = 0 \tag{3.8}$$

$$\partial_3\sigma^{33}\partial_3\phi_2^v + \partial_m\sigma^{m3}\partial_3\phi_1^v + \left(\partial_3\sigma^{3n}\partial_n + \partial_m\sigma^{mn}\partial_n\right)\phi_1 = 0 \tag{3.9}$$

$$\text{etc}\dots$$

In order to solve this set of equations, we must define our boundary conditions at each order. At $\mathcal{O}(1)$,

we will set

$$\begin{aligned}\sigma^{33}\partial_3\phi_0^v\big|_{z_+} &= w^\star|_{z_+} - \tilde{w}_0\\ \sigma^{33}\partial_3\phi_0^v\big|_{z_-} &= w^\star|_{z_-},\end{aligned}$$

where z_+ and z_- denote the evaluation at the upper and lower boundaries, respectively. The excess function, $\tilde{w}_0$, is defined at each x and y to make eq. (3.7) consistent with its boundary conditions. By vertically integrating eq. (3.7), we see that

$$\begin{aligned}\int_{z_-}^{z_+}\rho dz &= \sigma^{33}\partial_3\phi_0^v\big|_{z_-}^{z_+}\\ &= w^\star|_{z_-}^{z_+} - \tilde{w}_0,\end{aligned}$$

that is,

$$\tilde{w}_0 = w^\star|_{z_-}^{z_+} - \int_{z_-}^{z_+}\rho dz.$$

This completely determines ϕ_0^v along vertical lines at each x and y. Notice that we have not yet chosen the gradients of ϕ_0^v normal to the horizontal boundaries. We will save this freedom for later. Right now, we need to look at the $\mathcal{O}(\varepsilon)$ equations to get ϕ_0^h.

For the moment, let us think of eq. (3.8) as an equation for ϕ_1^v. We again need to specify vertical boundary conditions. We will define

$$\begin{aligned}\sigma^{33}\partial_3\phi_1^v\big|_{z_+} &= \tilde{w}_0 - \sigma^{3n}\partial_n\phi_0\big|_{z_+}\\ \sigma^{33}\partial_3\phi_1^v\big|_{z_-} &= -\sigma^{3n}\partial_n\phi_0\big|_{z_-}.\end{aligned} \tag{3.10}$$

Defining a new excess function is unnecessary because it can just be absorbed into the still undetermined function ϕ_0^h. As before, we vertically integrate eq. (3.8).

$$\begin{aligned}0 &= \left[\sigma^{33}\partial_3\phi_1^v + \sigma^{3n}\partial_n\phi_0\right]_{z_-}^{z_+} + \int_{z_-}^{z_+}\left(\partial_m\sigma^{m3}\partial_3\phi_0^v + \partial_m\sigma^{mn}\partial_n\phi_0\right)dz\\ &= \tilde{w}_0 + \int_{z_-}^{z_+}\left(\partial_m\sigma^{m3}\partial_3\phi_0^v + \partial_m\sigma^{mn}\partial_n\phi_0\right)dz\\ &= \tilde{w}_0 + \int_{z_-}^{z_+}\left(\partial_m\sigma^{m3}\partial_3\phi_0^v + \partial_m\sigma^{mn}\partial_n\phi_0^v\right)dz + \int_{z_-}^{z_+}\partial_m\sigma^{mn}\partial_n\phi_0^h dz\\ &= \tilde{w}_0 + \int_{z_-}^{z_+}\partial_m\sigma^{mj}\partial_j\phi_0^v dz + \int_{z_-}^{z_+}\partial_m\sigma^{mn}\partial_n\phi_0^h dz\end{aligned}$$

If we divide by H, the vertical integrals become vertical averages, which will be denoted with overbars. When taking these averages, remember that while ϕ^h was defined to be independent of z, no such assumption was made for σ^{ij}. This leaves us with an equation for ϕ_0^h,

$$\partial_m \overline{\sigma^{mn}} \partial_n \phi_0^h = -\frac{\tilde{w}_0}{H} - \partial_m \overline{\sigma^{mj} \partial_j \phi_0^v}. \tag{3.11}$$

If ϕ_0^h is chosen to be any solution to this equation, then eq. (3.8) for ϕ_1^v together with the boundary conditions (3.10) will be consistent. Now, we define boundary conditions for eq. (3.11) to be

$$\overline{\sigma^{mn}} \partial_n \phi_0^h \Big|_{\vec{x} \in \partial \mathcal{S}} = \overline{u^{\star m}}\Big|_{\vec{x} \in \partial \mathcal{S}}. \tag{3.12}$$

This choice of boundary condition will be made consistent with equation (3.11) in the following steps. First, we integrate eq. (3.11) over $\mathcal{S}$ and reorganize the result.

$$\begin{aligned}
-\frac{1}{H}\int_{\mathcal{S}} \tilde{w}_0 d\mathcal{S} - \oint_{\partial \mathcal{S}} \overline{\sigma^{mj} \partial_j \phi_0^v} dl_m &= -\int_{\mathcal{S}} \frac{\tilde{w}_0}{H} d\mathcal{S} - \int_{\mathcal{S}} \partial_m \overline{\sigma^{mj} \partial_j \phi_0^v} d\mathcal{S} \\
&= \int_{\mathcal{S}} \partial_m \overline{\sigma^{mn}} \partial_n \phi_0^h d\mathcal{S} \\
&= \oint_{\partial \mathcal{S}} \overline{\sigma^{mn}} \partial_n \phi_0^h dl_m \\
&= \oint_{\partial \mathcal{S}} \overline{u^{\star m}} dl_m
\end{aligned}$$

Next, we exploit the remaining freedoms in the boundary conditions of ϕ_0^v by choosing $\sigma^{mj} \partial_j \phi_0^v = u^{\star m} - \overline{u^{\star m}}$ at the horizontal boundaries of $\mathcal{V}$. This, together with the definition of $\tilde{w}_0$, gives us

$$-\frac{1}{H}\int_{\mathcal{S}} \left(w^\star|_{z_-}^{z_+} - \int_{z_-}^{z_+} \rho dz \right) d\mathcal{S} - \oint_{\partial S} \overline{\left(u^{\star m} - \overline{u^{\star m}}\right)} dl_m = \oint_{\partial \mathcal{S}} \overline{u^{\star m}} dl_m.$$

Noting that the third and fourth terms cancel, this simplifies to become

$$H \oint_{\partial \mathcal{S}} \overline{u^{\star m}} dl_m + \int_{\mathcal{S}} w^\star|_{z_-}^{z_+} d\mathcal{S} = \int_{\mathcal{V}} \rho d\mathcal{V}.$$

Now, if this equation holds, then the boundary conditions (3.12) will be consistent with the horizontal equation (3.11). From equation (3.2), we see that this is indeed the case. Therefore, equations (3.11) and (3.12) completely determine ϕ_0^h, and in turn, ϕ_0. Having ϕ_0 at our disposal, we can now tidy up eq. (3.8) a

bit.

$$\begin{aligned}\partial_3\sigma^{33}\partial_3\phi_1^v &= -\partial_m\sigma^{m3}\partial_3\phi_0^v - \left(\partial_m\sigma^{mn}\partial_n + \partial_3\sigma^{3n}\partial_n\right)\phi_0 \\ &= \left(\rho - \partial_3\sigma^{33}\partial_3\phi_0\right) - \partial_m\sigma^{m3}\partial_3\phi_0 - \left(\partial_m\sigma^{mn}\partial_n + \partial_3\sigma^{3n}\partial_n\right)\phi_0 \\ &= \rho - \partial_i\sigma^{ij}\partial_j\phi_0\end{aligned}$$

The first two terms in parentheses in the second line are zero via eq. (3.7). Since this equation along with boundary conditions (3.10) are consistent, this completely determines ϕ_1^v at each x and y. There is, as before, another freedom yet to be chosen – the gradients of ϕ_1^v normal to the horizontal boundaries. Again, we will choose them later.

We continue in the same manner, obtaining an equation for ϕ_2^v whose Neumann compatibility condition is the equation for ϕ_1^h. At this point, we might as well just derive the equations for the general order fields ϕ_p^v and ϕ_{p-1}^h where $p \geq 2$. We start with

$$\partial_3\sigma^{33}\partial_3\phi_p^v + \partial_m\sigma^{m3}\partial_3\phi_{p-1}^v + \left(\partial_m\sigma^{mn}\partial_n + \partial_3\sigma^{3n}\partial_n\right)\phi_{p-1} = 0 \tag{3.13}$$

and the vertical boundary conditions for ϕ_p^v

$$\sigma^{33}\partial_3\phi_p^v\big|_{z_\pm} = -\,\sigma^{3n}\partial_n\phi_{p-1}\big|_{z_\pm}.$$

Vertically averaging eq. (3.13) and rearranging a bit gives us

$$\frac{1}{H}\left[\sigma^{33}\partial_3\phi_p^v + \sigma^{3n}\partial_n\phi_{p-1}\right]_{z_-}^{z_+} + \overline{\partial_m\sigma^{mj}\partial_j\phi_{p-1}^v + \partial_m\sigma^{mn}\partial_n\phi_{p-1}^h} = 0,$$

which upon applying boundary conditions and rearranging some more gives us the horizontal equation

$$\partial_m\overline{\sigma^{mn}}\partial_n\phi_{p-1}^h = -\partial_m\overline{\sigma^{mj}\partial_j\phi_{p-1}^v}. \tag{3.14}$$

This equation will be compatible with homogeneous boundary conditions if we chose the gradients of ϕ_p^v

at the horizontal boundaries wisely. By integrating eq. (3.14) over $\mathcal{S}$, we find

$$\begin{aligned} 0 &= \oint_{\partial\mathcal{S}} \overline{\sigma^{mn}}\partial_n\phi^h_{p-1}dl_m \\ &= \int_{\mathcal{S}} \partial_m\overline{\sigma^{mn}}\partial_n\phi^h_{p-1}d\mathcal{S} \\ &= -\int_{\mathcal{S}} \partial_m\overline{\sigma^{mj}\partial_j\phi^v_{p-1}}d\mathcal{S} \\ &= -\oint_{\partial\mathcal{S}} \overline{\sigma^{mj}\partial_j\phi^v_{p-1}}dl_m. \end{aligned}$$

It is tempting now to let the gradients of ϕ^v_{p-1} for $p \geq 2$ be identically zero, but a more appropriate choice is

$$\sigma^{mj}\partial_j\phi^v_{p-1} = \left(\overline{\sigma^{mn}} - \sigma^{mn}\right)\partial_n\phi^h_{p-2}, \tag{3.15}$$

which is equivalent to

$$\sigma^{mj}\partial_j\phi^v_p = \begin{cases} \overline{u^{\star m}} - \sigma^{mn}\partial_n\phi^h_0, & p = 1 \\ -\sigma^{mn}\partial_n\phi^h_{p-1}, & p \geq 2. \end{cases}$$

This choice will average to zero, satisfying the above integral, as well as prevent inconsistencies later.

Finally, we can clean up eq. (3.13) to get an equation for ϕ^v_p.

$$\partial_3\sigma^{33}\partial_3\phi^v_p = \rho - \partial_i\sigma^{ij}\partial_j\left(\phi_0 + \phi_1 + \ldots + \phi_{p-1}\right)$$

This completes the derivation of the needed equations. It may seem like some of the boundary conditions were chosen only to be compatible with their corresponding differential equation, when in fact we chose them carefully so that the sum of their contributions is $u^{\star i}$ for each direction, i. At the horizontal boundaries,

$$\begin{aligned} \sigma^{mj}\partial_j\phi &= \left(\sigma^{mj}\partial_j\phi^v_0\right) + \left(\sigma^{mn}\partial_n\phi^h_0 + \sigma^{mj}\partial_j\phi^v_1\right) + \ldots + \left(\sigma^{mn}\partial_n\phi^h_{p-1} + \sigma^{mj}\partial_j\phi^v_p\right) + \ldots \\ &= \left(u^{\star m} - \overline{u^{\star m}}\right) + \left(\overline{\sigma^{mn}}\partial_n\phi^h_0\right) + \ldots + \left(\overline{\sigma^{mn}}\partial_n\phi^h_{p-1}\right) + \ldots \\ &= \left(u^{\star m} - \overline{u^{\star m}}\right) + \left(\overline{u^{\star m}}\right) + \ldots + (0) + \ldots \\ &= u^{\star m}, \end{aligned}$$

at the upper vertical boundary,

$$\begin{aligned}\sigma^{3j}\partial_j\phi\big|_{z_+} &= \left(\sigma^{33}\partial_3\phi_0^v\right) + \left(\sigma^{3n}\partial_n\phi_0 + \sigma^{33}\partial_3\phi_1^v\right) + \ldots + \left(\sigma^{3n}\partial_n\phi_{p-1} + \sigma^{33}\partial_3\phi_p^v\right) + \ldots \\ &= (w^\star - \tilde{w}_0) + (\tilde{w}_0) + \ldots + (0) + \ldots \\ &= w^\star,\end{aligned}$$

and at the lower vertical boundary,

$$\begin{aligned}\sigma^{3j}\partial_j\phi\big|_{z_-} &= \left(\sigma^{33}\partial_3\phi_0^v\right) + \left(\sigma^{3n}\partial_n\phi_0 + \sigma^{33}\partial_3\phi_1^v\right) + \ldots + \left(\sigma^{3n}\partial_n\phi_{p-1} + \sigma^{33}\partial_3\phi_p^v\right) + \ldots \\ &= (w^\star) + (0) + \ldots + (0) + \ldots \\ &= w^\star.\end{aligned}$$

3.2.2: Summary of the expansion

Tables (3.1) and (3.2) provide a concise overview of the steps involved in generating the n^{th}-order of the expansion in generalized and Cartesian coordinates, respectively. The problem is formulated recursively. The left hand side is the same at each step. However, it must be noted that the solution of the $(D-1)$-dimensional Poisson problem can be postponed or in some cases eliminated depending on the magnitude of the correction required. This will be discussed in section 3.2.3. For a very simple example problem solved using this expansion, see section 3.3.

3.2.3: Eliminating horizontal stages

Typically, solutions of the horizontal problems are more expensive than solutions of the vertical problems. Sometimes, we can skip the horizontal stages altogether if we know in advance that its solution will not contribute to the overall convergence of the method. For example, suppose

$$\sigma^{ij} = \begin{pmatrix} A(x,y) & D(x,y) & 0 \\ D(x,y) & B(x,y) & 0 \\ 0 & 0 & C(x,y,z) \end{pmatrix}$$

where A, B, C, and D are arbitrary functions of the variables listed. We see that

$$\partial_m\overline{\sigma^{mj}\partial_j\phi^v} = \partial_m\sigma^{mn}\partial_n\overline{\phi^v} + \partial_m\sigma^{m3}\overline{\partial_3\phi^v},$$

$\mathcal{O}(1)$	$\partial_3\sigma^{33}\partial_3\phi_0^v = \rho$	$\sigma^{33}\partial_3\phi_0^v\big\|_{z_\pm} = \begin{cases} w^\star\big\|_{z_+} - \tilde{w}_0, & z = z_+ \\ w^\star\big\|_{z_-}, & z = z_- \end{cases}$ $\sigma^{mj}\partial_j\phi_0^v\big\|_{\vec{x}\in\mathcal{A}_m} = \left[u^{\star m} - \overline{u^{\star m}}\right]_{\vec{x}\in\mathcal{A}_m}$
	$\partial_m\overline{\sigma^{mn}}\partial_n\phi_0^h = -\frac{\tilde{w}_0}{H} - \partial_m\overline{\sigma^{mj}\partial_j\phi_0^v}$	$\overline{\sigma^{mn}}\partial_n\phi_0^h\big\|_{\vec{x}\in\partial\mathcal{S}} = \overline{u^{\star m}}\big\|_{\vec{x}\in\partial\mathcal{S}}$
$\mathcal{O}(\varepsilon)$	$\partial_3\sigma^{33}\partial_3\phi_1^v = \rho - \partial_i\sigma^{ij}\partial_j\phi_0$	$\sigma^{33}\partial_3\phi_1^v\big\|_{z_\pm} = \begin{cases} \tilde{w}_0 - \sigma^{3n}\partial_n\phi_0\big\|_{z_+}, & z = z_+ \\ -\sigma^{3n}\partial_n\phi_0\big\|_{z_-}, & z = z_- \end{cases}$ $\sigma^{mj}\partial_j\phi_1^v\big\|_{\vec{x}\in\mathcal{A}_m} = \left[\overline{u^{\star m}} - \sigma^{mn}\partial_n\phi_0^h\right]_{\vec{x}\in\mathcal{A}_m}$
	$\partial_m\overline{\sigma^{mn}}\partial_n\phi_1^h = -\partial_m\overline{\sigma^{mj}\partial_j\phi_1^v}$	$\overline{\sigma^{mn}}\partial_n\phi_1^h\big\|_{\vec{x}\in\partial\mathcal{S}} = 0$
$\mathcal{O}(\varepsilon^p)$	$\partial_3\sigma^{33}\partial_3\phi_p^v = \rho - \partial_i\sigma^{ij}\partial_j\left(\sum_{r=0}^{p-1}\phi_r\right)$	$\sigma^{33}\partial_3\phi_p^v\big\|_{z_\pm} = -\sigma^{3n}\partial_n\phi_{p-1}\big\|_{z_\pm}$ $\sigma^{mj}\partial_j\phi_p^v\big\|_{\vec{x}\in\mathcal{A}_m} = -\sigma^{mn}\partial_n\phi_{p-1}^h\Big\|_{\vec{x}\in\mathcal{A}_m}$
	$\partial_m\overline{\sigma^{mn}}\partial_n\phi_p^h = -\partial_m\overline{\sigma^{mj}\partial_j\phi_p^v}$	$\overline{\sigma^{mn}}\partial_n\phi_p^h\big\|_{\vec{x}\in\partial\mathcal{S}} = 0$

Table 3.1: The general form of the leptic expansion. The indices i, j extend over all directions and the indices m, n do not include the vertical (thin) direction. The excess function is defined as $\tilde{w}_0 = w^\star|_{z_-}^{z_+} - \int_{z_-}^{z_+}\rho dz$.

but since $\sigma^{m3} = 0$ by assumption, the $\partial_m\sigma^{m3}\overline{\partial_3\phi^v}$ term is zero. Also, since each ϕ_p^v is the solution of an ordinary differential equation with Neumann BCs, we can choose solutions whose vertical averages are zero – eliminating the $\partial_m\sigma^{mn}\partial_n\overline{\phi^v}$ term as well. This means that the r.h.s. of the horizontal equations become zero for all but the $\mathcal{O}(1)$ stages. Since the boundary conditions are also zero for these problems, the solutions, $\phi_{p\geq 1}^h$, must be identically zero. Whenever σ^{ij} has this form, ϕ_0^h is the only horizontal function that needs to be found – eliminating most of the computation time.

In practice, σ^{ij} is often very close to the form shown above. In fact, many useful coordinate systems such as the Cartesian, cylindrical, and spherical systems are described by $\sigma^{ij} = Jg^{ij}$ which are *exactly* of this form. It is helpful to consider this while iterating. If we can find a value of P such that all $\phi_{p\geq P}^h$ will not significantly influence the overall convergence, or if we calculate that the norm of the horizontal equation's r.h.s. is below some threshold, then we can tell the leptic solver to stop performing horizontal solves in the interest of computation time. Alternatively, suppose we are about to find ϕ_p^h. We could simply set ϕ_p^h to zero everywhere and then set up the next vertical stage. If the vertical problem is consistent up to some prescribed tolerance, then we never needed the true solution of the horizontal equation to begin with! Otherwise, we can go back and solve the horizontal equation before moving on to the next vertical stage. This is a very economical way of deciding which horizontal solves are necessary.

$\mathcal{O}(1)$	$\partial_z^2 \phi_0^v = \rho$	$\partial_z \phi_0^v\big\|_{z_\pm} = \begin{cases} w^\star\big\|_{z_+} - \tilde{w}_0, & z = z_+ \\ w^\star\big\|_{z_-}, & z = z_- \end{cases}$ $\partial_m \phi_0^v\big\|_{\vec{x}\in\mathcal{A}_m} = \left[u^{\star m} - \overline{u^{\star m}}\right]_{\vec{x}\in\mathcal{A}_m}$
	$\nabla_h^2 \phi_0^h = -\frac{\tilde{w}_0}{H}$	$\partial_m \phi_0^h\big\|_{\vec{x}\in\partial\mathcal{S}} = \overline{u^{\star m}}\big\|_{\vec{x}\in\partial\mathcal{S}}$
$\mathcal{O}(\varepsilon)$	$\partial_z^2 \phi_1^v = \rho - \nabla^2 \phi_0$	$\partial_z \phi_1^v\big\|_{z_\pm} = \begin{cases} \tilde{w}_0, & z = z_+ \\ 0, & z = z_- \end{cases}$ $\partial_m \phi_1^v\big\|_{\vec{x}\in\mathcal{A}_m} = 0$
	No horizontal equation. $\phi_1^h = 0$	
$\mathcal{O}(\varepsilon^p)$	$\partial_z^2 \phi_p^v = \rho - \nabla^2 \left(\sum_{r=0}^{p-1} \phi_r\right)$	$\partial_z \phi_p^v\big\|_{z_\pm} = 0$ $\partial_m \phi_p^v\big\|_{\vec{x}\in\mathcal{A}_m} = 0$
	No horizontal equation. $\phi_p^h = 0$	

Table 3.2: The leptic expansion in Cartesian coordinates. The indices i, j extend over all directions and the indices m, n do not include the vertical (thin) direction. The excess function is defined as $\tilde{w}_0 = w^\star|_{z_-}^{z_+} - \int_{z_-}^{z_+} \rho dz$ and the horizontal Laplacian is $\nabla_h^2 = \partial_x^2 + \partial_y^2$.

3.2.4: An interpretation of ε

So far, we have derived a set of equations that produce a formal solution to the original Poisson problem based on the assumption that the parameter ε in eqs. (3.3) properly identifies terms of fundamentally different sizes. The procedure does not refer to a specific discretization of the equations, but heuristically depends on the existence of a small parameter. The latter is typically derived from an anisotropy inherent in the problem. This could be due to many causes, but to appeal to the interests of the author's own research, we will focus on an anisotropy in the geometry of the domain and numerical discretization. However, it is easy to generalize the foregoing argument regardless of the actual source of anisotropy. Naively, the aspect ratio of the domain could be used to define such a parameter. However, the following example shows that it must also depend on the details of the discretization.

Suppose we want to solve the isotropic, 2D Poisson equation in a rectangular domain. We will choose a uniform discretization with $N_x \times N_z$ cell-centers and we will define the aspect ratio to be $\alpha = H/L$. Upon switching to the dimensionless variables $\tilde{x} = x/L$ and $\tilde{z} = z/H$, Poisson's equation transforms as follows

$$\rho = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi = \frac{1}{H^2}\left(\alpha^2 L^2 \frac{\partial^2}{\partial x^2} + H^2 \frac{\partial^2}{\partial z^2}\right)\phi = \frac{1}{H^2}\left(\alpha^2 \frac{\partial^2}{\partial \tilde{x}^2} + \frac{\partial^2}{\partial \tilde{z}^2}\right)\phi$$

We will rescale the field variable, ϕ, so that it is dimensionless and of $\mathcal{O}(1)$, which is always possible. Now,

apart from an overall scaling of H^{-2}, it is natural to identify the small parameter ε with the aspect ratio of the grid. However, a little reflection shows that a more "quantitative" definition of ε cannot ignore the discretization altogether. Indeed, once discretized, the term involving x-derivatives is at most $\sim \alpha^2 N_x^2$ while the term involving z-derivatives is at least ~ 1. This means that if $\alpha N_x \ll 1$, then $\frac{\partial^2 \phi}{\partial x^2}$ will be fundamentally smaller than $\frac{\partial^2 \phi}{\partial z^2}$, and so the "smallness" of ε depends on both the aspect ratio of the domain and on the details of its discretization. It follows that, in the continuum limit, ε cannot be *a priori* ever guaranteed to be small.

Scotti and Mitran quantified these results in a more general manner. In their paper [67], they define the grid's *leptic ratio*, $\lambda = \min(\Delta x^m / H)$, where $\min \Delta x^m$ is the minimum grid spacing in the directions other than the vertical. We will show in section 3.3.1 that λ is related to our expansion parameter by $\varepsilon = \lambda^{-2}$. Note that the leptic ratio is controlled by the overall aspect ratio of the domain and by the degree of "coarseness" of the discretization in the horizontal directions. It is shown that if $\lambda > \mathcal{O}(1)$, then the computational grid is "thin," and the summarized equations of the previous section will indeed produce solutions whose sum converges to the solution of (3.1). This result extends to more exotic, N-dimensional geometries if we identify the symmetric tensor field, σ^{ij}, with Jg^{ij}.

At first, it would seem that this method is of very limited practical utility, being convergent only on rather coarse grids. Moreover, it it is at odds with the very sensible idea that finer grids should lead to better results. However, in this article we pursue the idea that even when the leptic ratio of the grid is below critical, so that the expansion will not converge, the method can still be used to accelerate the convergence of conventional methods. Loosely speaking, the idea is that the source term can be partitioned between a component that can be represented on a grid with $\lambda > 1$ and a remainder which needs a finer grid with $\lambda < 1$. Restricted to the former component, the expansion converges, while it diverges on the latter. On the contrary, traditional method such as BiCGStab tend to converge fast on the latter, and slow on the former. By judiciously blending both methods, we can achieve a uniformly high level of convergence. Also, note that since the expansion is formulated analytically, it can be implemented regardless of any particular choice of discretization of the domain. In the following examples, we will use a second-order scheme on a staggered grid, but the method is by no means restricted to this type of discretization.

Section 3.3: Convergence estimates

3.3.1: Restricted case

The heuristic arguments used earlier can be given an analytic justification in the case of a simple rectangular geometry. Once again, we limit to two dimensions, the extension to higher dimensional spaces being

trivial. The elliptic equation we wish to solve is

$$\left(\partial_x^2 + \partial_z^2\right)\phi\left(x,z\right) = \rho\left(x,z\right) \tag{3.16}$$

with homogeneous Neumann boundary conditions. Without loss of generality, we can set $\rho\left(x,z\right)$ equal to an arbitrary eigenfunction of the operator, $\cos\left(kx\right)\cos\left(mz\right)$. The exact, analytic solution of eq. (3.16) becomes simply

$$\phi\left(x,z\right) = -\frac{\cos\left(kx\right)\cos\left(mz\right)}{k^2+m^2}.$$

Now, let's investigate how the leptic solver would have arrived at a solution.

First, we will write eq. (3.16) as

$$\left(\varepsilon\partial_x^2 + \partial_z^2\right)\left(\phi_0 + \varepsilon\phi_1 + \ldots\right) = \cos\left(kx\right)\cos\left(mz\right), \tag{3.17}$$

where the ε is only used to identify small terms and will eventually be set to 1. Equating various powers of ε gives us

$$\begin{aligned}
\partial_z^2\phi_0 &= \cos\left(kx\right)\cos\left(mz\right) \\
\partial_z^2\phi_1 &= -\partial_x^2\phi_0 \\
\partial_z^2\phi_2 &= -\partial_x^2\phi_1 \\
&\text{etc}\ldots
\end{aligned}$$

The solution to the $\mathcal{O}\left(1\right)$ equation is

$$\phi_0\left(x,z\right) = \frac{\cos\left(kx\right)\cos\left(mz\right)}{-m^2}.$$

The constant of integration, that is $\phi_0^h\left(x\right)$, is identically zero since the BCs are homogeneous and there is no need for an excess function. The $\mathcal{O}\left(\varepsilon\right)$ equation becomes

$$\partial_z^2\phi_1 = -\frac{k^2}{m^2}\cos\left(kx\right)\cos\left(mz\right)$$

whose solution is

$$\phi_1\left(x,z\right) = -\frac{k^2}{m^2}\frac{\cos\left(kx\right)\cos\left(mz\right)}{-m^2}.$$

Continuing in this manner, we see that the solution at $\mathcal{O}(\varepsilon^n)$ is

$$\phi_n(x,z) = \left(-\frac{k^2}{m^2}\right)^n \frac{\cos(kx)\cos(mz)}{-m^2}$$

which means that if we terminate the leptic solver at $\mathcal{O}(\varepsilon^p)$, the solution we arrive at is given by the sum

$$\phi(x,z) = \frac{\cos(kx)\cos(mz)}{-m^2} \sum_{n=0}^{p} \left(-\frac{k^2}{m^2}\right)^n \tag{3.18}$$

where we can now identify ε with k^2/m^2. If $k^2/m^2 \geqslant 1$, then this geometric series will diverge as $p \to \infty$ and the leptic solver will ultimately fail. On the other hand, if $k^2/m^2 < 1$, then the series will be finite for all values of p and we can put the solution in a closed form,

$$\phi(x,z) = \frac{\cos(kx)\cos(mz)}{-(k^2+m^2)} \left\{1 - \left(-\frac{k^2}{m^2}\right)^{p+1}\right\}.$$

In the limit $p \to \infty$, the term in braces tends to 1 and we recover the exact, analytic solution of the elliptic equation.

Since the wavenumbers k and m are both positive, we can simply say that convergence of the leptic method requires $\max(k/m) < 1$. Analytically, this quantity depends on the harmonic content of the source, $\rho(x,z)$, and is, in principle, unbounded. Numerically, once a discretization has been chosen, k and m are limited to a finite number of values. If we let the source term be a general linear combination of eigenfunctions and if our rectangular domain has dimensions L by H divided uniformly into elements of size Δx by Δz, and it is discretized with a spectral method, then $\max(k) = \pi/\Delta x$ and $\min(m) = \pi/H$. This reproduces our convergence condition, $H/\Delta x < 1$, that is, $\lambda = \min(\Delta x^m/H) > 1$.

Notice that this convergence condition is a restriction on how we must discretize a given domain, it is not directly a restriction on the source term of the elliptic equation at hand. This means that the leptic method should converge similarly for all equations that use a particular uniform, rectangular grid. If the grid is not rectangular or uniform, then the relevant convergence condition is $H_i/\Delta x_i < 1$ at all grid positions, i. Here, H_i is the vertical height of the domain at x_i and Δx_i is the minimum horizontal grid spacing at x_i.

3.3.2: General case

Now, we will extend this argument to the more general case involving the positive-definite, symmetric tensor field, $\sigma^{ij}(x,z)$. We wish to perform a convergence analysis on

$$\left\{\partial_z\sigma^{zz}\partial_z+\varepsilon\left(\partial_x\sigma^{xx}\partial_x+\partial_x\sigma^{xz}\partial_z+\partial_z\sigma^{zx}\partial_x\right)\right\}\phi(x,z)=\rho(x,z). \tag{3.19}$$

Without the exact form of each $\sigma^{ij}(x,z)$, we cannot trivially diagonalize the operator in (k,m)-space. We can, however, diagonalize the operator in (x,z)-space by performing a small rotation by an angle $\frac{1}{2}\tan^{-1}\left(\frac{2\varepsilon\sigma^{xz}}{\sigma^{zz}-\varepsilon\sigma^{xx}}\right)$. This casts the equation into the simpler form

$$\left\{\partial_z\left[\sigma^{zz}+\mathcal{O}\left(\varepsilon^2\right)\right]\partial_z+\partial_x\left[\varepsilon\sigma^{xx}+\mathcal{O}\left(\varepsilon^2\right)\right]\partial_x\right\}\phi(x,z)=\rho(x,z),$$

where the x and z now represent the new coordinates. We might as well just let $\sigma^{zz}+\mathcal{O}\left(\varepsilon^2\right)\rightarrow\sigma^{zz}$ and $\varepsilon\sigma^{xx}+\mathcal{O}\left(\varepsilon^2\right)\rightarrow\varepsilon\sigma^{xx}$ so that

$$\left\{\partial_z\sigma^{zz}\partial_z+\varepsilon\partial_x\sigma^{xx}\partial_x\right\}\phi(x,z)=\rho(x,z). \tag{3.20}$$

Even though each term of eq. (3.20) is functionally different than the corresponding terms of eq. (3.17), their magnitudes are equal. This means that a convergence analysis of eq. (3.20) must lead to a restriction of the form $\lambda>\mathcal{O}(1)$. Rotating back to eq. (3.19) cannot possibly change this restriction due to the vanishing size of the rotation angle. This shows that even in the general case of eq. (3.1), the leptic solver will converge as long as the discretization is chosen to satisfy $\lambda>\mathcal{O}(1)$.

3.3.3: The leptic solver as a preconditioner

Let us return to the simple case of solving $\nabla^2\phi=\cos(kx)\cos(mz)$ on a rectangular domain. After applying n_l iterations of the leptic solver, the residual, r, is found via eq. (3.18) with $p=n_l-1$.

$$\begin{aligned}
r &= \cos(kx)\cos(mz)-\nabla^2\left\{\frac{\cos(kx)\cos(mz)}{-m^2}\sum_{n=0}^{n_l-1}\left(-\frac{k^2}{m^2}\right)^n\right\}\\
&= \cos(kx)\cos(mz)-\frac{k^2+m^2}{m^2}\cos(kx)\cos(mz)\sum_{n=0}^{n_l-1}\left(-\frac{k^2}{m^2}\right)^n\\
&= \left\{1+\sum_{n=1}^{n_l}\left(-\frac{k^2}{m^2}\right)^n-\sum_{n=0}^{n_l-1}\left(-\frac{k^2}{m^2}\right)^n\right\}\cos(kx)\cos(mz)\\
&= \left(-\frac{k^2}{m^2}\right)^{n_l}\cos(kx)\cos(mz)
\end{aligned}$$

The last line comes from collapsing the telescoping set of sums. This gives us an amplification factor for each eigenmode of the residual. That is, if we are given a generic residual and perform an eigenvector expansion,

$$r(x,z) = \sum_k \sum_m r(k,m) \cos(kx) \cos(mz),$$

then the magnitudes of the individual components, $r(k,m)$, will be amplified or attenuated by k^2/m^2 each time we iterate. If the grid is constructed such that $\lambda > \mathcal{O}(1)$, then $r(k,m)$ will always be attenuated since $\max\left(k^2/m^2\right) < 1$. If, however, the grid's leptic ratio is $\sim \mathcal{O}(1)$, then only those eigenmodes with $k^2/m^2 < 1$ will diminish and those with $k^2/m^2 > 1$ will be amplified. In (x,z)-space, this effect appears as a diverging solution, but in (k,m)-space, we can see that the solution is split into converging and diverging parts – we are conditioning the solution. For this reason, even though preconditioning is normally understood as the action of substituting the original operator with a modified one with better spectral properties [74], we will use preconditioning to mean the action of replacing an initial guess with one that has better spectral support.

As an example, consider a $64 \times 64 \times 16$ grid with $\Delta x = (1, 1, 0.1)$. This fixes $\varepsilon = H^2/\Delta x^2$ at $1.6^2/1^2 = 2.56$, which is large enough to cause problems for the leptic solver.[2] In order to learn how the solvers are treating the modes on this grid, let's apply them to

$$\nabla^2 \phi = \sum_{i=1}^{32} \sum_{j=1}^{32} \sum_{k=1}^{8} \cos\left(\frac{2\pi i x}{L}\right) \cos\left(\frac{2\pi j y}{L}\right) \cos\left(\frac{2\pi k z}{H}\right)$$

with homogeneous boundary conditions. This is a residual equation whose r.h.s. harbors every periodic mode supported by the grid in equal amounts (except for the zero modes, which must be removed to be consistent with the boundary conditions). We only consider periodic modes to facilitate spectral analysis via FFT. In one test, we solved this equation with a BiCGStab solver and in another separate test, we used the leptic method. BiCGStab stalled in 26 iterations and the leptic solver began to diverge after 24 iterations. Once progress came to a halt, we performed an FFT to locate which modes were converging slowest. To simplify the visualization, we found the (k_x, k_y) slice that contained the most slowly converging modes (which, consistent with the previous analysis, is the smallest value $k_z = 2\pi/H$). The results are in figures 3.2 and 3.3, which were generated by VisIt [75].

These plots represent the base-10 logarithm of the Fourier coefficients of each mode. It is apparent that each method has its own distinct problem region. The BiCGStab solver has the most trouble dealing with low frequency modes while the leptic solver has trouble with high frequency modes. Since the leptic solver produced a residual whose largest modes can easily be handled by BiCGStab, we apply the BiCGStab using

[2] For the remainder of this document, we will be dealing with a geometric anisotropy quantified by λ. The perturbation parameter is then $\varepsilon = (H/\Delta x)^2$.

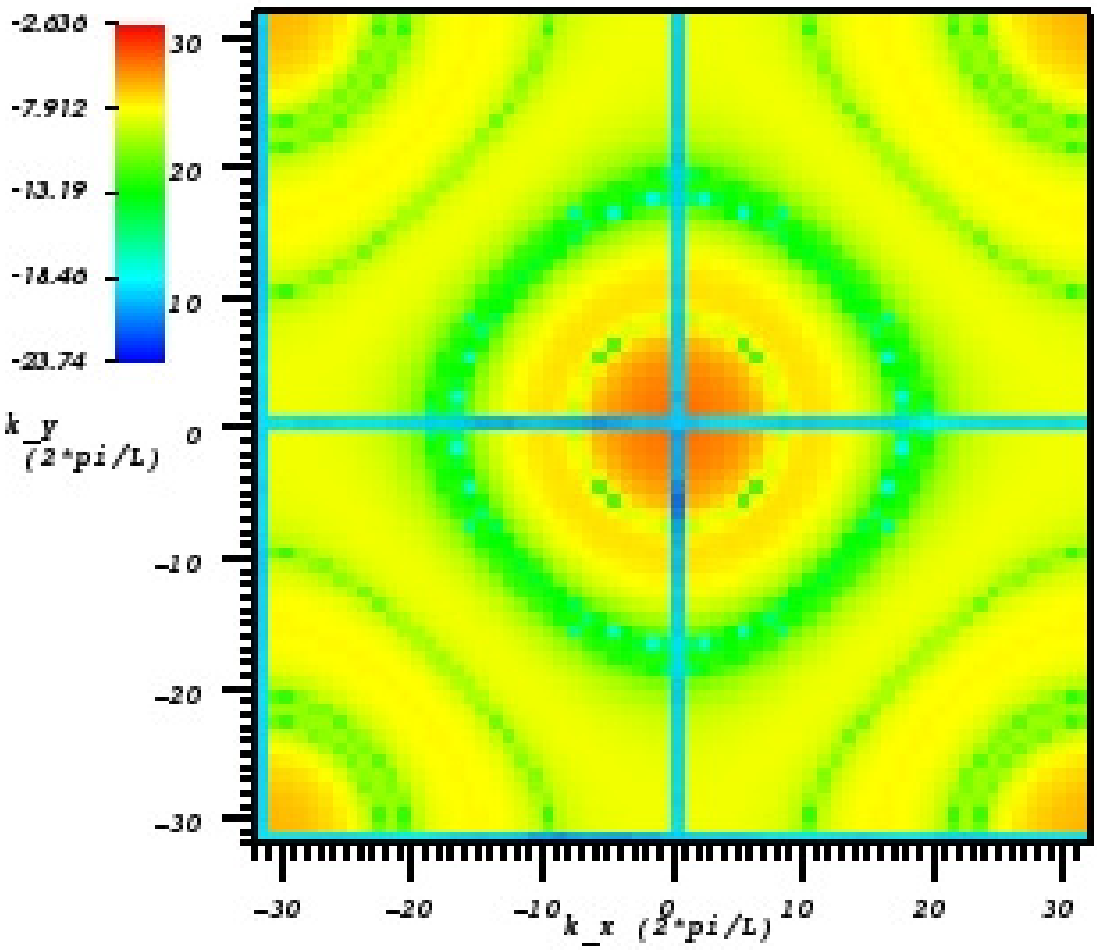


Figure 3.2: After 26 iterations of BiCGStab, this is a 2-dimensional slice of the residual error's FFT on a logarithmic scale. Notice the large error near the center of the plot, indicating BiCGStab's difficulty eliminating low frequency errors. The vertical and horizontal lines indicating a very low residual (the "crosshairs") are the zero-frequency modes that must be fixed to agree with the boundary conditions.

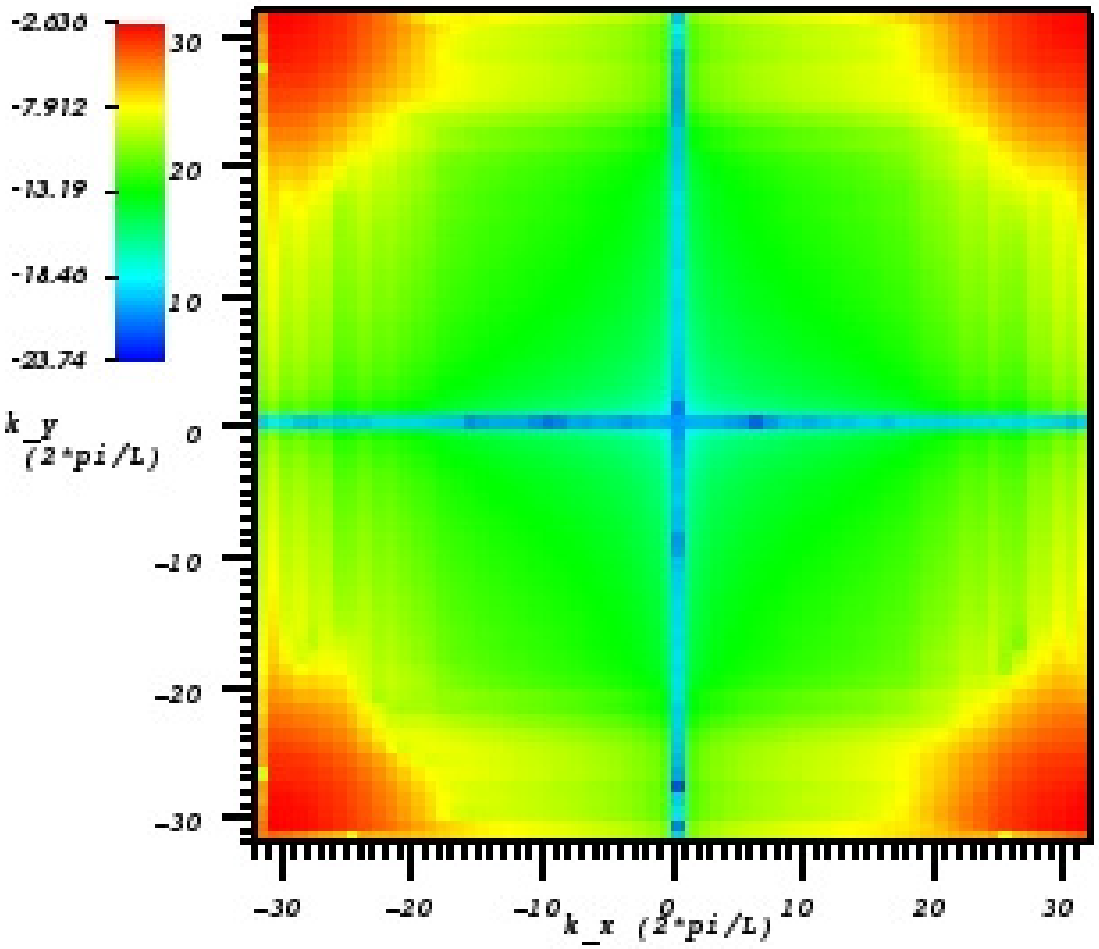


Figure 3.3: The residual error after 24 iterations of the leptic solver. This solver eliminates low frequency errors much more effectively than high frequency errors, indicating the leptic solver's potential to serve as a preconditioner for BiCGStab.

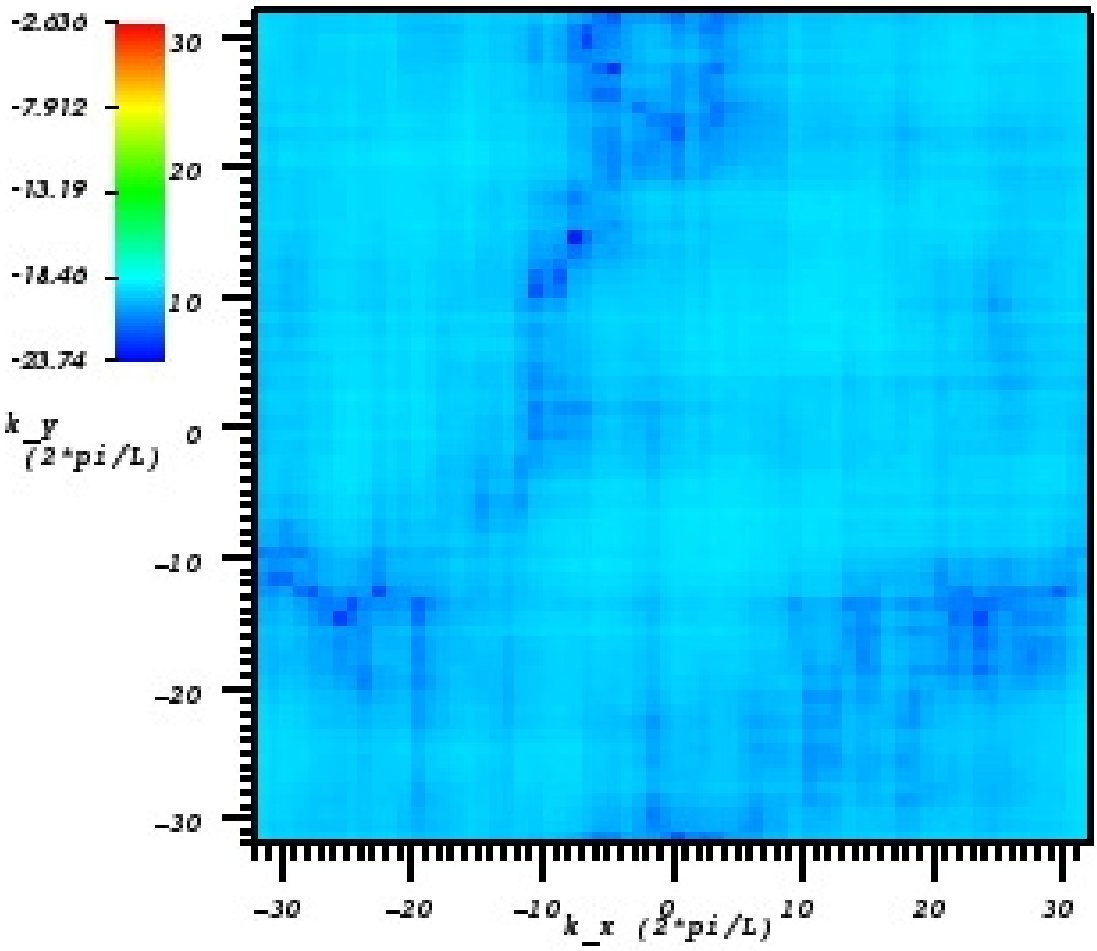


Figure 3.4: The residual error of the BiCGStab method when given an initial guess generated by the leptic solver.

as initial guess the output of the leptic solver after 18 iterations. Now BiCGStab is able to converge quickly (figure 3.4).

We should mention that this is just an illustration. In this example, the BiCGStab method on such a small grid would have converged on its own in a reasonable number of iterations. On a larger grid, BiCGStab alone often converges too slowly to be a viable solution method and sometimes stalls due to the condition number of the operator. Further complications arise when we use mapped coordinates because this tends to drive the condition number of the operator even higher. In these situations, using the leptic method to generate a suitable initial guess becomes quite useful. We will illustrate this effect in further detail in section 3.4.

Section 3.4: Demonstrations

In this section, we will create a sample problem on various numerical domains in order to compare the effectiveness of traditional solvers with methods that utilize the leptic solver. Our traditional solver of choice will be the BiCGStab method preconditioned with the incomplete Cholesky factorization (IC) of the elliptic operator [70]. For simplicity, we will use a rectangular domain and both the r.h.s. and boundary conditions will be generated by the vector field

$$\begin{aligned} u^{\star 1} &= \left(\frac{z}{L_z}\right)^2 \sin\left(\frac{\pi y}{L_y}\right) + \frac{x}{\sqrt{2}L_z} \\ u^{\star 2} &= \left(\frac{z}{L_z}\right)^2 \sin\left(\frac{\pi x}{L_x}\right) \\ u^{\star 3} &= -\left(\frac{z}{L_z}\right)^2 \cos\left(\frac{\pi z}{4L_z}\right) \\ \rho &= \partial_i u^{\star i}. \end{aligned}$$

This vector field produces a nontrivial source term that is harmonically rich – $\max(k/m)$ can be made arbitrarily large by increasing the resolution. To compare the solvers, we will plot the relative residual as a function of the iteration number. For what follows, the vertical and horizontal solves of the leptic solver will each be counted each as an iteration.

3.4.1: High leptic ratio - Cartesian coordinates

First, we set $N = (64, 64, 16)$ and $\Delta x = (0.1, 0.1, 0.001)$. This fixes ε at 0.0256, which lies well within the region where the leptic solver outperforms traditional methods. Figure 3.5 shows the results. The leptic solver is clearly the more efficient method. In only 6 iterations, it is able to achieve a relative residual of 10^{-10}. We would have needed 170 iterations of the BiCGStab/IC solver to obtain a residual error of that magnitude.

3.4.2: Borderline cases - Cartesian coordinates

When $\varepsilon = \mathcal{O}(1)$, the leptic solver may or may not be the most efficient solver. We will denote these situations as *borderline cases*. In the first borderline case, we will bring ε to 1 by setting $N = (64, 64, 10)$ and $\Delta x = (0.1, 0.1, 0.01)$. Figure 3.6 shows that the leptic solver requires approximately 5 times as many iterations as it did in the $\varepsilon = 0.0256$ example to achieve an $\mathcal{O}\left(10^{-10}\right)$ residual error. The BiCGStab/IC method, however, converges a bit more quickly than it did in the previous example. It required only 90 iterations to catch up to the leptic method. This is empirical evidence of our theoretical assertion – by raising

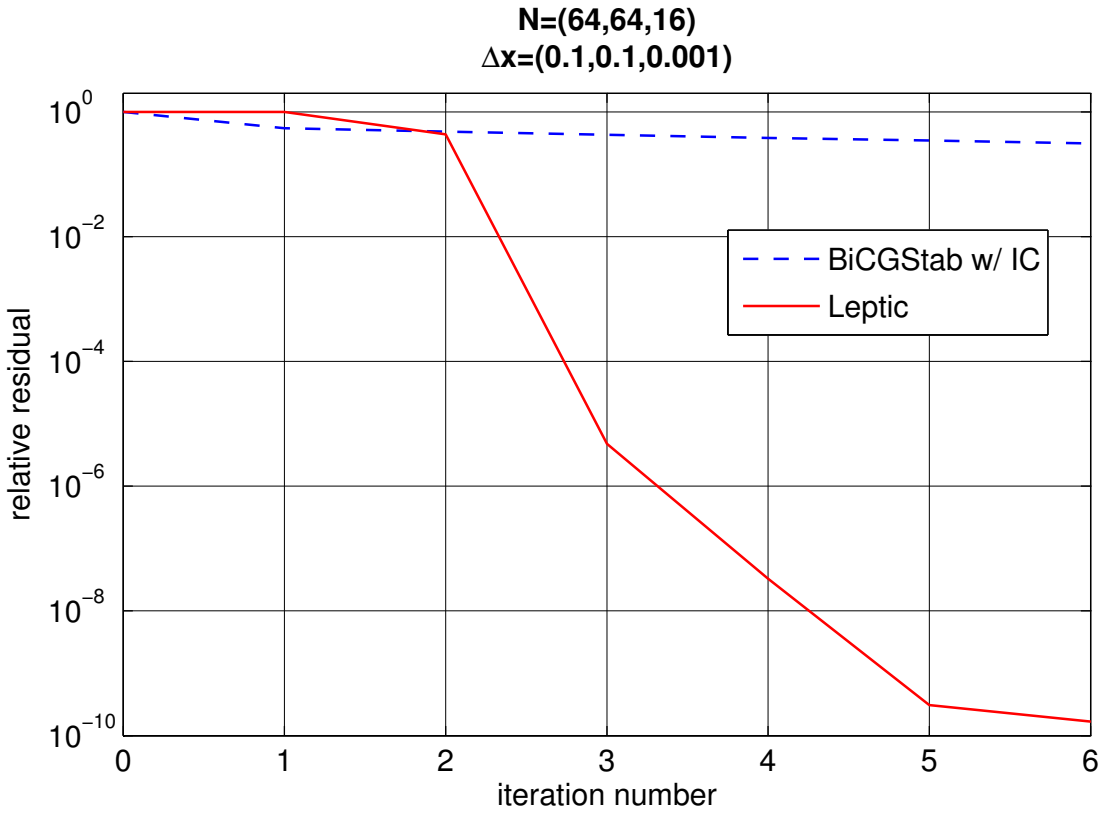


Figure 3.5: With $\varepsilon \approx 1/40$, the leptic solver is clearly more efficient than BiCGStab. Since we are using Cartesian coordinates, the leptic solver only needed to perform one horizontal solve.

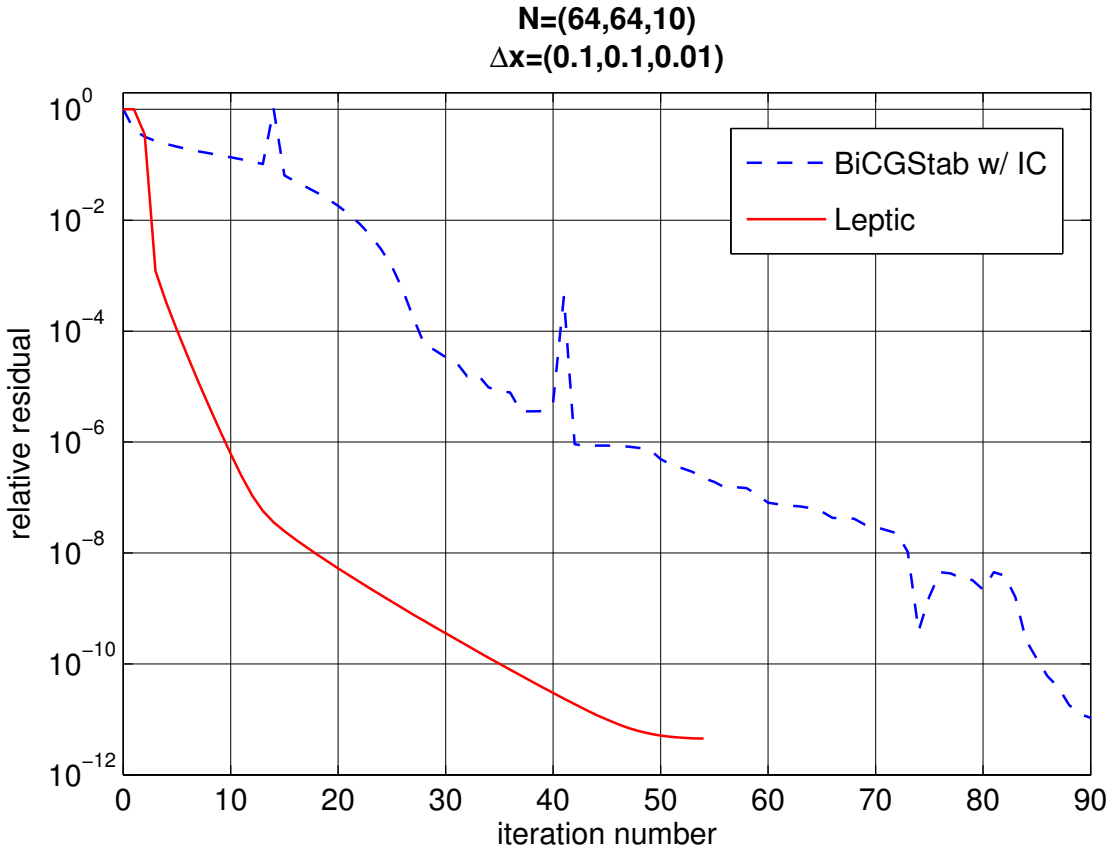


Figure 3.6: The convergence patterns of the leptic and BiCGStab solvers when $\varepsilon = 1$ and condition number $\approx 10^{5.6}$. The spikes in the BiCGStab residual are due to restarts.

ε (decreasing the lepticity), the leptic solver becomes less effective and the traditional method becomes more effective. In this specific borderline case, the leptic solver outperforms the BiCGStab method.

By varying N_x and N_y, we can generate an entire class of grids with the same ε. For example, if we bring N up to $(256, 256, 10)$, the BiCGStab method should not converge as rapidly as before. On the other hand, ε is still 1, which means the leptic solver should perform almost as well as it did on the $64 \times 64 \times 10$ grid. This is because most of the leptic solver's convergence relies on the vertical solver. This is true in general when the horizontal solver is able to be switched off (see section 3.2.3) – as the horizontal domain grows, the leptic solver outperforms traditional relaxation methods. This effect is shown in figure 3.7. By comparing figures 3.6 and 3.7, we see that unlike the leptic method, the BiCGStab/IC method is in fact slowed down by the larger horizontal grid.

The true value of the leptic solver is illustrated by when we use it to generate a suitable initial guess for the BiCGStab/IC solver (see section 3.3.3). This initial guess has an error that is dominated by high wavenumbers. The BiCGStab solver then rapidly removes those errors as shown by the dash-dot curve in

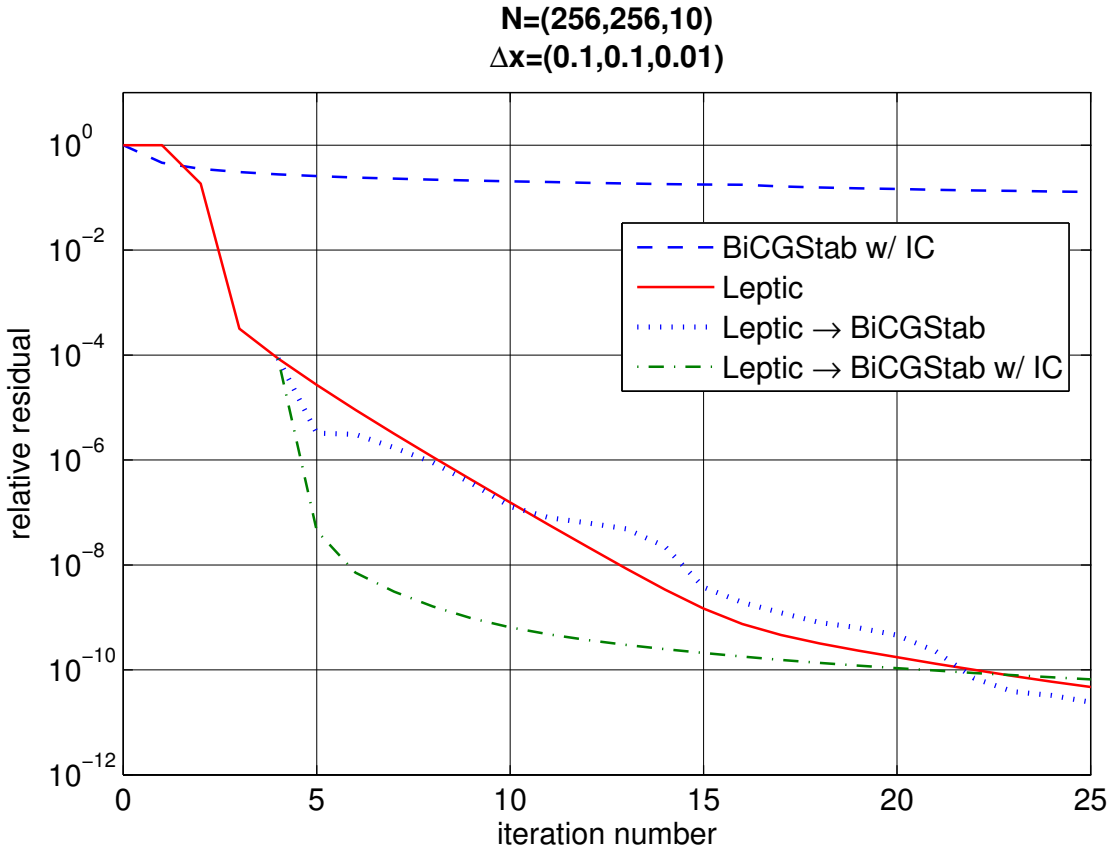


Figure 3.7: The performance of various solution methods with $\varepsilon = 1$ and condition number $\approx 10^{6.8}$. After a few iterations of the leptic solver, BiCGStab can achieve a fast convergence rate. Using a preconditioner such as an incomplete Cholesky decomposition can drive this rate even higher.

figure 3.7.

As our final isotropic, borderline case, we will set $N = (50, 50, 50)$ and $\Delta x = (0.1, 0.1, 0.004)$. This is appropriate for a cubic, vertically stratified domain and brings ε up to 4. The BiCGStab/IC solver does not provide immediate convergence and the leptic method would have started to diverge after its third iteration, but when we combine the two methods as we did in the last example, we see a much more rapid convergence than either method could individually achieve (figure 3.8).

3.4.3: High leptic ratio - Mapped coordinates

When the metric is diagonal, several of the terms in our expansion (sec. 3.2.2) vanish. This means we can remove much of the code to produce a more efficient algorithm. This reduced code is what generated the results of the previous sections. However, in these simple geometries the value of the leptic expansion is somewhat limited because it is normally possible to employ fast direct solvers. Not so in the case we consider now, where we apply the full algorithm by considering a non-diagonal metric. The metric we will

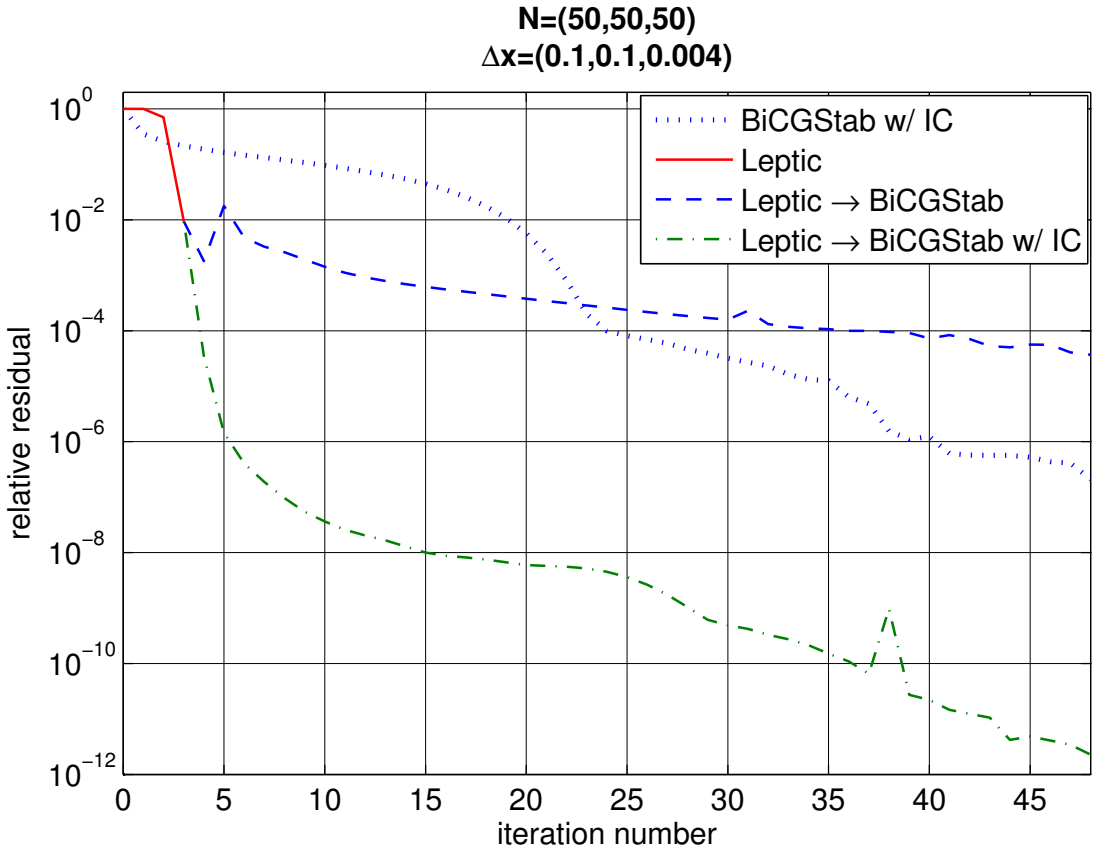


Figure 3.8: Performance of the solvers with $\varepsilon = 4$ and condition number $\approx 10^{6.2}$. The leptic solver began diverging after it's third iteration, so control was passed to the Krylov solver. Again, the leptic solver proves most valuable as a preconditioner for the BiCGStab/IC solver when $\varepsilon = \mathcal{O}(1)$.

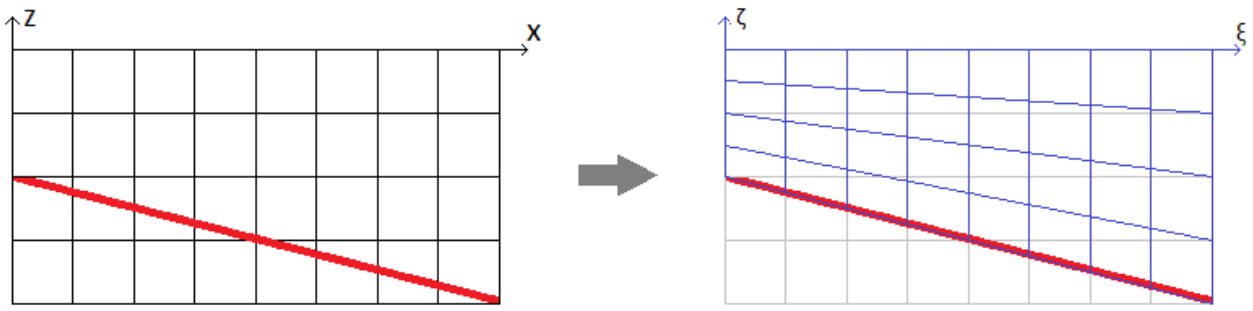


Figure 3.9: A cross section of the coordinate mapping. The thick line denotes the lower vertical boundary.

use is routinely employed in meteorological and oceanic simulations of flows over non uniform terrain. It maps the physical domain characterized by a variable topography $z = h(x,y)$ to a rectangular computational domain. Although any relief may be specified, it is sufficient for our illustrative purposes to simply let the depth go from $d/2$ to d linearly as x goes from 0 to L, where $d < 0$ (Figure 3.9).

We define $h(\xi) = \frac{d}{2} + \frac{d}{2L}\xi$, where $(x,y,z) \to \left(\xi, \eta, \frac{h(\xi)}{d}\zeta\right)$. This means that the symmetric tensor, $\sigma^{ij} = Jg^{ij}$, is given by

$$Jg^{ij} = \begin{pmatrix} \frac{\xi+L}{2L} & 0 & -\frac{\zeta}{2L} \\ 0 & \frac{\xi+L}{2L} & 0 \\ -\frac{\zeta}{2L} & 0 & \frac{4L^2+\zeta^2}{2L(\xi+L)} \end{pmatrix}^{ij}.$$

For the next two examples, we will be seeking the solution,

$$\phi_{\text{sol}}(\xi,\eta,\zeta) = \cos\left(\frac{2\pi\xi}{L}\right)\cos\left(\frac{2\pi\eta}{L}\right)\cos\left(\frac{2\pi\zeta}{d}\right)$$

by setting $u^{\star i} = Jg^{ij}\partial_j\phi_{\text{sol}}$ and $\rho = \partial_i u^{\star i}$. Despite its simplicity, this will again generate harmonically rich sources due to the metric tensor.

Setting $N = (256,256,64)$ and $\Delta x = (0.25,0.25,0.0025)$ gives us $\varepsilon = 0.4096$. After 58 iterations, the BiCGStab method reaches a relative residual of $\mathcal{O}\left(10^{-5}\right)$. As shown in figure 3.10, this is matched by the leptic method with only the vertical solve of the first iteration. It should be pointed out that the full 3D problem that is sent to the BiCGStab solver has a non-diagonal tensor, Jg^{ij}, which makes deriving the matrix elements for a preconditioner rather difficult. However, the leptic solver sends a diagonal, vertically averaged tensor, $\overline{Jg^{mn}}$, to its 2D horizontal solver. The matrix elements of the resulting 2D elliptic operator are easy to produce, making an incomplete factorization an economically viable option. For comparison purposes, however, we did not take advantage of this fact in order to keep the leptic solver's horizontal relaxation scheme as similar as possible to the 3D relaxation scheme.

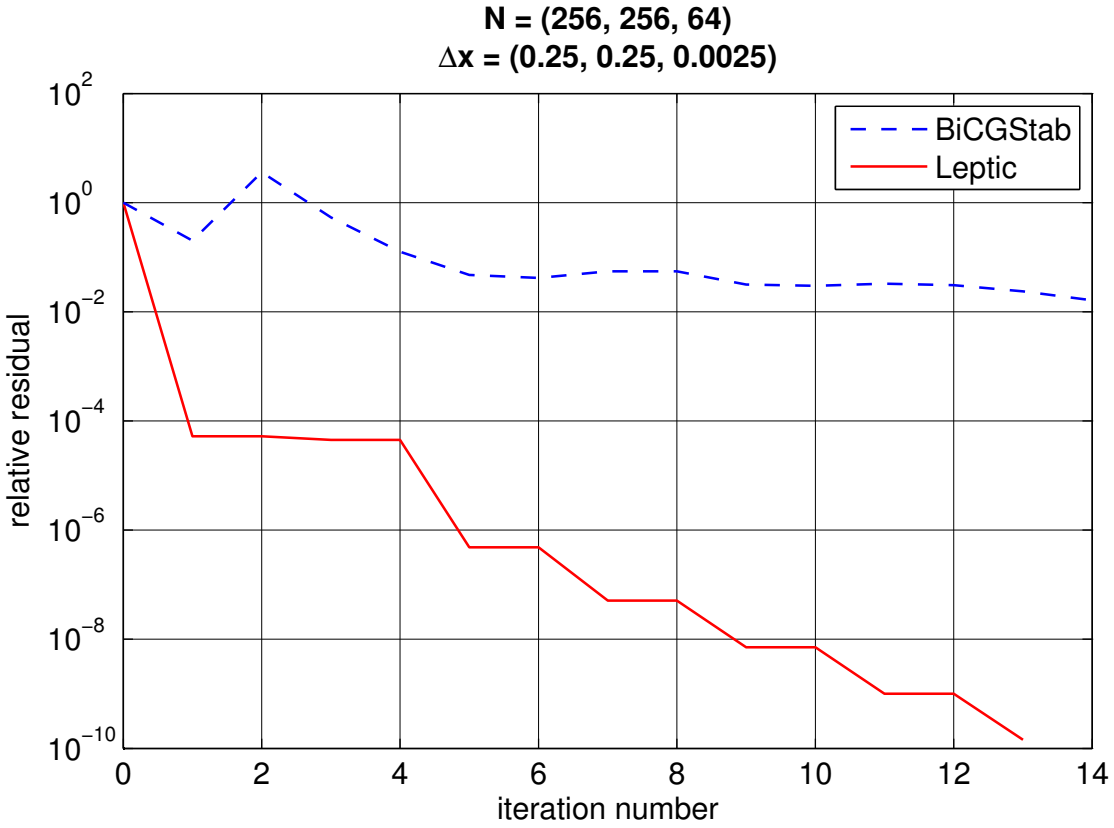


Figure 3.10: Performance of the leptic and Krylov solvers with an anisotropic σ^{ij} and $\varepsilon \approx 0.4$. Since our solver is using Chombo's matrix-free methods (known as *shell matrices* in some popular computing libraries such as PETSc [1]), the Cholesky decomposition of the elliptic operator can not be performed.

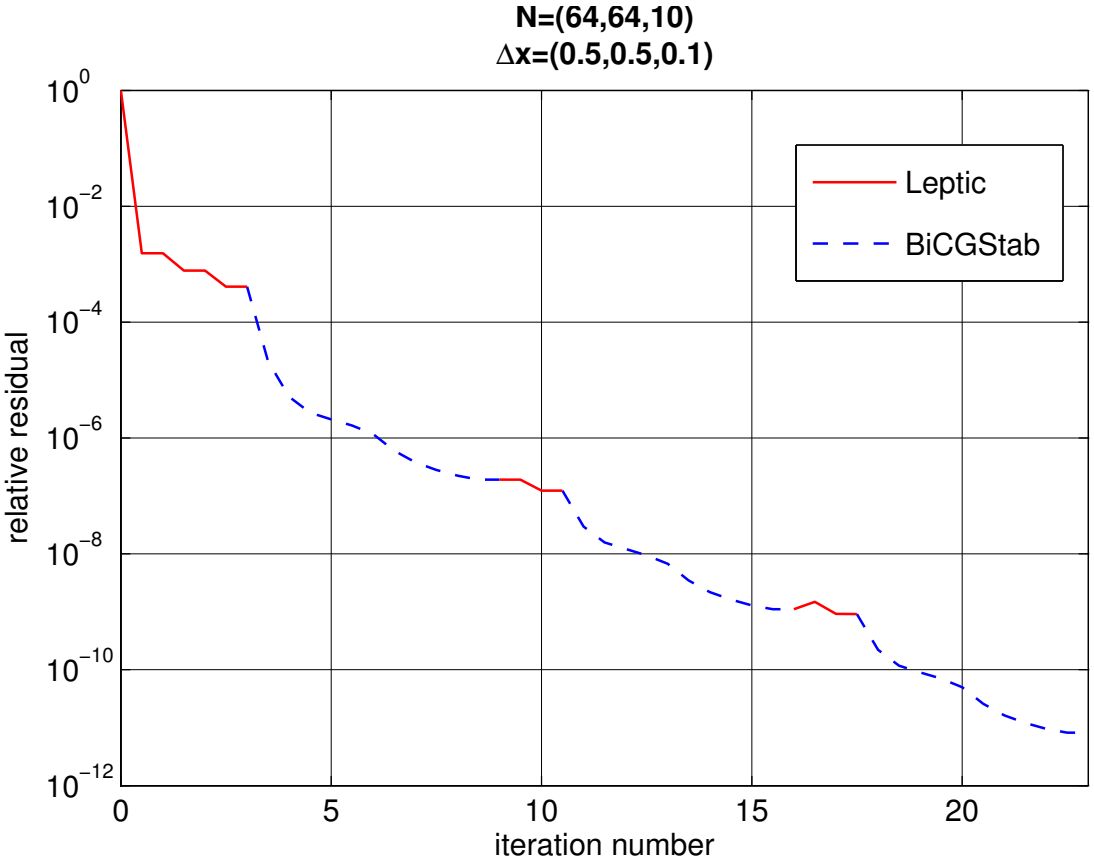


Figure 3.11: The convergence pattern of a hybrid leptic/Krylov method when $\varepsilon = 4$ on an anisotropic domain. In this case, neither method would have individually provided rapid convergence, but when the methods are combined, we see a very rapid convergence and the introduction of another preconditioner (eg. IC) is not necessary.

3.4.4: Borderline case - Mapped coordinates

For this scenario, we set $N = (64, 64, 10)$ and $\Delta x = (0.5, 0.5, 0.1)$ giving us $\varepsilon = 4$. To solve our sample problem on this grid, we let the leptic and BiCGStab solvers work together, iteratively. As soon as one solver begins to stall or diverge, control is passed to the other solver. The effectiveness of this algorithm is explained in section 3.3.3 – the leptic solver first reduces low frequency errors until high frequency errors begin to dominate the residual, then the BiCGStab solver reduces high frequency errors until low frequency errors dominate the error. This continues until the residual error has been reduced over the entire spectrum supported by the grid. Figure 3.11 illustrates the effectiveness of this algorithm rather convincingly.

Section 3.5: Discussion

The leptic method was originally devised as a method to add a physically appropriate amount of dispersion when numerically modeling the propagation of nonlinear waves in a dispersive medium. In this paper, we generalized the method so it could be used to actually speed up the numerical solution of Poisson problems characterized by a high level of anisotropy. The key idea is that instead of (or in addition to) preconditioning the operator to achieve overall better spectral properties, we precondition the initial guess (or the restarts) to achieve better spectral support of the residual by coupling the leptic expansion to Krylov methods. However, since the former converges on its own if the lepticity of the grid is sufficiently large, it is easy to see that it could be used within multigrid methods as well. Namely, when the coarsening reaches a point that the lepticity of the grid is below critical, the leptic expansion can be used in lieu of the relaxing stage to generate an exact solution at the coarse level. This would likely cut down the layers of coarsening.

In its full generality, this method can be used to solve anisotropic Poisson equations in arbitrary, D-dimensional coordinate systems. In many cases, however, numerical analysis is performed using simple, rectangular coordinates without an anisotropic tensor, σ^{ij}. This simplification reduces much of the computation. For these common purposes, we included a summary of the Cartesian version of the expansion. Both the general and Cartesian expansions are summarized in section 3.2.2.

When implementing the leptic method for numerical work, the computational domain should be split in all but the vertical (stiff) dimension. The vertical ordinary differential equations require a set of integrations – one for each point in the horizontal plane. With this domain decomposition, these integrations are independent of one another and the solutions to the vertical problems can be found via embarrassingly parallel methods. On the other hand, the solutions to the horizontal equations cannot be parallelized as trivially, making their solutions more costly to arrive at. In light of this, section 3.2.3 was provided to discuss when it is appropriate to eliminate these expensive stages.

The computational cost of a single iteration is of the same order as a preconditioned step of a Krylov method. The savings are obtained in the faster rate of convergence, shown in the examples above over a wide range of anisotropic conditions, as well as the relative ease with which the leptic expansion can be parallelized. The best rate of convergence is achieved by using the leptic expansion at the beginning and every time the convergence rate of the Krylov method slows down. We did not attempt to predict a priori after how many steps the switch is necessary. If the Poisson problem, as it often happens, is part of a larger problem that is solved many times, a mock-up problem should be solved at the beginning to determine empirically the best switch pattern.

Adapting the expansion to accept Dirichlet boundary conditions would require a new derivation similar

to that of section 3.2, but the job would be much simpler since Dirichlet elliptic operators have a trivial null space, thereby eliminating the need to consider compatibility conditions among the second-order operators and their boundary conditions.

CHAPTER 4: Time marching schemes[1]

Much of our algorithm derives from Dan Martin's Ph.D. thesis [49] and the Chombo AMR framework [68]. We have modified their methods to accommodate several needs.

1. We have maintained general covariance so that the domain can be described in curvilinear coordinates. This will be useful when mapping a logically rectangular coordinate system around irregular boundaries at the ocean floor.

2. We have added robust methods to account for incompressibility and gravitational effects. This includes a 3D Poisson solver specifically designed to work efficiently on high-aspect ratio domains and a stable integration of the stiff forcing terms that give rise to buoyancy oscillations.

3. We allow for any other well-behaved (non-stiff) forcing terms via the generic symbol $\vec{f}_{\text{ext}}$. For our purposes, this includes tidal and sponge forcing.

Section 4.1: The piecewise-parabolic method for low-Mach flows

In chapter 2, we discussed how SOMAR discretizes differential operators and generates metric components within the framework of finite differences. The stencils were constructed to make the evolution of eq. 2.1 freestream preserving in curvilinear coordinates. Despite this, no mention was made of how to compute the *advecting velocity*. This is because how we decide to compute the advecting velocity has far more impact on the accuracy of SOMAR than on freestream preservation.

A common choice for integrating nonlinear advection terms is to use finite volume methods (FVMs). In fact, since FVMs produce intermediate face and time averaged fluxes, they are easily and naturally tailored for use in both adaptive and static mesh refinement algorithms. Indeed, most of the work in AMR has involved finite volume integrations of the underlying hyperbolic conservation laws (eg., Athena [77], CASTRO [78], AMRClaw [79], Enzo [80]). For this reason, SOMAR computes advecting velocities and fluxes using a curvilinear, low-Mach adaptation of the Piecewise-Parabolic Method (PPM) [81].

[1]This chapter previously appeared as part of an article in the Journal of Computational Physics. The original citation is as follows: Edward Santilli and Alberto Scotti. The stratified ocean model with adaptive refinement (somar). *Journal of Computational Physics*, 291(0):60 – 81, 2015.

Suppose we want to compute one timestep of an advection problem written in conservative form,

$$\frac{\partial \Phi}{\partial t} = -\frac{1}{J}\partial_i F^i$$
$$F^i = J u^i_{\mathrm{AD}} \Phi,$$

where Φ represents any advected quantity, such as a scalar field or Cartesian velocity component, and u_{AD} is the advecting velocity. We integrate both sides from t^n to $t^{n+1} = t^n + \Delta t$ and approximate the face-averaged and time-averaged flux,

$$\Phi^{n+1} = \Phi^n - \Delta t \mathbf{D}\vec{F}^{n+1/2}$$
$$\vec{F}^{n+1/2} \approx J\, \vec{u}^{n+1/2}_{\mathrm{AD}}\, \Phi^{n+1/2}.$$

The algorithm that SOMAR uses to estimate $\vec{u}^{n+1/2}_{\mathrm{AD}}$ and $\Phi^{n+1/2}$ is detailed in Colella, *et. al.*[82], however, when fluxes are generated by a Riemann solver, the result is scaled by the Jacobian determinant, J.

For our purposes, we have no need for numerical protection against shocks. Therefore, we remove the artificial viscosity and slope flattening mechanisms. Also, in the interest of accuracy, we use fourth-order slopes and we never apply a limiter to upwinded velocity estimates. On the other hand, when we are upwinding the buoyancy, we find that a limiter is crucial to the stability of the advection scheme and use the extremum-preserving parabolic profile limiter defined in section 2.2 of Sekora and Colella [83].

Section 4.2: Single-level update with $\bar{b}(z) \equiv 0$

When there is no background stratification, we can perform the complete update in a way that does not require a separate treatment of the terms that give rise to buoyancy oscillations. This minimizes operator splitting errors and simplifies the CC projection step. In this section, we will split the external forces into two parts: forces that can be incorporated explicitly, $\vec{f}_{\mathrm{E}}$, and forces that must be incorporated implicitly or semi-implicitly, $\vec{f}_{\mathrm{I}}$. Coriolis effects are contained within $\vec{f}_{\mathrm{I}}$.

In short, the theme of this algorithm is to extrapolate the velocity to $t^{n+1/2}$, use this to advect/diffuse the buoyancy to t^{n+1}, then use a time-averaged buoyancy to update the velocity to t^{n+1}. As explained in section 4.1, the extrapolation is performed using a multidimensional upwinding scheme, which appropriately assembles the normal flux data generated by the 1D piecewise parabolic method for use in a 2D or 3D algorithm. From this point on, we will simply denote the application of these methods to a generic dynamical

variable Φ as

$$\Phi^{n+1} = \mathrm{PPM}\left(\Phi^n, u^i_{\mathrm{AD}}, S^n, \Delta t\right),$$

where u^i_{AD} is ideally the time and face averaged advecting velocity and S^n is the source term needed to trace characteristic lines back to t^n. This source term is the sum of the viscous/diffusive terms and all body forces acting on the fluid – including those forces that may require an implicit updating scheme.

4.2.1: Unsplit numerical algorithm

1. *Compute advecting velocities.* These are the velocities that are extrapolated to face centers (FC) at time $t^{n+1/2}$ then projected exactly, in the discrete sense (MAC projection).

$$\begin{aligned}
u^{n,i}_{\mathrm{edge}} &= \mathrm{Av}^{\mathrm{CC}\to\mathrm{FC}}\left(u^{n,i}\right) \\
S^{n,\alpha} &= \nu\nabla^2 u^{n,\alpha} - b^n\hat{z}^\alpha + f^{n,\alpha}_{\mathrm{E}} + f^{n,\alpha}_{\mathrm{I}} \\
u^{\star,\alpha}_{\mathrm{AD}} &= \mathrm{PPM}\left(u^{n,\alpha}, u^{n,i}_{\mathrm{edge}}, S^{n,\alpha}, \frac{\Delta t}{2}\right)
\end{aligned}$$

All upwinding is built in to the PPM operation. We obtain the best results with fourth order slopes and no limiting in these velocity updates. The tracing scheme uses free-slip physical BCs and conservative linear interpolation at the coarse/fine (CF) interface. The viscous source term uses no-slip physical BCs and quadratic interpolation at the CF interface to compute the Laplacian. Once $\nabla^2 u^{n,\alpha}$ is known in the level's interior, we fill its physical and CF interface ghosts using constant (zeroth-order) extrapolation, which is just a direct copy of the adjacent cell values in the domain's interior.

Once the FC velocities, $u^{\star,\alpha}_{\mathrm{AD}}$, have been predicted, they need to be MAC projected.

$$\begin{aligned}
\nabla^2\phi &= \frac{1}{J}\partial_i J u^{\star,i}_{\mathrm{AD}} \\
u^i_{\mathrm{AD}} &= u^{\star,i}_{\mathrm{AD}} - g^{ij}\partial_j\phi
\end{aligned}$$

The BCs for ϕ are homogeneous Neumann at physical boundaries and quadratic interpolation at CF interfaces with a scaled pressure from the coarser level, $0.5\,\Delta t\,\pi^{l-1}$, where Δt is the time step on *this* level, not the coarser one. This produces the final result, u^i_{AD}, which will be used in later stages to perform conservative updates of the CC buoyancy and velocity.

2. *Advect/diffuse scalars.* In our case, the only scalar that needs to be updated is the buoyancy, b. We do this by predicting $b^{n+1/2}$ at face centers which can then be used in a semi-implicit diffusive update. Just as before, we fill b^n's ghosts at CF interfaces using conservative linear interpolation in the tracing scheme and quadratic interpolation in the diffusive source term, but its physical BCs are problem dependent. After computing $\nabla^2 b^n$ over the level's interior, we fill its physical and CF interface ghosts using constant extrapolation.

$$
\begin{aligned}
b^{n+1/2} &= \mathrm{PPM}\left(b^n, u^i_{\mathrm{AD}}, \kappa\nabla^2 b^n, \frac{\Delta t}{2}\right) \\
S_b^{n+1/2} &= -\frac{1}{J}\partial_i J u^i_{\mathrm{AD}} b^{n+1/2} \\
(1-\frac{\kappa\Delta t}{2}\nabla^2)b^{n+1} &= (1+\frac{\kappa\Delta t}{2}\nabla^2)b^n + \Delta t S_b^{n+1/2}
\end{aligned}
$$

3. *Advect/diffuse velocity.* The details of this update are similar to the first step. We begin with a new prediction of the velocity at all face centers, this time using the projected, time-centered advecting velocity.

$$
\begin{aligned}
S^{n,\alpha} &= \nu\nabla^2 u^{n,\alpha} - b^n \hat{z}^\alpha + f_{\mathrm{E}}^{n,\alpha} + f_{\mathrm{I}}^{n,\alpha} \\
u_H^{\star,\alpha} &= \mathrm{PPM}\left(u^{n,\alpha}, u^i_{\mathrm{AD}}, S^{n,\alpha}, \frac{\Delta t}{2}\right) \\
u_H^i &= u_H^{\star,i} - g^{ij}\partial_j \phi
\end{aligned}
$$

To save an elliptic solve, we use the ϕ computed in step 1. Next, we copy u^i_{AD} to the components of u_H normal to the cell faces. This produces a better approximation of the advective source term since u_{AD} is exactly divergence-free while u_H is only approximately so. The advective source term is then calculated by

$$
\left[\frac{1}{J}\partial_j J u^j u^\alpha\right]^{n+1/2} \approx \frac{1}{J}\partial_j J u^j_{AD} u^\alpha_H.
$$

Ideally, the time–centered pressure gradient should be included in the source terms for the viscous solver, but this information is not known and would be expensive to compute. We instead include a lagged pressure gradient, $\nabla^i \pi^{n-\frac{1}{2}}$, which is freely available from the previous time step. This method, first introduced by Bell, Colella, and Glaz [84], was shown to produce a second–order velocity update by Brown, Cortez, and Minion [85]. We can now collect all of our source terms and solve for the new

velocity.

$$S^{n+1/2,\alpha} = -\left[\frac{1}{J}\partial_j J u^j u^\alpha\right]^{n+1/2} - \mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}(g^{\alpha i}\partial_i \pi^{n-1/2}) - b^{n+1/2}\hat{z}^\alpha$$

$$(1 - \frac{\nu\Delta t}{2}\nabla^2)u^{\star,\alpha} = (1 + \frac{\nu\Delta t}{2}\nabla^2)u^{n,\alpha} + \Delta t S^{n+1/2,\alpha}$$

The time-centered buoyancy is estimated as the average of b^n and b^{n+1}. All of the BCs in this step are the same as in step 1 with the exception of the pressure. The CF interface ghosts of $\pi^{n+1/2}$ are filled using a quadratic interpolation in space and conservative, linear interpolation in time from the coarser level. The physical BCs of $\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}(g^{\alpha i}\partial_i \pi^{n+1/2})$ are set by quadratic extrapolation, which has proven to be a more stable choice than homogeneous Neumann with a biased stencil. The stability properties of this choice were explored by Guy and Fogelson [86].

4. *Apply external forces.* Although we included the external forces, $\vec{f}_{\mathrm{E}}$ and $\vec{f}_{\mathrm{I}}$, into our source terms, $\vec{S}$, we only used these forces to promote accuracy during the characteristic tracing scheme and did *not* incorporate them into a complete update of the velocity. Since we have not specified the nature of these external forces, we leave it to the reader to apply the appropriate updating scheme. If $\vec{f}_{\mathrm{I}} \equiv 0$, then this update can be a simple midpoint method,

$$u^{\star,i} \leftarrow u^{\star,i} + \Delta t\, f_{\mathrm{E}}^{n+1/2,i},$$

but if there are stiff forces present, a more expensive operation may be needed.

5. *CC velocity projection.* In this final step, we remove the lagged pressure gradient from $u^{\star,i}$ and replace it with a more current estimate.

$$\nabla^2 \pi^{n+1/2} = \frac{1}{J}\partial_i J\left(\frac{1}{\Delta t}u^{\star,i} + g^{ij}\partial_j \pi^{n-1/2}\right)$$

$$u^{n+1,i} = u^{\star,i} + \Delta t\,\mathrm{Av}^{\mathrm{FC}\to\mathrm{CC}}\left(g^{ij}\partial_j \pi^{n-1/2} - g^{ij}\partial_j \pi^{n+1/2}\right)$$

During the elliptic solve for the pressure, we apply homogeneous Neumann BCs at the physical boundaries and interpolation from the coarser level at the CF interfaces. The interpolation used is quadratic in space and linear in time. Once $\pi^{n+1/2}$ is known, we apply quadratic extrapolation BCs on its CC gradient at the physical boundaries.

Section 4.3: Single-level update with $\bar{b}(z) \not\equiv 0$ and Coriolis effects

When a non-trivial background stratification is present, we perform a semi-implicit update of the terms that lead to buoyancy oscillations (BV update). Additionally, we can include Coriolis effects at little extra computational effort. The dynamical equations we wish to semi-implicitly solve are

$$\frac{\partial u^i}{\partial t} = -b\frac{\partial \xi^i}{\partial z} - f\varepsilon^i{}_{jk}\frac{\partial \xi^j}{\partial z}u^k - g^{ij}\partial_j p, \tag{4.1a}$$

$$\frac{\partial b}{\partial t} = N^2 w, \tag{4.1b}$$

$$0 = \partial_i J u^i, \tag{4.1c}$$

where we consider the special case $\vec{\Omega} = \hat{z}f/2$ in the Coriolis term.[2] The associated semi-implicit system is

$$u^{n+1,i} = u^{\star,i}_{\text{ex}} - \Delta t\left\{\left[\theta b^{n+1} + (1-\theta)b^n\right]\frac{\partial \xi^i}{\partial z} + f\varepsilon^i{}_{jk}\frac{\partial \xi^j}{\partial z}\left[\theta u^{n+1,k} + (1-\theta)u^{n,k}\right] - g^{ij}\partial_j p^{n+1/2}\right\}, \tag{4.2a}$$

$$b^{n+1} = b^{\star}_{\text{ex}} + \Delta t N^2\left[\theta w^{n+1} + (1-\theta)w^n\right], \tag{4.2b}$$

where it is assumed that an explicit advective/diffusive update has already been performed, taking $u^{n,i}$ and b^n to $u^{\star,i}_{\text{ex}}$ and $b^{\star}_{\text{ex}}$. In what follows, we analytically solve eqs. (4.2) for the final state and extract an elliptic equation for the pressure.

4.3.1: Derivation

We begin by defining the σ and ω block matrices and their powers as

$$\sigma = \begin{bmatrix} 0 & \frac{1}{N}\frac{\partial \xi^i}{\partial z} \\ -N\frac{\partial z}{\partial \xi^j} & 0 \end{bmatrix}, \quad \sigma\sigma = \begin{bmatrix} -\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j} & 0 \\ 0 & -1 \end{bmatrix}, \quad \sigma\sigma\sigma = -\sigma, \quad \ldots,$$

$$\omega = \begin{bmatrix} \varepsilon^i{}_{sj}\frac{\partial \xi^s}{\partial z} & 0 \\ 0 & 0 \end{bmatrix}, \qquad \omega\omega = \begin{bmatrix} \frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j} - \delta^i_j & 0 \\ 0 & 0 \end{bmatrix}, \quad \omega\omega\omega = -\omega, \quad \ldots,$$

noting that $\sigma\sigma + \omega\omega = -1$ and $\sigma\omega = \omega\sigma = 0$. The symbol δ^i_j is the Kronecker delta, or 3×3 identity matrix. In these terms, equations (4.2) can be written as

$$L(\theta)q^{n+1} = (q^{\star}_{\text{ex}} - q^n) + L(\theta - 1)q^n - \begin{bmatrix} g^{ij}\partial_j\left(\Delta t\, p^{n+1/2}\right) \\ 0 \end{bmatrix}, \tag{4.3}$$

[2] On Earth with synoptic frequency $\omega_E = 7.2921\times 10^{-5}\,\text{rad/s}$ and latitude Θ, the Coriolis parameter is $f = 2\omega_E\sin(\Theta)$. We use the so-called f-plane approximation by restricting f to be constant throughout the domain's extent and neglecting the horizontal component [4].

where $L(\theta) = 1 + \Delta t \theta N \sigma + \Delta t \theta f \omega$ and $q = (u^i, b)^T$. This can also be expressed in predictor-corrector format,

$$q^{\star\star} = L(\theta)^{-1}(q^{\star}_{\text{ex}} - q^n) + L(\theta)^{-1} L(\theta - 1) q^n \tag{4.4a}$$

$$q^{n+1} = q^{\star\star} - L(\theta)^{-1} \begin{bmatrix} g^{ij}\partial_j \left(\Delta t \, p^{n+1/2} \right) \\ 0 \end{bmatrix}. \tag{4.4b}$$

The finite, closed algebra formed by σ and ω allows us to completely express the inverse of $L(\theta)$ as $L(\theta)^{-1} = 1 + c_1\sigma + c_2\sigma\sigma + c_3\omega + c_4\omega\omega$. Requiring $L(\theta)^{-1}L(\theta) = 1$ gives us the coefficients,

$$c_1 = \frac{-\Delta t \theta N}{1 + (\Delta t \theta N)^2}, \qquad c_2 = \frac{(\Delta t \theta N)^2}{1 + (\Delta t \theta N)^2}, \qquad c_3 = \frac{-\Delta t \theta f}{1 + (\Delta t \theta f)^2}, \quad \text{and} \quad c_4 = \frac{(\Delta t \theta f)^2}{1 + (\Delta t \theta f)^2}, \tag{4.5}$$

which in turn allows us to write an expression for $L(\theta)^{-1}L(\theta - 1)$,

$$\begin{aligned} L(\theta)^{-1}L(\theta-1) &= [1 + c_1\sigma + c_2\sigma\sigma + c_3\omega + c_4\omega\omega]\,[1 + \Delta t(\theta-1)N\sigma + \Delta t(\theta-1)f\omega] \\ &= 1 - \frac{\Delta t N}{1 + (\Delta t \theta N)^2}\sigma + \frac{\Delta t^2 \theta N^2}{1 + (\Delta t \theta N)^2}\sigma\sigma - \frac{\Delta t f}{1 + (\Delta t \theta f)^2}\omega + \frac{\Delta t^2 \theta f^2}{1 + (\Delta t \theta f)^2}\omega\omega \\ &= 1 + \frac{c_1}{\theta}\sigma + \frac{c_2}{\theta}\sigma\sigma + \frac{c_3}{\theta}\omega + \frac{c_4}{\theta}\omega\omega, \end{aligned}$$

where c_1/θ is understood to be $-\Delta t N/[1 + (\Delta t \theta N)^2]$ when $\theta = 0$. A similar definition applies to all c_i/θ. Now, we can write an explicit expression for the predicted state,

$$\begin{aligned} q^{\star\star} &= (1 + c_1\sigma + c_2\sigma\sigma + c_3\omega + c_4\omega\omega)\,(q^{\star}_{\text{ex}} - q^n) + \left(1 + \frac{c_1}{\theta}\sigma + \frac{c_2}{\theta}\sigma\sigma + \frac{c_3}{\theta}\omega + \frac{c_4}{\theta}\omega\omega\right) q^n \\ &= q^{\star}_{\text{ex}} + \left(\frac{c_1}{\theta}\sigma + \frac{c_2}{\theta}\sigma\sigma + \frac{c_3}{\theta}\omega + \frac{c_4}{\theta}\omega\omega\right)(\theta q^{\star}_{ex} + (1-\theta)q^n). \end{aligned}$$

By defining q^θ, eqs. (4.4) are now expressed as

$$q^\theta = \theta q^{\star}_{ex} + (1-\theta)q^n \tag{4.6a}$$

$$q^{\star\star} = q^{\star}_{\text{ex}} + \left(\frac{c_1}{\theta}\sigma + \frac{c_2}{\theta}\sigma\sigma + \frac{c_3}{\theta}\omega + \frac{c_4}{\theta}\omega\omega\right) q^\theta \tag{4.6b}$$

$$q^{n+1} = q^{\star\star} - (1 + c_1\sigma + c_2\sigma\sigma + c_3\omega + c_4\omega\omega) \begin{bmatrix} g^{ij}\partial_j \left(\Delta t \, p^{n+1/2} \right) \\ 0 \end{bmatrix}. \tag{4.6c}$$

We only need to extract the velocity components from eqs. (4.6) since the buoyancy can be updated more simply using eq. (4.2b). Therefore, the velocity prediction is

$$\begin{aligned} u^{\star\star,i} &= u^{\star,i}_{\text{ex}} + \frac{c_1}{\theta}\left(\frac{1}{N}\frac{\partial \xi^i}{\partial z}\right) b^\theta + \frac{c_2}{\theta}\left(-\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right) u^{\theta,j} + \frac{c_3}{\theta}\left(\varepsilon^i_{sj}\frac{\partial \xi^s}{\partial z}\right) u^{\theta,j} + \frac{c_4}{\theta}\left(\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j} - \delta^i_j\right) u^{\theta,j} \\ &= u^{\star,i}_{\text{ex}} - \frac{\Delta t}{1+(\Delta t\theta N)^2}\frac{\partial \xi^i}{\partial z}\left(b^\theta + \Delta t\theta N^2 w^\theta\right) - \frac{\Delta t f}{1+(\Delta t\theta f)^2}\left[\varepsilon^i_{sj}\frac{\partial \xi^s}{\partial z} + \Delta t\theta f\left(\delta^i_j - \frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right)\right] u^{\theta,j}. \end{aligned}$$

Similarly, the velocity correction is

$$\begin{aligned} u^{n+1,i} &= u^{\star\star,i} - \left\{\delta^i_j + c_2\left(-\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right) + c_3\left(\varepsilon^i_{sj}\frac{\partial \xi^s}{\partial z}\right) + c_4\left(\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j} - \delta^i_j\right)\right\} g^{jk}\partial_k\left(\Delta t\, p^{n+1/2}\right) \\ &= u^{\star\star,i} - \left\{\delta^i_j - \frac{(\Delta t\theta N)^2}{1+(\Delta t\theta N)^2}\frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right. \\ &\qquad \left. - \frac{\Delta t\theta f}{1+(\Delta t\theta f)^2}\left[\varepsilon^i_{sj}\frac{\partial \xi^s}{\partial z} + \Delta t\theta f\left(\delta^i_j - \frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right)\right]\right\} g^{jk}\partial_k\left(\Delta t\, p^{n+1/2}\right). \end{aligned}$$

We now write the velocity update in its final form,

$$b^{\star\star} = b^\theta + \Delta t\theta N^2 w^\theta \tag{4.7a}$$

$$\mathcal{F}^i_j = \frac{\Delta t f}{1+(\Delta t\theta f)^2}\left[\varepsilon^i_{sj}\frac{\partial \xi^s}{\partial z} + \Delta t\theta f\left(\delta^i_j - \frac{\partial \xi^i}{\partial z}\frac{\partial z}{\partial \xi^j}\right)\right] \tag{4.7b}$$

$$g^{ij}_{\text{BV}} = g^{ij} - \frac{(\Delta t\theta N)^2}{1+(\Delta t\theta N)^2}\frac{\partial \xi^i}{\partial z}\frac{\partial \xi^j}{\partial z} - \theta\mathcal{F}^{ij} \tag{4.7c}$$

$$u^{\star\star,i} = u^{\star,i}_{\text{ex}} - \frac{\Delta t}{1+(\Delta t\theta N)^2}\frac{\partial \xi^i}{\partial z} b^{\star\star} - \mathcal{F}^i_j\, u^{\theta,j} \tag{4.7d}$$

$$u^{n+1,i} = u^{\star\star,i} - g^{ij}_{\text{BV}}\partial_j\left(\Delta t\, p^{n+1/2}\right). \tag{4.7e}$$

Note that the perturbed metric, g^{ij}_{BV}, can be alternatively expressed as

$$g^{ij}_{\text{BV}} = \frac{g^{ij}}{1+(\Delta t\theta f)^2} - \left[\frac{(\Delta t\theta N)^2}{1+(\Delta t\theta N)^2} - \frac{(\Delta t\theta f)^2}{1+(\Delta t\theta f)^2}\right]\frac{\partial \xi^i}{\partial z}\frac{\partial \xi^j}{\partial z} + \frac{\Delta t\theta f}{1+(\Delta t\theta f)^2}\varepsilon^{ijk}\frac{\partial z}{\partial \xi^k}.$$

Requiring eq. (4.7e) to be divergenceless results in an elliptic equation for the pressure,

$$\nabla^2_{\text{BV}} = \frac{1}{J}\partial_i J g^{ij}_{\text{BV}}\partial_j, \tag{4.8a}$$

$$\nabla^2_{\text{BV}}\left(\Delta t\, p^{n+1/2}\right) = \frac{1}{J}\partial_i J u^{\star\star,i}. \tag{4.8b}$$

If we assume f to be uniform throughout the domain, the Levi-Civita term in the elliptic operator can be abandoned since

$$\begin{aligned}\partial_i J f \varepsilon^i_{sj} \frac{\partial \xi^s}{\partial z} g^{jk} \partial_k p^{n+1/2} &= f \partial_i J \varepsilon^{isk} \frac{\partial z}{\partial \xi^s} \partial_k p^{n+1/2} \\ &= -f \epsilon^{zij} \partial_i \partial_j p^{n+1/2} \\ &\equiv 0.\end{aligned}$$

Also note that setting $f = \mathcal{F} = 0$ eliminates the Coriolis effects altogether.

We can garner a better understanding of eq. (4.7e) if we write it in the form of a projection and expand in powers of Δt,

$$\begin{aligned}u^{n+1,i} &= \left[\delta^i_j - g^{ik}_{\mathrm{BV}} \partial_k \left(\nabla^2_{\mathrm{BV}}\right)^{-1} \frac{1}{J} \partial_j J\right] u^{\star\star,j} \\ &= [\underbrace{\delta^i_j - g^{ik} \partial_k \left(\nabla^2\right)^{-1} \frac{1}{J} \partial_j J}_{\text{Standard projector}} + \underbrace{\mathcal{O}(\Delta t)}_{\text{Final update}}] u^{\star\star,j}.\end{aligned}$$

Thus the modified projection in eq. (4.7e) can be interpreted as a standard projector applied to $u^{\star\star,i}$, followed by a final explicit update. The standard projector alone is not equipped to send $u^{\star\star,i}$ to a divergence-free vector space. Instead, it carefully situates the irrotational component of $u^{\star\star,i}$ so that the final explicit update finishes the job of the standard projector – leaving $u^{n+1,i}$ divergenceless as required.

4.3.2: Split numerical algorithm

1. *Compute advecting velocities.* This is performed just as in the unsplit method, but with the buoyant forcing term, $-b\hat{z}^\alpha$, excluded.

2. *Advect/diffuse scalars.* This is performed exactly as in the unsplit method, which already excludes the advection term due to $\bar{b}$.

3. *Advect/diffuse velocity.* Again, this step is performed just as in the unsplit method, but with all buoyant forcing terms, $-b\hat{z}^\alpha$, excluded.

4. *Semi-implicit BV update.* To include the effects of eqs. (4.1) semi-implicitly without an additional elliptic solve, we alter the final CC velocity projection. We first perform as much of the update as we

can without the new pressure estimate via all but the last of eqs. (4.7). Then, we compute the pressure using a modified Poisson problem,

$$\nabla^2_{\mathrm{BV}}\left(\Delta t\,\pi^{n+1/2}\right) = \frac{1}{J}\partial_i J u^{\star\star,i}. \tag{4.9}$$

This Poisson problem uses the same BCs on $\pi^{n+1/2}$ and $g^{ij}_{\mathrm{BV}}\,\partial_j\pi^{n+1/2}$ as we did in the unsplit method's CC projection. Once the pressure is known, we can finalize the update,

$$\begin{aligned} u^{n+1,i} &= u^{\star\star,i} - g^{ij}_{\mathrm{BV}}\,\partial_j\left(\Delta t\,\pi^{n+1/2}\right), \\ b^{n+1} &= b^{\star}_{ex} + \Delta t N^2\left(\theta w^{n+1} + (1-\theta)w^n\right). \end{aligned}$$

4.3.3: The stability of the semi-implicit update

If we cast the dynamical eqs. (4.1) into the form $\partial_t q = Aq$ with $q = (u, v, w, b)^T$, the non-trivial eigenvalues of A are pure imaginary conjugate pairs, with magnitudes equal to $(N^2\sin^2\alpha + f^2\cos^2\alpha)^{1/2}$, where α is the angle of the internal waves with the vertical [87]. On Earth, f ranges from 0 at the equator to approximately $1.458\times10^{-4}\,\mathrm{rad/s}$ at the poles, while typical oceanic values of N are $\mathcal{O}(10^{-3})\,\mathrm{rad/s}$. We must find values of θ and Δt that cause these eigenvalues to lie completely within the stability region of the updating scheme and assess the impact on q. Noting that $N \gg f$ in this context, we will perform our analysis in the simplifying limit $f \to 0$. Without the Coriolis term, the basis of our update matrices reduces to $\{1, \sigma, \sigma\sigma\}$.

Since we wish to analyze the semi-implicit update without regard to any prior update, we will set $q^{\star}_{ex} \equiv q^n$. With these simplifications, the update equation (4.3) becomes

$$q^{n+1} = L^{-1}(\theta)L(\theta-1)q^n - L^{-1}(\theta)\begin{bmatrix} g^{ij}\partial_j\left(p^{n+1/2}\,\Delta t\right) \\ 0 \end{bmatrix}, \tag{4.10}$$

where $L(\theta)^{-1} = 1 + c_1\sigma + c_2\sigma\sigma$ and the coefficients are the same as in eqs. (4.5). Without loss of generality, we will also confine ourselves to a 2D Cartesian coordinate system and assume an initial state of the form $e^{i(kx+mz)}$. By requiring a divergenceless final velocity, we can algebraically eliminate $g^{ij}\partial_j p^{n+1/2}$ from eq. (4.10) and cast the update into the form $q^{n+1} = Sq^n$, where

$$S = \frac{1}{(1+\Delta t^2 N^2\theta^2)k^2 + m^2}\begin{bmatrix} pm^2 & -pkm & \Delta tkm \\ -pkm & pk^2 & -\Delta tk^2 \\ -\Delta tN^2km & \Delta tN^2k^2 & p(k^2+m^2) \end{bmatrix}$$

and

$$p = \frac{(1+\Delta t^2 N^2 \theta(\theta-1))k^2 + m^2}{k^2+m^2}.$$

The eigenvalues of S are zero and two complex conjugates. To eliminate the zero eigenvalue, we define

$$Q = \begin{bmatrix} \frac{-im}{\sqrt{k^2+m^2}} & \frac{ik}{\sqrt{k^2+m^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

which projects u onto its solenoidal component and leaves b undisturbed. Denoting α as the angle that the wavevector makes with the vertical, the resulting projected matrix has the form

$$QSQ^{T\star} = \frac{1}{1+\Delta t^2 N^2 \theta^2 \sin^2\alpha} \begin{bmatrix} p & -i\Delta t \sin\alpha \\ -i\Delta t N^2 \sin\alpha & p \end{bmatrix}.$$

This matrix has the same two complex conjugates eigenvalues that were present in S, and its determinant is readily found to be

$$\det[QSQ^{T\star}] = \frac{\Delta t^2(\theta-1)^2 N^2 \sin^2\alpha + 1}{\Delta t^2 \theta^2 N^2 \sin^2\alpha + 1}.$$

For a stable method, we require this determinant to be ≤ 1 which restricts $\theta \geq 1/2$, as expected from a θ-method.

For $\theta = 1/2$, the determinant is exactly 1 and the scheme is unconditionally stable, which is not surprising since it coincides with the standard Crank-Nicholson scheme. For $1/2 < \theta \leq 1$, damping is expected for all wavenumbers with $\alpha \neq 0$, $\alpha = 0$ being the degenerate case of purely horizontal motion. Expanding about Δt to get

$$\det[QSQ^{T\star}] = 1 + \Delta t^2(1-2\theta)N^2 \sin^2\alpha + O(\Delta t^4),$$

shows that the numerical scheme is most effective at damping waves propagating in a near horizontal direction (k large). In practice, a judicious choice of θ provides an extra degree of flexibility useful to damp noise that may accumulate at high frequencies. Of particular interest is the choice $\theta = 1/2 + \mathcal{O}(\Delta t)$, which was shown by Canuto, *et. al.* [88] to retain global second-order accuracy while damping the spurious oscillations that may arise with a Crank-Nicolson scheme.

The preceding analysis demonstrates the stability of our semi-implicit scheme in the $f \to 0$ limit. Given the relative sizes of N and f, this is a reasonable limiting case, but it is worthwhile to briefly consider the

$f \neq 0$ case as well. Let us rewrite the hyperbolic part of the update in Cartesian coordinates as

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \\ Nw \\ b \end{bmatrix} = \begin{bmatrix} 0 & -f & 0 & 0 \\ f & 0 & 0 & 0 \\ 0 & 0 & 0 & -N \\ 0 & 0 & N & 0 \end{bmatrix}\begin{bmatrix} u \\ v \\ Nw \\ b \end{bmatrix}.$$

Notice that we have altered our definition of q to exhibit the block nature of the update matrix, allowing us to analyze the (u, v) and the (Nw, b) blocks individually. Since each of these blocks are identical up to a multiplicative function, the stability properties of updating the (Nw, b) components of q are identical to those of the (u, v) components upon substituting $Nw \to u$, $b \to v$, and $N \to f$. Furthermore, the stability properties of using a semi-implicit scheme to update each of these blocks is well documented in literature discussing theta methods. For example, appendix D of Canuto, *et. al.* shows that stability requires $1/2 \leq \theta \leq 1$, which is precisely what we discovered in the $f \to 0$ limit. This statement holds true regardless of the relative magnitudes of N and f, but breaks down in the presence of a curvilinear system with a Jacobian determinant far from unity.

Section 4.4: Multi-level update

To incorporate our single-level algorithms into the framework of AMR, we need to take measures at the beginning and end of each multi-level (composite) time step. These measures are called initialization and synchronization routines and they are thoroughly described Martin, *et. al.* [49, 50], so we only provide an outline here.

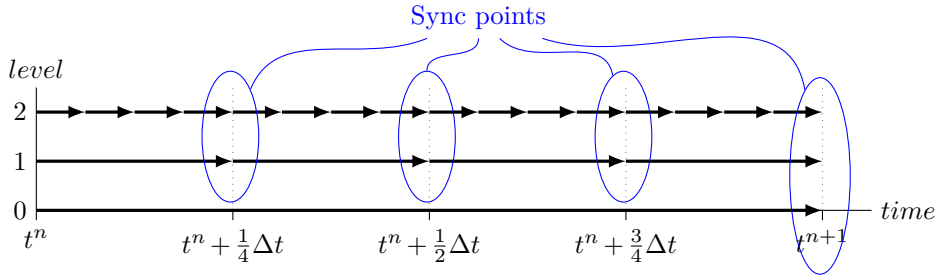


Figure 4.1: A diagram of one complete time step with subcycling.

Figure 4.1 provides a schematic of one complete time step of 3 levels with a refinement ratio of 4. Level 0 is the coarsest level and its grids span the entire domain. The level 1 grids only cover a portion of the domain and have $\Delta\xi^i$ that are 4 times smaller than that of level 0. Similarly, the level 2 grids are 4 times

finer than those of level 1. The CFL stability constraint allows us to update level 0 with a time step that is 4 times larger than that of level 1 and 16 times larger than that of level 2.

In order to march forward by Δt, we apply one of the single-level updating schemes described in section 4.2 or 4.3 to level 0. This brings level 0 to what is called a "sync point." Synchronization is the process of enforcing constraints and improving coarse level data with the knowledge of fine level data. This cannot be done until all finer levels are also at t^{n+1}, so before we finalize the level 0 update, we move on to level 1 and begin marching. This recursive process is called subcycling. It provides refinement in time in order to reduce both numerical diffusion and computational effort.

Section 4.5: Solving elliptic equations with AMR and anisotropy

The time marching schemes of sections 4.2 and 4.3 require both single-level and composite projections. We must therefore discuss how to solve the associated Poisson problems. Since our solver must deal with the anisotropy of coastal domains *and* the density stratification of the ocean, this is not always a trivial task. What's more, we must keep in mind that we are only concerning ourselves with non-hydrostatic flows – the corresponding Poisson problems for all of the following test cases are of maximal dimension. In this chapter, we will separately discuss three regimes delineated by the value of the *leptic ratio*, $\lambda = \min(\Delta\xi^m/H)$, where m is a horizontal index and H the local depth. The leptic ratio and effects of grid lepticity have been discussed in great detail by Vitousek and Fringer [23], Scotti and Mitran [89], and Santilli and Scotti [25].

4.5.1: Case: $\lambda \ll 1$ – Krylov methods with semicoarsening

For grids that are not considered leptic, we solve Poisson's equation on a single level using BiCGStab [90] with multigrid acceleration. To solve over an AMR hierarchy, we also perform multigrid (MG), but we restrict and prolong through the existing AMR levels. This process, thoroughly analyzed by Martin [49] and henceforth referred to as "Martin's algorithm," works quite well on isotropic Poisson problems. To bring Martin's algorithm into the context of stratified flows, we have added the following features.

1. *Generalized elliptic operator.* We solve $[\alpha + \beta f \partial_i \sigma^{ij} \partial_j]\phi = \rho$, where α and β are constant coefficients, f is any function of position, and σ^{ij} is any tensor field. We typically set $\alpha = 0$, $\beta = 1$, $f = J^{-1}$, and $\sigma^{ij} = Jg^{ij}$ for standard projections in curvilinear coordinates, but we can also accommodate more complex needs, such as the operator of equation (4.8a). This generalization also aids in solving diffusion problems with a very general subgrid model placed into σ^{ij}.

2. *Anisotropic refinement of AMR levels.* We have modified the core Chombo code to allow for different amounts of refinement in each direction. This is useful when modeling a vertically stratified fluid. It allows us to completely resolve the vertical stratification at the base level, requiring only horizontal refinement of the adaptive grids.

3. *Conditional semi-coarsening.* When grid anisotropy is detected, our MG algorithm restricts data only in those directions that promote isotropy at the bottom of the V-cycle. When solving over an AMR hierarchy, this is done in a way that respects the AMR levels that exist while tending towards isotropy on the MG levels created by the V-solver. All of this fits nicely into the framework of Martin's original AMR+MG algorithm.

To illustrate points number 2 and 3, consider a problem with 3 AMR levels that are anisotropically refined (see figure 4.2). Since we have assumed the grids are not leptic, we do not need to identify the vertical direction. The semi-coarsening algorithm simply coarsens in those directions that will promote isotropy. To this end, we will restrict ourselves to 2D without loss of generality (and without assuming hydrostasy) in an effort to keep figures 4.2 through 4.4 as simple as possible.

Suppose we wish to perform a MAC or CC projection on level 1 of figure 4.2 with the aid of semi-coarsening. Since the goal of semi-coarsening is to provide the Krylov solver with an isotropic problem, we need to coarsen level 1 in only the ξ^0 direction by a factor of 2. This will bring the grid scales, $\Delta\xi^i$, from $(2,4)$ to $(4,4)$, thereby achieving isotropy. We can then coarsen further if we wish to reduce the computational load, but we must preserve the isotropy that we worked to achieve by coarsening in *all* directions. This process is illustrated in figure 4.3. Once we reach the bottom of the V-cycle, we use BiCGStab to solve the simplified Poisson problem. Finally, we work our way back up the V-cycle, refining and relaxing our solution as we go. When we are performing a composite elliptic solve, the MG algorithm needs to coarsen data in a way that passes through the existing AMR levels and then tends towards isotropy. This pattern is illustrated in figure 4.4.

4.5.2: Case: $\lambda > 1$ – The leptic iterative method

When all of the horizontal wavenumbers supported by a discretization are smaller than the vertical ones, we say that the underlying grids are *leptic*. Such grids provide a considerable challenge for Krylov/MG methods because the spectrum of the discrete Laplacian splits into 2 disjoint sets, which drives the condition number up and reduces the effectiveness of the MG's relaxation methods. In many cases, semi-coarsening

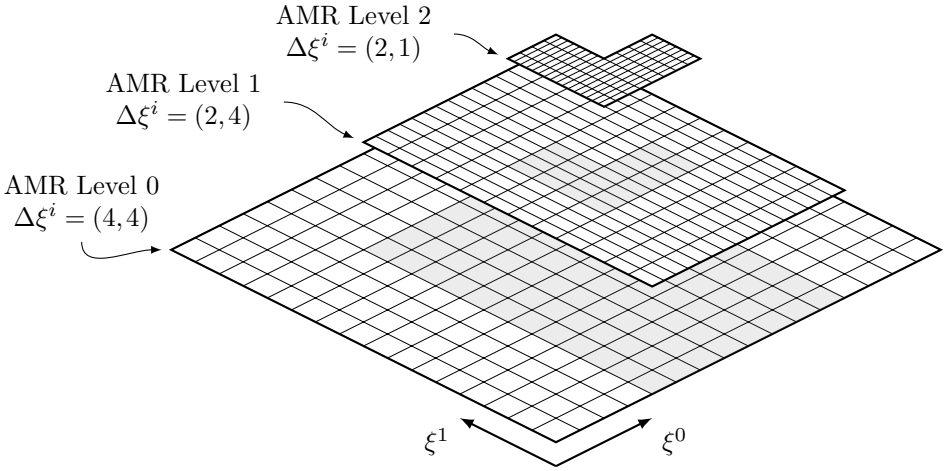


Figure 4.2: An example of a 2D AMR grid hierarchy. We restrict ourselves to 2D only to simplify the visualization of the grid hierarchy. In practice, these ideas are used to solve 3D (nonhydrostatic) Poisson problems. The grid scales, $\Delta\xi$, have been nondimensionalized and are provided to exhibit the grid anisotropy.

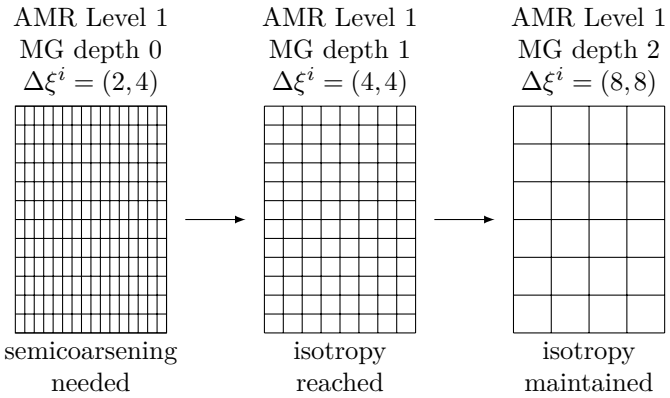


Figure 4.3: Restriction pattern of a V-cycle when performing a Poisson solve only on AMR level 1.

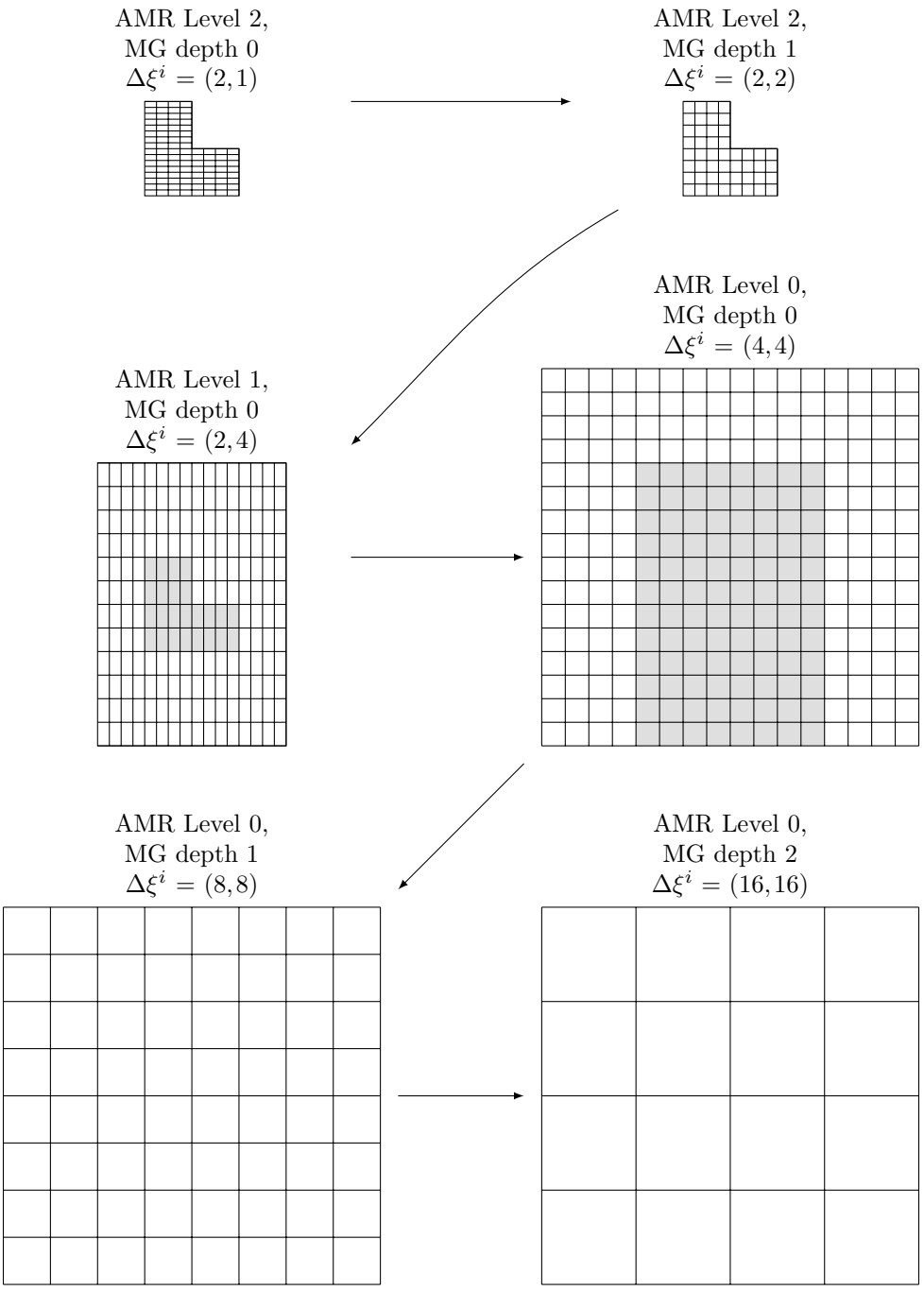


Figure 4.4: Restriction pattern of a V-cycle when performing a Poisson solve over AMR levels 0 through 2. We begin with relaxation at AMR level 2, MG depth 0. We then develop corrections iteratively through the coarser grids, eventually using BiCGStab on the coarsest grid at AMR level 0, MG depth 2. We then refine and relax the corrections back up to the finest grid, completing the V-cycle. The shadows of figure 4.2 have been reproduced to provide reference.

provides little relief because there is a practical limit to the depth of a V-cycle. For these situations, we use the *leptic iterative method*.

Formally, leptic iteration is an asymptotic expansion of Poisson's equation about a small parameter, $k/m \sim \lambda^{-1}$, where k and m are horizontal and vertical wavenumbers, respectively. Analytically, this expansion is not guaranteed to converge since it is always possible to force a mode with $k > m$. However, when the modes are restricted to a set of values determined by the choice of discretization, we can impose convergence through a horizontally coarse grid that satisfies the condition $\max(H/\Delta\xi^m) < \mathcal{O}(1)$.

For many cases in the ocean and atmosphere, where the domain's aspect ratio is far from unity, this can be ideal. For example, suppose we want to analyze the long-term evolution of a horizontally propagating internal wave in a stratified fluid. For such a problem, Vitousek and Fringer [23] showed that it is not possible to invoke the hydrostatic approximation without drastically sacrificing the dynamics. Furthermore, it would not be practical to slave the horizontal resolution to that of the vertical – the wave may travel hundreds or even thousands of multiples of H, requiring a coarse horizontal $\Delta\xi^i$ for tractibility. In this situation, we can choose a leptic base grid that completely resolves the fluid's stratification and use AMR to horizontally refine the travelling feature. The leptic method can then be used on these grids to accurately compute the nonhydrostatic pressure.

The method itself was first introduced by Scotti and Mitran [89] as a way providing the correct amount of dispersion needed to balance nonlinear steepening of internal waves propagating in a shallow, stratified ocean. The method was then cast into a more general mathematical form by Santilli and Scotti [25]. For a thorough analysis of the method, the reader is encouraged to read the original papers. We only provide an outline of the major details here. Most notably, Scotti and Mitran were able to show just how much dispersion is introduced when the leptic iteration is terminated at order M.

When we keep only the lowest ($M = 0$) order of the leptic asymptotic expansion, we recover the hydrostatic approximation with just as much computational effort. The interesting details arise when $M > 0$. In this case, we obtain a dispersion relationship that agrees with the exact dispersion relationship to order $\mathcal{O}(k^{2(M+1)})$. As a consequence, the short wavelength modes ($kH > 1$) travel at the wrong phase speed and must be filtered out. These fast modes are unstable for M odd, and stable for M even. The split time-marching scheme of section 4.3 was developed specifically to address this issue.

4.5.3: Case: $\lambda \lesssim 1$ – The leptic preconditioner

For grids that are neither leptic nor isotropic, both Krylov and leptic iteration fail. For this we need to develop a hybrid method. Luckily, these two methods effectively iterate on a different subset of the

error spectrum. In the Fourier sense, Krylov and MG methods tend to remove the highest-frequency errors supported by the grid while the leptic method only damps out those error modes that satisfy $kH < 1$. This complimentary convergence pattern has been shown in Santilli and Scotti to be very efficient on "borderline", $\lambda \lesssim 1$, grids.

The hybrid solver, composed of a triggered switching among leptic and Krylov methods, is nothing more than a preconditioning algorithm for the Krylov solver. We first use leptic iteration until convergence halts, which signals that the residual error has been reduced to mostly high frequencies. The result then serves as a preconditioned residual equation for the Krylov method, which can effectively remove these high frequency errors. When convergence halts, we once again turn to the leptic method, where the remaining low frequency errors can be removed. This process continues until all errors are sufficiently subdued.

As an interesting point, this form of preconditioner is exceptionally scalable. While most preconditioners, such as incomplete LU or Cholesky factorization, require the entire matrix operator to be computed and stored, the leptic preconditioner does not. It works entirely by splitting the 3D Poisson equation into a set of 1D Poisson equations that can be made tridiagonal and almost embarassingly parallel. In the case of Neumann BCs, a 2D Poisson problem must also be solved that generates the constants of integration produced by the set of 1D integrations. The resulting 2D operator is well-conditioned in the absence of the the high-frequency vertical modes. Therefore, in the context of 3D projection schemes, the leptic preconditioner has cost and scalability on the order of a 2D Krylov solver while promoting convergence of the far more expensive 3D Krylov solver.

CHAPTER 5: Test Cases[1]

Section 5.1: Lock exchange

To demonstrate the unsplit algorithm ($\bar{b} \equiv 0$) in Cartesian coordinates, we reproduce the well established lock exchange experiment and compare some basic features with the results of Härtel, *et. al.* [2, 3] (Härtel1 and Härtel2 from here on). To non-dimensionalize, we set both $\Delta b_T = b_{T\max} - b_{T\min}$ and the channel half-height to 1.

We distinguish cases by the choice of boundary conditions and two dimensionless quantities, the Grashof and Schmidt numbers. The Grashof number has the form of a squared Reynolds number and measures the ratio of buoyant to viscous forces. The Schmidt number is the ratio of kinematic viscosity to mass diffusivity. Substituting our unit scales, we have

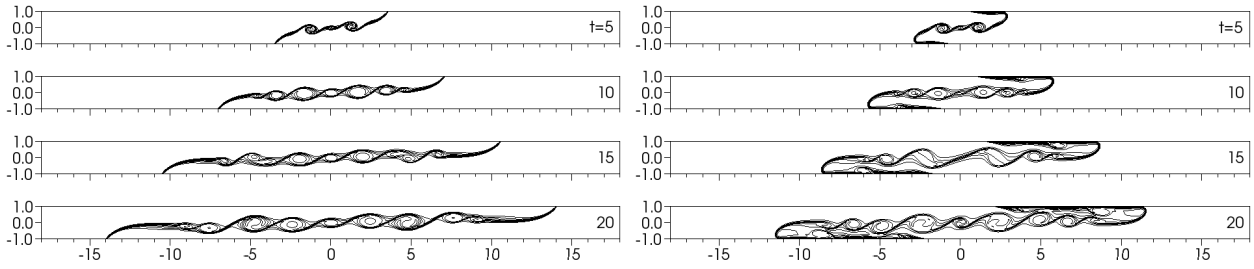
$$Gr = \left(\frac{1}{\nu}\right)^2 \quad \text{and} \quad Sc = \frac{\nu}{\kappa}.$$

We solve for the pressure using Krylov methods with multigrid acceleration and do not use the leptic method. Note that since our grids are isotropic, the multigrid algorithm will not use semicoarsening, but coarsens equally in all directions to preserve isotropy. This reduces the algorithm described in section 4.5.1 to a standard-issue V-cycle.

5.1.1: 2D flow: Speed of the wave front

Our first flow is a 2D computation in a channel with a non-dimensional length of 36. We initialize the velocity to zero and buoyancy to zero and one on the left and right halves of the domain, respectively. We use a Grashof number of 1.25×10^6 ($Re \approx 710$) and Schmidt number of 0.71. These are the same physical parameters as the free-slip and no-slip runs described in Härtel1. However, at 2304×128 cells, our computational grids are a bit more refined in order to facilitate multigrid acceleration and domain decomposition. We did not use AMR in the 2D case, and since the cell dimensions are isotropic, our MG scheme simply coarsens in all directions. Härtel also uses a very different solution method which employs a

[1]This chapter previously appeared as part of an article in the Journal of Computational Physics. The original citation is as follows: Edward Santilli and Alberto Scotti. The stratified ocean model with adaptive refinement (somar). *Journal of Computational Physics*, 291(0):60 – 81, 2015.



(a) Free-slip boundaries, $Gr = 1.25 \times 10^6$ $Sc = 0.71$.

(b) No-slip boundaries, $Gr = 1.25 \times 10^6$ $Sc = 0.71$.

Figure 5.1: Buoyancy contours of a two-dimensional lock-exchange flows. All length and time units are non-dimensionalized for comparison with Härtel, *et. al.* [2, 3].

vorticity-streamfunction formulation, RKW3 time integration, a spectral representation of the longitudinal direction (flow direction), and sixth order compact finite differencing scheme in the nonlinear terms.

Figure 5.1 shows the results of the the free-slip and no-slip runs at $t = 5$, 10, 15, and 20. These are contour plots of the buoyancy field and were designed for comparison with figures 3 and 11 in Härtel1. The position of the head, serving as a proxy for the time-averaged speed of the head, is indistignushable from Härtel1 at all times into the flow. We also get good agreement with the structure of the head and vortex patterns.

5.1.2: 3D, no-slip flow: The lobe-and-cleft instability

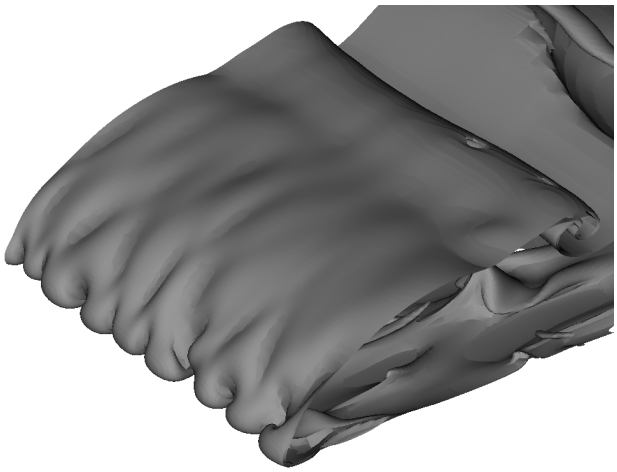
For the 3D case, we looked at a flow with $Gr = 1.5 \times 10^6$, $Sc = 0.71$, periodic BCs in the spanwise, or y–direction, and no-slip BCs elsewhere. The domain extents are $30 \times 3 \times 2$ units, non-dimensionalized again by the channel half-height. We discretize the domain at the base level into $480 \times 48 \times 32$ cells shared among 180 processors, each with a 16^3 computational grid. We allow for one adaptive level with a refinement factor of 4 in each direction. The base level cells are tagged for refinement based on two criteria:

1. Refine if neighboring cells have a buoyancy difference greater than 0.4.

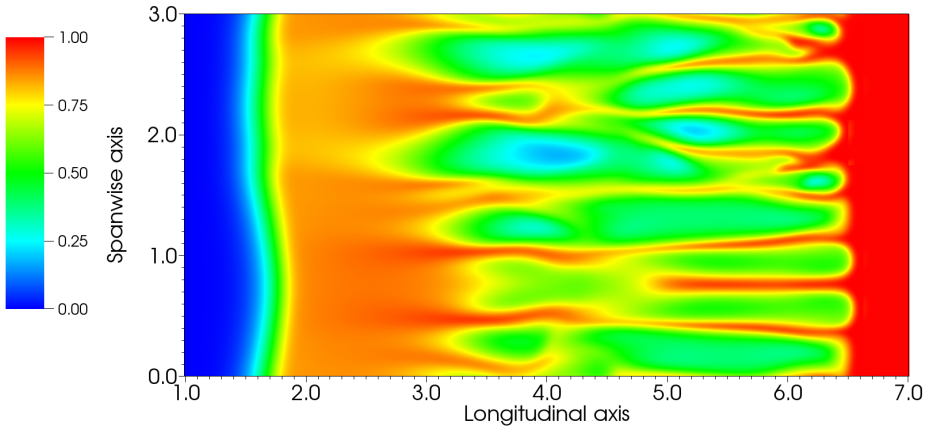
2. Refine if the local vorticity magnitude is greater than 35% of the global maximum.

The tagging and regridding process took place every 4 timesteps. For this run, we also used subcycling (temporal refinement on the fine level by 4).

Figure 5.2 shows two different visualizations of the wave front after 200 iterations ($t = 11.36$) when the lobe and cleft structure is clearly evident. The instability was provoked by providing an initial buoyancy



(a) Buoyancy isosurface ($b_T = 0.5$) depicting the instability at the front. The image is zoomed into the lower, left-traveling head.



(b) The buoyancy field at the upper vertical boundary. The image is zoomed into the lobe and cleft structure of the upper, right-travelling head.

Figure 5.2: The lobe and cleft instability. Both images were taken from a no-slip run with $Gr = 1.5 \times 10^6$ and $Sc = 0.71$ at t=11.36.

condition with a non-trivial spanwise dependence.

$$X_I = \frac{1}{40}\sin\left(\frac{\pi y}{15}\right),$$

$$b_T(\vec{x}, t=0) = \begin{cases} 0 & : x < X_I \\ 1 & : x > X_I \end{cases}.$$

We also used a tanh subcell profile for those cells that encompass X_I. If x_l and x_r are the positions of the left and right faces of a cell, then for all cells that satisfy $x_l < X_I < x_r$, the initial buoyancy is

$$f = \frac{X_I - x_r}{x_l - x_r},$$

$$b_T(\vec{x}, t=0) = \frac{1}{2}\left\{\tanh\left[m\left(2f-1\right)\right]+1\right\}.$$

The smoothing can be controlled by the parameter m. We used $m = 2$ for the run detailed here.

Although we used a different initial spanwise perturbation, the Froude number, $Fr \approx 0.57$, and the frontal structure are virtually identical with the corresponding results demonstrated in Härtel1. (figures 4 and 5 in [2]). The crucial difference is that at each timestep, our adaptive mesh uses between 18% and 33% of the 4.25 million cells needed by Härtel's simulation.

Section 5.2: Beam generation

We now follow Jalali, *et. al.* [91] and analyze the generation of internal waves over an obstacle in a linearly stratified fluid with tidal forcing at lab and large (oceanic) scales. The obstacle, illustrated in figure 5.3, is a ridge with 20% of each side at critical slope. We use no-slip boundary conditions at the sea floor, free-slip conditions at the rigid lid, and provide sponge layers at the lateral inflow/outflow boundaries to prevent wave reflections. Our choice of discretization (described in what follows) will restrict the grid lepticity to $\mathcal{O}(10^{-2})$, which rules out the use of the leptic method. We do, however, make heavy use of the semi-coarsening algorithm of section 4.5.1 since our horizontal grid scales will be on the order of 10 times larger than those of the vertical.

In consideration of the stratification and the bottom viscous boundary layer, we found that the finite volume PPM needed at least 512 cells evenly spaced in the vertical. For the lab-scale simulation, this spans 3.28 m and 2.9 m in the far and near fields, respectively. This is in contrast to Jalali's DNS, which relied on only 321 cells and a stretched grid. Despite using more cells in the vertical, we did not perform a DNS of the flow because the Stokes boundary layer thickness at the hill is approximately one-fourth of our cell

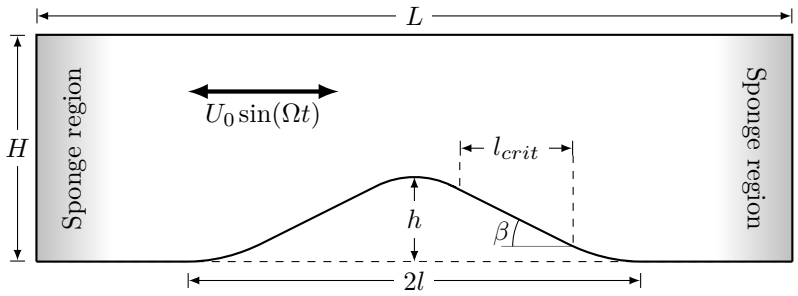


Figure 5.3: Diagram of beam generation problem setup (not drawn to scale). The background buoyancy increases linearly with depth at a rate $-d\bar{b}/dz = N^2$ for all runs.

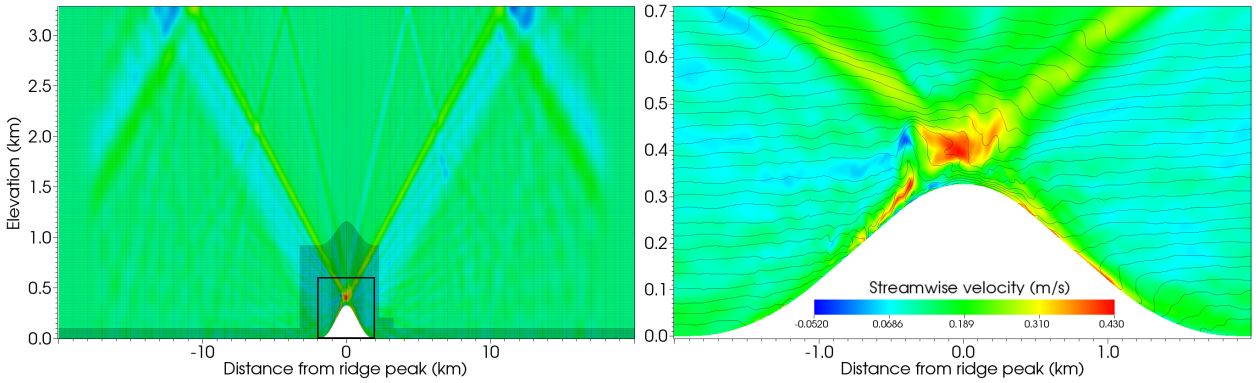


Figure 5.4: Large-scale calculation at a model ridge with critical slope at Ex = 0.066. The color plots represent the streamwise velocity field showing internal wave beams. In the left panel, the computational grid is superimposed to emphasize the higher resolution near the topographic relief and the beams after adaptive mesh refinement. The right panel shows the steep, near-bottom isopycnals.

height. Interestingly, since we did not use vertical stretching, the internal wave beams in the far field were more resolved in our study than in Jalali. Furthermore, using more cells to avoid stretching can prove more efficient for two reasons. First, the time step is limited by a CFL stability condition that is proportional to the minimum grid size. Second, given more cells, the semi-coarsening Poisson algorithm is able to reach a coarser and more isotropic bottom grid.

In the horizontal, the situation is similar – for the lab-scale run, we uniformly distribute 1280 cells over the entire domain length while Jalali used only 897 cells clustered near the generation site. For the large-scale run, we need more horizontal resolution since we expect more overturns due to an increased Reynold's number. To this end, we provide a base grid of 640 evenly spaced cells and and rely on anisotropic (horizontal only) refinement when needed. Eligibility for refinement by a factor of 4 is based on vorticity and undivided differences of density deviation. Figure 5.4 shows the adaptive mesh structure, along with the large-scale beam formation in the streamwise velocity component and isopycnal overturns.

The dimensions of the domain and obstacle for the lab-scale and large-scale runs are given in tables 5.1 and 5.2, respectively. Although more overturns were expected at the ridge and in the beams in the large-

Lab scale parameters

Dimensional				Non-dimensional	
L	$= 40\,\mathrm{m}$	l	$= 1.9\,\mathrm{m}$	ϵ	$= 1.0$
H	$= 3.28\,\mathrm{m}$	l_{crit}	$= 0.38\,\mathrm{m}$	Ex	$= 0.066$
β	$= 15^\circ$	h	$= 0.328\,\mathrm{m}$	Fr	$= 0.1$
U_0	$= 0.125\,\mathrm{m/s}$	ν	$= 10^{-6}\,\mathrm{m^2/s}$	Re	$= 1.6 \times 10^4$
Ω	$= 1.0\,\mathrm{rad/s}$	N^2	$= 14.93\,\mathrm{rad^2/s^2}$	$\mathrm{Re_s}$	$= 177$

Table 5.1: The physical parameters used in the lab-scale runs of Jalali (case CEX1) and our SOMAR tests.

Large (oceanic) scale parameters

Dimensional				Non-dimensional	
L	$= 40\,\mathrm{km}$	l	$= 1.9\,\mathrm{km}$	ϵ	$= 1.0$
H	$= 3.28\,\mathrm{km}$	l_{crit}	$= 380\,\mathrm{m}$	Ex	$= 0.066$
β	$= 15^\circ$	h	$= 328\,\mathrm{m}$	Fr	$= 0.1$
U_0	$= 0.125\,\mathrm{m/s}$	ν	$= 1.0 \times 10^{-4}\,\mathrm{m^2/s}$	Re	$= 1.6 \times 10^5$
Ω	$= 0.001\,\mathrm{rad/s}$	N^2	$= 1.493 \times 10^{-5}\,\mathrm{rad^2/s^2}$	$\mathrm{Re_s}$	$= 559$

Table 5.2: The physical parameters given to SOMAR for the large-scale run. The dimensionless parameters that govern the beam structure, ϵ, Ex, and Fr, are the same as in the lab-scale case, however we expect a much thinner boundary layer and more overturns throughout the beam.

scale run, the average structure of the beams were not expected to change since the problem's criticality and excursion were kept the same as in the lab-scale case. The mixing due to increased overturns was parametrized by an eddy viscosity, $\langle \nu_{sgs} \rangle$, at the generation sites and throughout the length of the beam. This profile is an idealized version of the cycle-averaged eddy viscosity generated by the LES studies of Rapaka, *et. al.*[92], obtained through personal communication.

To compare the vertical beam structure generated by SOMAR with Jalali's DNS, we compute the r.m.s of the baroclinic, horizontal velocity component, u_{bc}, over the seventh tidal cycle. We then normalize by the maximum tidal velocity, U_0, and consider a vertical cross-section at $x/l = -1$. Since Jalali's data is 3D, it is also averaged in the spanwise direction. The results are provided in figure 5.5. The relative differences among the peak velocities of Jalali and SOMAR are approximately 4.2% in the single-level run and 2.7% in the AMR run.

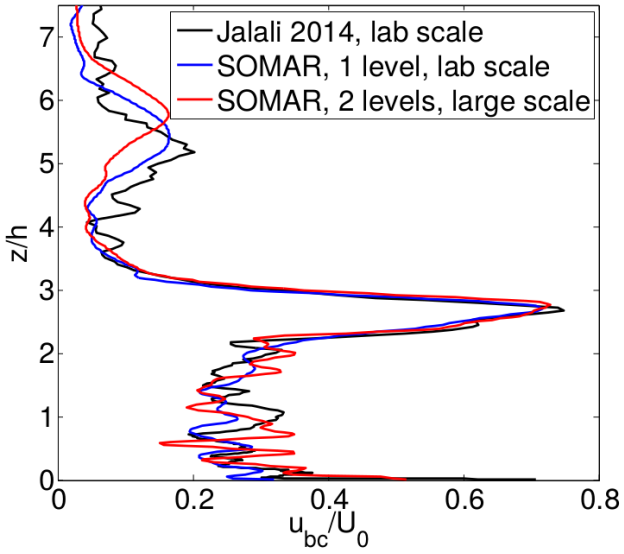


Figure 5.5: A comparison of the internal wave beam using three solution methods. The profiles shown are the root-mean-square, horizontal, baroclinic velocity sections at $x/l = -1$ over the seventh tidal cycle.

Section 5.3: DJL solution

An exact solution of the Euler equations (inviscid, non-rotating form of (1.17)) for a wave travelling in a two-dimensional channel with an undisturbed buoyancy distribution, $\bar{b}(z)$, can be written as

$$b_T(\theta, z) = \bar{b}(z - \eta(\theta, z)), \tag{5.1a}$$

$$\psi(\theta, z) = c\eta(\theta, z), \tag{5.1b}$$

$$\theta = x - ct, \tag{5.1c}$$

where the streamfunction, ψ, and the total buoyancy distribution, b_T, are expressed in terms of the isopycnal displacement, η, and the wave speed, c. We obtain η and c by solving the no-shear limit of the Dubreil-Jacotin-Long (DJL) equation [93, 94, 95]

$$\nabla^2\eta + \frac{N^2(z-\eta)}{c^2}\eta = 0, \tag{5.2a}$$

$$\eta(x, 0) = \eta(x, H) = 0, \tag{5.2b}$$

$$\lim_{x\to\pm\infty} \eta(x, z) = 0, \tag{5.2c}$$

where $N^2(z) = -\partial_z\bar{b}$. This nonlinear eigenvalue problem admits a family of solutions (η_E, c_E) parameterized by the total energy of the wave $E > 0$. Note that in the limit $E \to 0$ the DJL equation recovers the standard Taylor-Goldstein equation. For a given value of E, eqs. (5.2) can be solved numerically using the technique

described by Stastna, *et. al.* [96, 97]. Once the displacement and speed are known, eqs. (5.1) at $t = 0$ provides an initial condition for SOMAR.

5.3.1: 2D, solitary wave propagation

For this test, we consider a channel $128\text{m} \times 1\text{m}$ filled with water ($\rho_0 = 10^3 \text{Kg/m}^3$) with a background stratification given by

$$\bar{b}(z) = \frac{\tanh\left(\frac{H-z_0}{\delta}\right) - \tanh\left(\frac{z-z_0}{\delta}\right)}{\tanh\left(\frac{H-z_0}{\delta}\right) - \tanh\left(\frac{-z_0}{\delta}\right)},$$

where $z_0 = 0.8\text{m}$ is the pycnocline's center position and $\delta = 0.1\text{m}$ is its thickness. We generate a DJL wave with a total (kinetic and potential) energy per unit of spanwise length equal to $410\,\text{Jm}^{-1}$, quite close to the conjugate state solution, that is the solution with largest displacement (top panel of figure 5.6). The exact wave solution travels at a speed $c = 0.45\,\text{m/s}$, with a maximum displacement of 33 cm, or 1/3 of the depth of the water column. We use no-slip conditions at the top and bottom boundaries and non-reflecting boundaries (sponge layers) at the lateral boundaries that damp the velocity and buoyancy deviation to zero.

We discretize the domain into 32×32 cells and provide 2 adaptive levels. The base level (level 0) has a refinement factor of 2 in all directions while level 1 has a refinement factor of 4×1. Since the aspect ratio of the cells on the base level is 128 to 1, there is no hope of reaching isotropy using semi-coarsening. We therefore turn to the leptic method. Levels 0 and 1 have grid lepticities of 4 and 2, respectively, so we use the leptic method as a stand-alone Poisson solver for the pressure as described in section 4.5.2. On the finest level (level 2), the grid lepticity is 0.5, so we use the leptic solver as a preconditioner for our Krylov solver as described in section 4.5.3.

It should be pointed out that the horizontal discretization on the coarsest level provides inadequate resolution, with only 2 to 3 cells across the span of the wave. The finest level, however, provides approximately 20 cells across the span of the wave, making the adaptively refined levels crucial to the proper evolution of the waveform. This demonstration therefore serves to test the interplay of several ideas put forth in this paper: the anisotropic adaptivity of our grids, the leptic method as a level and composite Poisson solver and preconditioner, and the split algorithm of section 4.3.

Due to the physics of these waves, the disturbance associated with them extends across the entire water column. Thus, if a cell is tagged for refinement, we tag for refinement the entire column that contains that cell. In practice we tag entire vertical sections based on the value of the buoyancy deviation projected onto

the vertical structure of mode-1 waves. That is, we solve the linear eigenvalue equation,

$$\frac{\partial^2 \varphi}{\partial z^2} + \frac{N(z)^2}{a^2}\varphi = 0,$$
$$\varphi(0) = \varphi(H) = 0,$$

for the vertical structure function, $\varphi_0(z)$, associated with the largest wave speed, a_0. Then we project the buoyancy deviation to get an amplitude function,

$$A(x) = \int_0^H b(x,z)\,\varphi_0(z)\,dz,$$

which is used to tag entire vertical sections that satisfy $|A(x)| > A_{\text{thresh}}$. The parameter A_{thresh} is chosen small enough so that the fine grids do not fill the domain, but large enough that the entire waveform is refined. We found $A_{\text{thresh}} = 0.5$ to be appropriate based on coarse grid trials.

The DJL solution is an *exact* travelling wave solution to the nonlinear Euler equations and therefore our initial waveform should remain intact for all $t > 0$. As the bottom panel of figure 5.6 shows, this is indeed the case. After 200s, the depression has travelled 90 channel heights, which coincides with the expected speed of $c = 0.45\,\text{m/s}$ given by the DJL equation. Notice that this is significantly faster than the speed of linear disturbances for this particular stratification, $c_0 = 0.35\,\text{m/s}$. Equally important, the amplitude and overall form of the depression has not noticeably changed. The wave does exhibit some unbalanced dispersion, as is expected of any second-order numerical scheme, but this effect is small. More precisely, by integrating over $x = x_{\text{trough}} \pm 4\text{m}$ and $z = 0$ to 1, we find that the waveform has lost 2.5% of its total initial energy while the entire domain has lost only 0.3%. The 2.2% difference remains in the wake as dispersive numerical error. The color plots of figure 5.6 show the fraction of the maximum initial kinetic energy that is extracted from the travelling wave. Note that we are using a logarithmic scale and the error in the trailing end of the waveform is $< \mathcal{O}(10^{-3})$.

5.3.2: 3D, solitary wave propagation

As a final demonstration, we embed the solitary wave of section 5.3.1 into a 3D domain of length $512 \times 512 \times 1$, non-dimensionalized by the domain height. Our time scales are non-dimensionalized by our choice of $b_{\max} - b_{\min} = 1$. To produce a practical test of our numerical schemes, we set up the wave so that it travels obliquely to the grid orientation. More precisely, if we define b_T^{DJL} and $\vec{u}^{\text{DJL}}$ as the solution to Euler's equations that we used as the initial condition in the 2D demonstration, then we can embed that solution

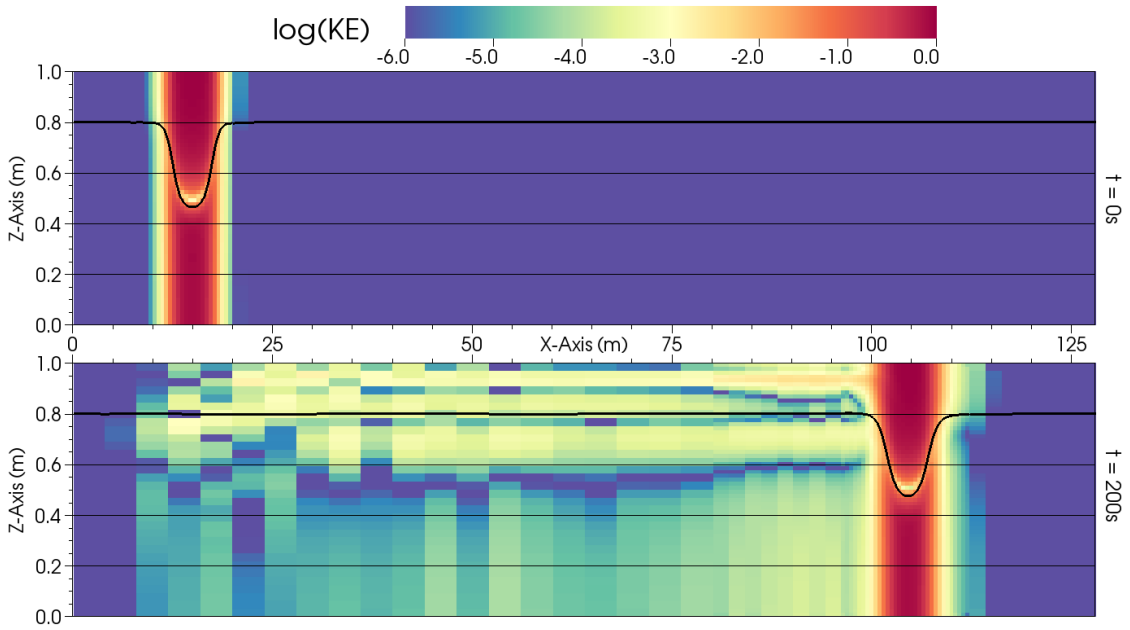


Figure 5.6: The solution to the DJL equation propagating in a long, thin channel. The thick black line traces the median isopycnal and the color plot depicts log(KE), where the energy has been normalized by its maximum value at $t = 0$. The top panel illustrates the initial waveform. The bottom panel shows that after 200 seconds, the waveform has travelled approximately 90 channel heights, but only a trace amount of energy (colors on the blue end of the spectrum are less than $\mathcal{O}(10^{-3})$) has leaked from the trailing end.

into our 3D domain as follows

$$\begin{aligned}
\vec{x}\,' &= \left(R^{-1}\cdot\vec{x}\right) - \vec{x}_0\,' \\
E\left(y'\right) &= \frac{1}{2}\tanh\left(\frac{y'+96}{6.4}\right) - \frac{1}{2}\tanh\left(\frac{y'-96}{6.4}\right) \\
b_T\left(\vec{x}\right) &= E\left(y'\right)\, b_T^{\mathrm{DJL}}\left(x', z\right) \\
\vec{u}\left(\vec{x}\right) &= E\left(y'\right)\, R\cdot\vec{u}^{\mathrm{DJL}}\left(x', z\right),
\end{aligned}$$

where R is the standard 45° rotation matrix about the z-axis, $\vec{x}\,'$ are the rotated coordinates, $\vec{x}_0\,'$ is an offset used to push the initial wave into the domain, and $E(y')$ is an envelope function used to prevent the extruded waveform from running into the sponge regions at each lateral boundary. For brevity, we omitted the explicit functional dependence of $\vec{x}\,'$ on $\vec{x}$ in the last two definitions. The top left panel of figure 5.7 shows how the resulting embedded wave is situated in the plane of the undisturbed pycnocline at $z = 0.8$. The dashed line represents the x'-axis, which serves as the "center of extrusion." The top panel of figure 5.8 shows the initial wave in the $x'z$-plane, that is, at the center of extrusion. From this perspective, the initial condition is identical to that of the 2D demonstration, but the grids are coarser by a geometric factor of $\sqrt{2}$.

In our units, the exact wave solution travels at a speed $c = 0.45$ and the isopycnals have a maximum

displacement of 1/3 of the domain height. We use no-slip conditions at the top and bottom boundaries and non-reflecting boundaries (sponge layers) at the lateral boundaries that damp the velocity and buoyancy deviation to zero. We discretize the domain into $128 \times 128 \times 32$ cells and provide 2 adaptive levels. The base level (level 0) has a refinement factor of 2 in all directions while level 1 has a refinement factor of $4 \times 4 \times 1$. We tag entire vertical columns of cells for refinement by observing the buoyancy deviation projected onto the vertical structure function, as explained in section 5.3.1. Also, since the cell sizes are identical to those of the 2D demonstration on all levels, our Poisson solvers follow the same prescription – use the stand-alone leptic method as described in section 4.5.2 on levels 0 and 1 and use the leptic preconditioner with a Krylov solver as described in section 4.5.3 on level 2.

Due to the envelope that tapers to zero, the entire length of the depression does not behave as a solitary wave. Figure 5.7 shows that these tapered ends travel slower than the bulk of the waveform, where $E(y') \approx 1$. This shear drains energy from the disturbance, leaving a wake in its path. Although this energy sink eventually destroys the solitary wave, this effect is not immediate throughout the entire disturbance. By $t = 600$, the effect of the shear is noticeably distorting the solitary wave in both figures 5.7 and 5.8. Despite this, the wave exhibits very small amounts of dispersion throughout the simulation as illustrated by figure 5.8 and quantified by the feature's almost constant speed of 0.449 ± 0.007. This is right on target for the speed of the solitary wave, but significantly higher than the speed of linear disturbances for this stratification, $c_0 = 0.35$.

Without AMR, a computational domain with $1024 \times 1024 \times 64$ cells would be needed to match the resolution provided by SOMAR's finest adaptive level. At each timestep, SOMAR's entire grid hierarchy never used more than 6% of the cells needed by a single-level solver. Furthermore, if we consider the fact that SOMAR also refines in time, this 3D simulation ultimately updated only 4.5% of the cells that would have been needed by a single-level solver. This simulation took approximately 12 hours to complete on 32 cores using 2.93GHz Intel X5670 processors.

To put this last simulation into context, consider a recent numerical study of nonlinear solitary waves propagating across the South China Sea [16], which used a mature, highly optimized non-hydrostatic solver (SUNTANS). To propagate nonlinear waves across the South China Sea on a grid consisting of 11 million grid points using a cluster with similar capabilities to the one employed in our simulation, SUNTANS required about 0.1 CPU hours per time step. Contrast this with our code, which requires 0.04 CPU hours per fine-level time step to run on an *effective* grid 6 times larger. Moreover our code, which is run nominally inviscid, is only very mildly dissipative. At the end of the simulation, the total energy in the domain is 92% of the energy present at $t = 0$.

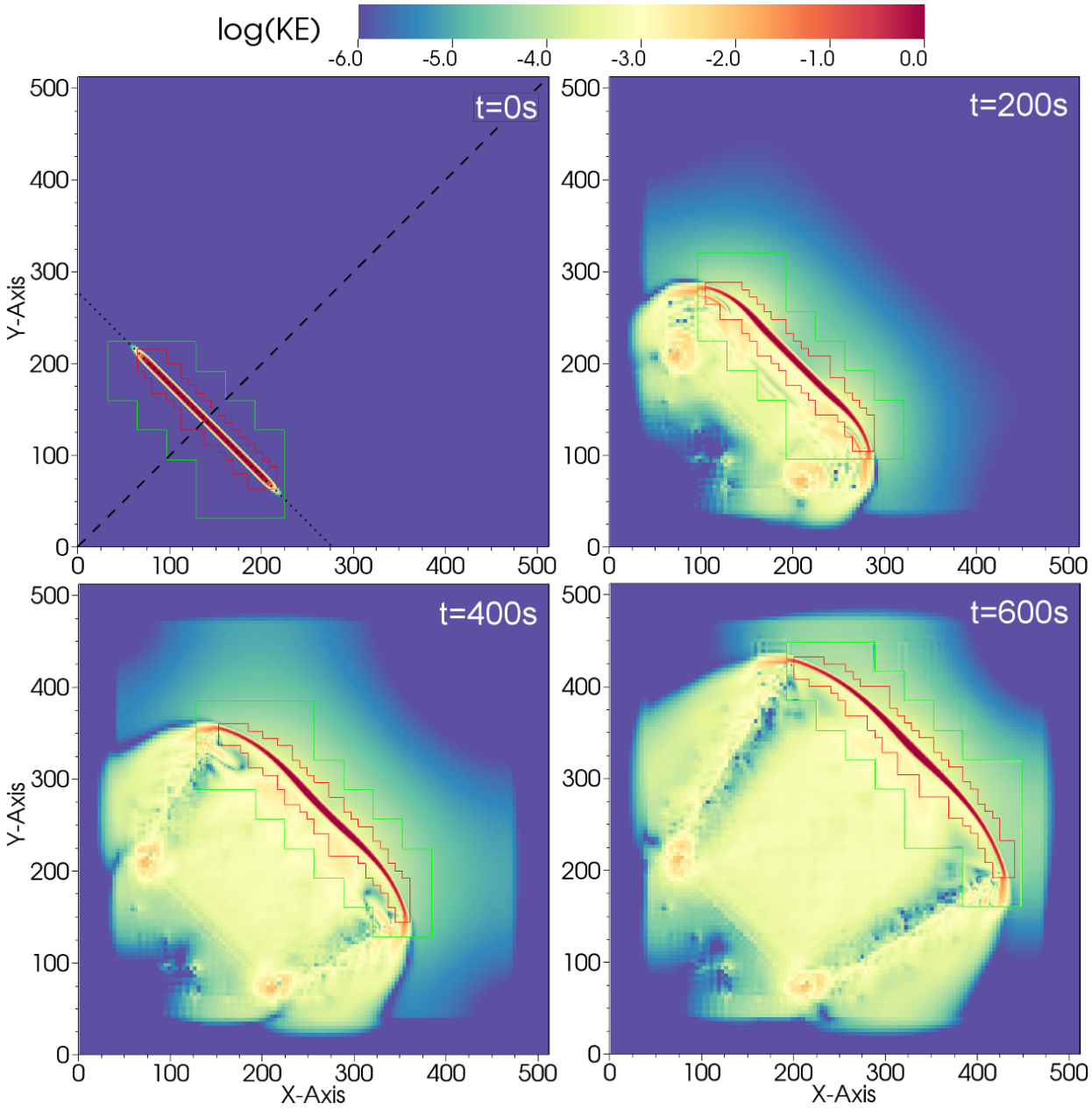


Figure 5.7: An overhead view of the 3D solitary wave's evolution. Each panel is a horizontal slice through the domain at $z = 0.8$, the elevation of the unperturbed pycnocline. The color plots depict log(KE), where the energy has been normalized by its maximum initial value. The top right panel shows the initial structure of the waveform. The dashed line represents the x'-axis, or "center of extrusion," while the dotted line illustrates the y' direction through which the 2D DJL solution was extruded and masked with a tanh envelope. The green and red markers outline the extents of the first and second adaptively refined levels, respectively. All lengths have been non-dimensionalized by the domain height.

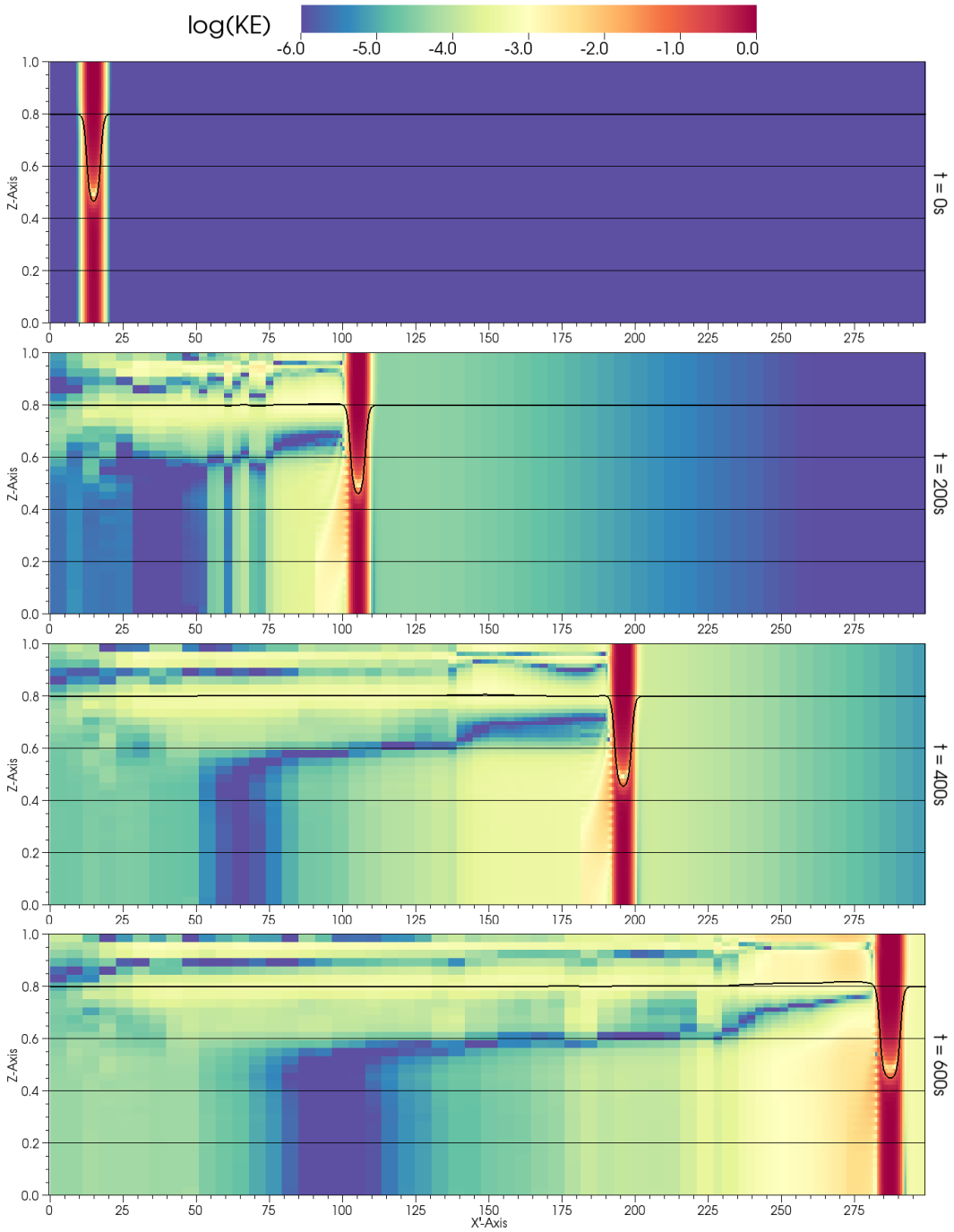


Figure 5.8: A lateral view of the 3D solitary wave's evolution. Each panel is a slice through the x'-axis (dashed line in the top left panel of figure 5.7). The origin of this slice has been shifted for comparison with figure 5.6. The thick black line traces the median isopycnal and the color plot depicts log(KE), where the energy has been normalized by its maximum value at $t = 0$. The energy left in the wake is of order 10^{-2}, which is greater than the results of the 2D simulation. This is especially prominent in the final time step because the shear produced at the edges of the wave (as shown in figure 5.7) has migrated into the center of the feature. After 600 seconds, the wave has travelled approximately 270 channel heights, which agrees well with its expected speed of $c_0 = 0.45$ channel heights per second.

CHAPTER 6: Summary and Future Work

We have presented in this document the Stratified Ocean Model with Adaptive Refinement (SOMAR), which combines the flexibility of Adaptive Mesh Refinement with a set of numerical tools specifically optimized to model baroclinic oceanic flows over a range of scales that straddle the hydrostatic-nonhydrostatic divide. This is a particularly challenging regime, since the traditional techniques that work well in nonhydrostatic flows characterized by moderate domain anisotropy become less and less effective as the anisotropy of the domain increases, especially in three-dimensional simulations.

SOMAR solves the two- or three-dimensional Navier-Stokes or Euler equations in the Bousinnesq approximation in arbitrary coordinates, which allows the inclusion of non-trivial topography. When a background stratification is present, the underlying hyperbolic problem that physically gives rise to internal waves is dealt with a special time-integrator, which ensures stability.

Three test cases are discussed in this paper. In the first test, we consider the evolution of two- and three-dimensional gravity currents released via a lock-exchange mechanism. This test is meant to evaluate the overall accuracy of the Navier-Stokes hydrodynamic solver. In both dimensions, the results agree very well with published benchmark results, though in the three-dimensional case, due to the use of Adaptive Refinement, our simulation employed as little as 18% and no more than 33% of the grid points used in the benchmark simulation.

In the second test, we consider a tidal flow over a ridge generating an internal wave beam. The benchmark is a Direct Numerical Simulation at laboratory scale, though we also consider a similar set up but at field scales. This test shows how the code deals with topography. The comparison with the DNS benchmark yields essentially identical results.

Finally, we consider the propagation of a specific type of nonlinear internal wave in a long stratified channel, for which an exact solution of the 2D Euler equations is known. The purpose of the test is to assess the effects of numerical dissipation and dispersion, using multiple levels of refinement, on the propagation of internal waves. In the test, the wave propagates over a distance of 270 channel depths with minimal distortion, a testament to the low numerical dissipation and dispersion of the scheme.

The tests considered indicate that SOMAR is a robust and effective platform for the numerical simulation of a range of baroclinic flows that are encountered in regional and coastal ocean simulations. The hierarchical nature of the regridding process also opens new possibilities, such as the ability to embed a highly resolved

Large Eddy Simulation within a SOMAR simulation, which is currently being pursued.

REFERENCES

[1] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2011. http://www.mcs.anl.gov/petsc.

[2] C. Härtel, E. Meiburg, and F. Necker. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 1. Flow topology and front speed for slip and no-slip boundaries. *Journal of Fluid Mechanics*, 418:189–212, September 2000.

[3] C. Härtel, F. Carlsson, and M. Thunblom. Analysis and direct numerical simulation of the flow at a gravity-current head. Part 2. The lobe-and-cleft instability. *Journal of Fluid Mechanics*, 418:213–229, September 2000.

[4] J. Pedlosky. *Geophysical Fluid Dynamics*. Springer study edition. Springer New York, 1992.

[5] J. Pedlosky. *Waves in the Ocean and Atmosphere: Introduction to Wave Dynamics*. Springer, 2003.

[6] D. B. Haidvogel and A. Beckmann. *Numerical ocean circulation modeling*. Imperial College Press, 1999.

[7] T. Kuhlbrodt, A. Griesel, M. Montoya, A. Levermann, M. Hofmann, and S. Rahmstorf. On the driving process of the Atlantic meridional overturning circulation. *Rev. Geophys.*, 45(RG2001):RG2001, 2007.

[8] G.N. Ivey, K.B. Winters, and J.R. Koseff. Density stratification, turbulence, but how much mixing? *Annual Review of Fluid Mechanics*, 40(1):169–184, 2008.

[9] Kirk Bryan. A numerical method for the study of the circulation of the world ocean. *Journal of Computational Physics*, 4(3):347–376, 1969.

[10] B Fox-Kemper and D Menemenlis. Can large eddy simulation techniques improve mesoscale rich ocean models? *Ocean modeling in an eddying regime*, pages 319–337, 2008.

[11] P. Sagaut. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Scientific Computation. Springer, 2006.

[12] Alberto Scotti. Large eddy simulation in the ocean. *International Journal of Computational Fluid Dynamics*, 24(10):393–406, 2010.

[13] Karl R. Helfrich and W. Kendall Melville. LONG NONLINEAR INTERNAL WAVES. *Ann. Rev. of Fluid Mech.*, 38(1):395–425, 2006.

[14] John Marshall, Helen Jones, and Christopher Hill. Efficient ocean modeling using non-hydrostatic algorithms. *Journal of Marine Systems*, 18(1âĂŞ3):115 – 134, 1998.

[15] O.B. Fringer, M. Gerritsen, and R.L. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling*, 14(3âĂŞ4):139 – 173, 2006.

[16] Z. Zhang, O. B. Fringer, and S. R. Ramp. Three-dimensional, nonhydrostatic numerical simulation of nonlinear internal wave generation and propagation in the south china sea. *Journal of Geophysical Research.Oceans*, 116(5), 2011.

[17] Dujuan Kang and Oliver Fringer. Energetics of barotropic and baroclinic tides in the Monterey Bay area. *Journal of Physical Oceanography*, 42(2):272–290, 2012.

[18] S.M. Jachec, O.B. Fringer, M.G. Gerritsen, and R.L. Street. Numerical simulation of internal tides and the resulting energetics within Monterey Bay and the surrounding area. *Geophys. Res. Lett.*, 33, 2006. doi:10.1029/2006GL026314.

[19] C Guo, X Chen, V Vlasenko, and N Stashchuk. Numerical investigation of internal solitary waves from the luzon strait: Generation process, mechanism and three-dimensional effects. *Ocean Modelling*, 38(3):203–216, 2011.

[20] V Vlasenko and N Stashchuk. Three-dimensional shoaling of large-amplitude internal waves. *Journal of Geophysical Research: Oceans (1978–2012)*, 112(C11), 2007.

[21] Vasiliy Vlasenko, Jose C Sanchez Garrido, Nataliya Stashchuk, Jesus Garcia Lafuente, and Miguel Losada. Three-dimensional evolution of large-amplitude internal waves in the strait of gibraltar. *Journal of Physical Oceanography*, 39(9):2230–2246, 2009.

[22] Abbas A. Dorostkar. *Three-dimensional dynamics of nonlinear internal waves*. PhD thesis, Queen's University, Kingston, Ontarion, Canada, 2012.

[23] Sean Vitousek and Oliver B. Fringer. Physical vs. numerical dispersion in nonhydrostatic ocean modeling. *Ocean Modelling*, 40(1):72 – 86, 2011.

[24] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64 – 84, 1989.

[25] Edward Santilli and Alberto Scotti. An efficient method for solving highly anisotropic elliptic equations. *J. Comput. Phys.*, 230(23):8342–8359, September 2011.

[26] G. K. Batchelor. Small-scale variation of convected quantities like temperature in turbulent fluid part 1. general discussion and the case of small conductivity. *Journal of Fluid Mechanics*, 5:113–133, 1 1959.

[27] Stephen M Griffies. *Fundamentals of ocean climate models*. Princeton University Press Princeton, 2004.

[28] S.R. Massel. *Ocean Surface Waves: Their Physics and Prediction*. Advanced series on ocean engineering. World Scientific, 1996.

[29] R.E. Meyer. *Introduction to Mathematical Fluid Dynamics*. Dover Books on Physics Series. Dover Publications, 1971.

[30] Lynne D. Talley. Salinity patterns in the ocean. In *Encyclopedia of Global Environmental Change (ISBN 0-471-97796-9) Editor-in-Chief*, volume 1, pages 629–640. John Wiley and Sons, 2002.

[31] S.A. Thorpe. *An Introduction to Ocean Turbulence*. Cambridge University Press, 2007.

[32] A.E. Gill. *Atmosphere-ocean Dynamics*. Number v. 30 in Atmosphere-ocean Dynamics. Academic Press, 1982.

[33] Harley Flanders. Differentiation under the integral sign. 80(6):615–627, 1973.

[34] Osbourne Reynolds. The sub-mechanics of the universe. In *Papers on Mechanical and Physical Subjects*, volume 3. Cambridge University Press, 1903.

[35] T. Frankel. *The Geometry of Physics: An Introduction*. Cambridge University Press, 2004.

[36] P.K. Kundu, I.M. Cohen, and D.R. Dowling. *Fluid Mechanics*. Elsevier Science, 2011.

[37] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics*. Number v. 6. Elsevier Science, 2013.

[38] Kenneth V. Mackenzie. NineâĂŘterm equation for sound speed in the oceans. *The Journal of the Acoustical Society of America*, 70(3), 1981.

[39] G. Gross. *Oceanography.* Merrill Physical Science. C. E. Merrill Publishing Company, 1976.

[40] B. Cushman-Roisin. *Introduction to Geophysical Fluid Dynamics.* Prentice Hall, 1994.

[41] D.R. Durran. *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics.* J.E.Marsden and others. Springer, 1999.

[42] S.T. Thornton and J.B. Marion. *Classical Dynamics of Particles and Systems.* Brooks/Cole, 2004.

[43] D. Lovelock and H. Rund. *Tensors, Differential Forms, and Variational Principles.* Dover Books on Mathematics Series. Dover, 1989.

[44] V.I. Arnold and B.A. Khesin. *Topological Methods in Hydrodynamics.* Applied Mathematical Sciences. Springer, 1998.

[45] R. Hooke. *Micrographia.* Observation LVIII. Royal Society of London, 1665.

[46] G.S. Settles. *Schlieren and Shadowgraph Techniques: Visualizing Phenomena in Transparent Media.* Engineering online library. Springer Berlin Heidelberg, 2001.

[47] D. E. Mowbray and B. S. H. Rarity. A theoretical and experimental investigation of the phase configuration of internal waves of small amplitude in a density stratified liquid. *Journal of Fluid Mechanics*, 28:1–16, 4 1967.

[48] R. Burden and J. Faires. *Numerical Analysis.* Cengage Learning, 2004.

[49] Daniel F. Martin. *An Adaptive Cell–Centered Projection Method for the Incompressible Euler Equations.* PhD thesis, University of California at Berkeley, 1998.

[50] Daniel F. Martin, Phillip Colella, and Daniel Graves. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. *J. Comput. Phys.*, 227(3):1863–1886, January 2008.

[51] Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. A conservative adaptive projection method for the variable density incompressible navierâĂŞstokes equations. *Journal of Computational Physics*, 142(1):1 – 46, 1998.

[52] Ralph Shapiro. Smoothing, filtering, and boundary effects. *Reviews of Geophysics*, 8(2):359–387, 1970.

[53] Sanjiva K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103(1):16 – 42, 1992.

[54] David L. Brown, Ricardo Cortez, and Michael L. Minion. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168(2):464–499, 2001.

[55] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics.* Springer-Verlag, 3rd edition, December 2001.

[56] James W. Demmel. *Applied Numerical Linear Algebra.* SIAM, aug 1997.

[57] O.B. Fringer, M. Gerritsen, and R.L. Street. An unstructured-grid, finite-volume, nonhydrostatic, parallel coastal ocean simulator. *Ocean Modelling*, 14(3-4):139–173, 2006.

[58] James Binney and Scott Tremaine. *Galactic Dynamics.* Princeton University Press, 2nd edition, January 1988.

[59] S. Majid Hassanizadeh. On the transient non-fickian dispersion theory. *Transport in Porous Media*,

23:107–124, 1996.

[60] Prateek Sharma and Gregory W. Hammett. Preserving monotonicity in anisotropic diffusion. *Journal of Computational Physics*, 227(1):123–142, 2007.

[61] Steven A. Cummer, Bogdan-Ioan Popa, David Schurig, David R. Smith, John Pendry, Marco Rahm, and Anthony Starr. Scattering theory derivation of a 3d acoustic cloaking shell. *Phys. Rev. Lett.*, 100(2):024301, Jan 2008.

[62] D. R. Smith, D. Schurig, and J. J. Mock. Characterization of a planar artificial magnetic metamaterial surface. *Phys. Rev. E*, 74(3):036604, Sep 2006.

[63] V. Armenio and F. Roman. Large eddy simulation of environmental shallow water coastal flows. *ERCOFTAC Bulletin*, 78:46–53, 2009.

[64] P. Degond, F. Deluzet, A. Sangam, and M.-H. Vignal. An asymptotic preserving scheme for the euler equations in a strong magnetic field. *Journal of Computational Physics*, 228(10):3540–3558, 2009.

[65] P. Degond, F. Deluzet, and C. Negulescu. An asymptotic preserving scheme for strongly anisotropic elliptic problems. *arXiv:0903.4984v2*, March 2009.

[66] P. Degond, A. Lozinski, J. Narski, and C. Negulescu. An Asymptotic-Preserving method for highly anisotropic elliptic equations based on a micro-macro decomposition. *arXiv:1102.0904v1*, February 2011.

[67] Alberto Scotti and Sorin Mitran. An approximated method for the solution of elliptic problems in thin domains: Application to nonlinear internal waves. *Ocean Modelling*, 25(3-4):144–153, 2008.

[68] P. Colella, D. T. Graves, J. N. Johnson, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. https://apdec.org/designdocuments/ChomboDoc/ChomboDesign/chomboDesign.pdf, 2011.

[69] A. M. Bruaset and A. Tveito. A numerical study of optimized sparse preconditioners. *BIT Numerical Mathematics*, 34:177–204, 1994. 10.1007/BF01955867.

[70] Keita Teranishi and Padma Raghavan. A hybrid parallel preconditioner using incomplete cholesky factorization and sparse approximate inversion. In Timothy J. Barth, Michael Griebel, David E. Keyes, Risto M. Nieminen, Dirk Roose, Tamar Schlick, Olof B. Widlund, and David E. Keyes, editors, *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 755–762. Springer Berlin Heidelberg, 2007.

[71] K. Lipnikov, M. Shashkov, D. Svyatskiy, and Yu. Vassilevski. Monotone finite volume schemes for diffusion equations on unstructured triangular and shape-regular polygonal meshes. *Journal of Computational Physics*, 227(1):492–512, 2007.

[72] J. Boussinesq. Théorie des ondes et des remous qui se propagent le long d'un canal rectangulaire horizontal, en communiquant au liquide contenu dans ce canal des vitesses sensiblement pareilles de la surface au fond. *Journal de Mathématique Pures et Appliquées*, pages 55–108, 1872.

[73] Tony F. Chan and Howard C. Elman. Fourier analysis of iterative methods for elliptic problems. *SIAM Review*, 31(1):20–49, 1989.

[74] Henk A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, jun 2003.

[75] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen

Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[76] Edward Santilli and Alberto Scotti. The stratified ocean model with adaptive refinement (somar). *Journal of Computational Physics*, 291(0):60 – 81, 2015.

[77] J. M. Stone, T. A. Gardiner, P. Teuben, J. F. Hawley, and J. B. Simon. Athena: A New Code for Astrophysical MHD. *Astrophysical Journal, Supplement*, 178:137–177, September 2008.

[78] A. S. Almgren, V. E. Beckner, J. B. Bell, M. S. Day, L. H. Howell, C. C. Joggerst, M. J. Lijewski, A. Nonaka, M. Singer, and M. Zingale. CASTRO: A New Compressible Astrophysical Solver. I. Hydrodynamics and Self-gravity. *Astrophysical Journal*, 715:1221–1238, June 2010.

[79] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35:2298–2316, 1998.

[80] Enzo 2.4 documentation - Hydro and MHD methods. `https://enzo.readthedocs.org/en/latest/physics/hydro_methods.html`. Accessed: 2015-04-09.

[81] P. Colella and P. R. Woodward. The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. *Journal of Computational Physics*, 54:174–201, September 1984.

[82] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, and B. Van Straalen. AMR Godunov unsplit algorithm and implementation. Unpublished manuscript, 2008.

[83] M. Sekora and P. Colella. Extremum-Preserving Limiters for MUSCL and PPM. *ArXiv e-prints*, March 2009.

[84] John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 85(2):257 – 283, 1989.

[85] David L. Brown, Ricardo Cortez, and Michael L. Minion. Accurate projection methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 168(2):464 – 499, 2001.

[86] Robert D. Guy and Aaron L. Fogelson. Stability of approximate projection methods on cell-centered grids. *Journal of Computational Physics*, 203(2):517 – 538, 2005.

[87] Bruce Sutherland. *Internal gravity waves*. Cambridge University Press, 2010.

[88] C. Canuto. *Spectral methods in fluid dynamics*. Springer series in computational physics. Springer-Verlag, 1988.

[89] A. Scotti and S. Mitran. An approximated method for the solution of elliptic problems in thin domains: Application to nonlinear internal waves. *Ocean Modelling*, 25(3âĂŞ4):144 – 153, 2008.

[90] der V. van. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *Society for Industrial and Applied Mathematics.SIAM Journal on Scientific and Statistical Computing*, 13(2):631–14, 03 1992. Copyright - Copyright] ÂĬ 1992 Society for Industrial and Applied Mathematics; Last updated - 2012-02-17.

[91] M. Jalali, N. Rapaka, and S. Sarkar. Tidal flow over topography: effect of excursion number on wave energetics and turbulence. *Journal of Fluid Mechanics*, 750:259–283, 2014.

[92] Narsimha R. Rapaka, Bishakhdatta Gayen, and Sutanu Sarkar. Tidal conversion and turbulence at a model ridge: direct and large eddy simulations. *Journal of Fluid Mechanics*, 715:181–209, 1 2013.

[93] Robert R Long. Some aspects of the flow of stratified fluids: I. a theoretical investigation. *Tellus*, 5(1):42–58, 1953.

[94] M. L. Dubreil-Jacotin. Sur la determination rigoureuse des ondes permanentes periodiques d'amplitude finie. *J. Math. Pure Appl.*, 13:217–291, 1934.

[95] Chia-Shun Yih. Exact solutions for steady two-dimensional flow of a stratified fluid. *Journal of Fluid Mechanics*, 9(02):161–174, 1960.

[96] Marek Stastna and Kevin G. Lamb. Large fully nonlinear internal solitary waves: The effect of background current. *Physics of Fluids (1994-present)*, 14(9):2987–2999, 2002.

[97] M. Dunphy, C. Subich, and M. Stastna. Spectral methods for internal waves: indistinguishable density profiles and double-humped solitary waves. *Nonlinear Processes in Geophysics*, 18(3):351–358, 2011.